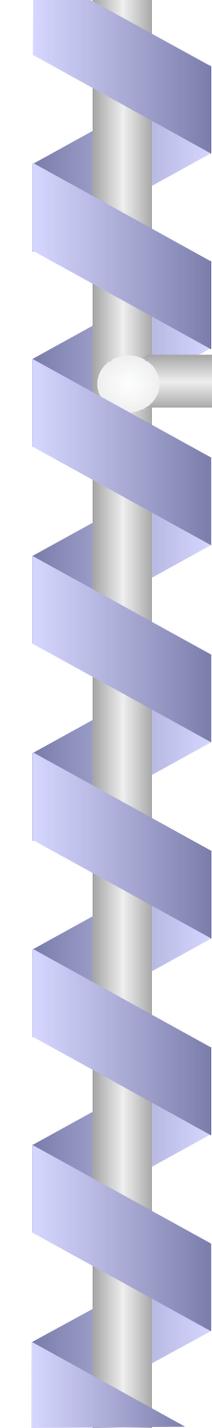




Universidade de São Paulo
Instituto de Ciências Matemáticas e de Computação
SSC 124: Análise e Projeto Orientados a Objetos

Introdução à Orientação a Objetos e UML

Profa. Dra. Elisa Yumi Nakagawa
2º semestre de 2016



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

Fases dos Modelos de Processo de Software

DEFINIÇÃO

Análise de Sistema

Planejamento

Análise de Requisitos

CONSTRUÇÃO

Projeto

Codificação

Teste

MANUTENÇÃO

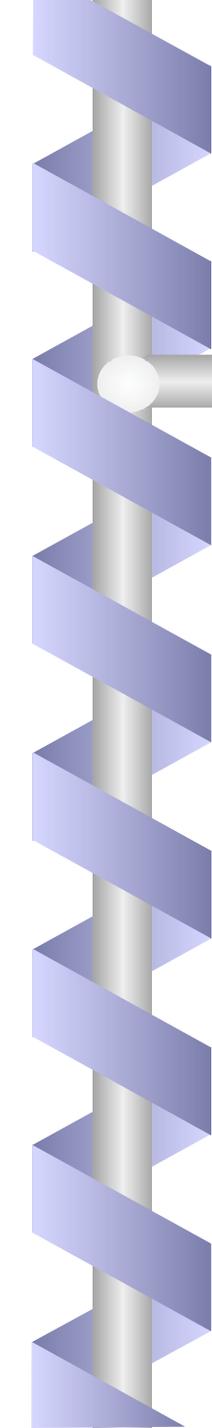
Entendimento

Modificação

Revalidação

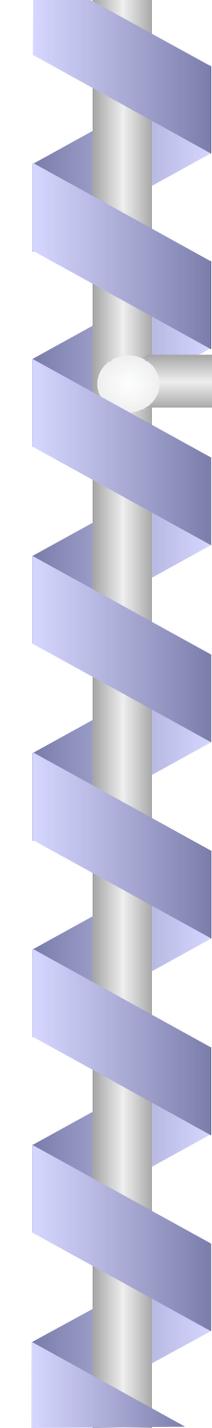
ATIVIDADES DE APOIO

- *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*



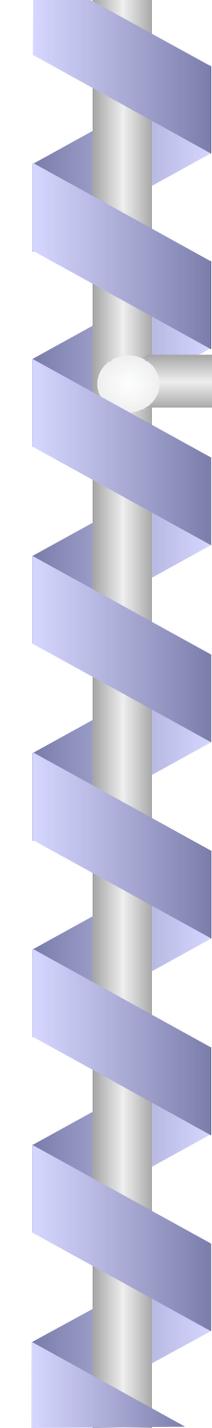
Introdução

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



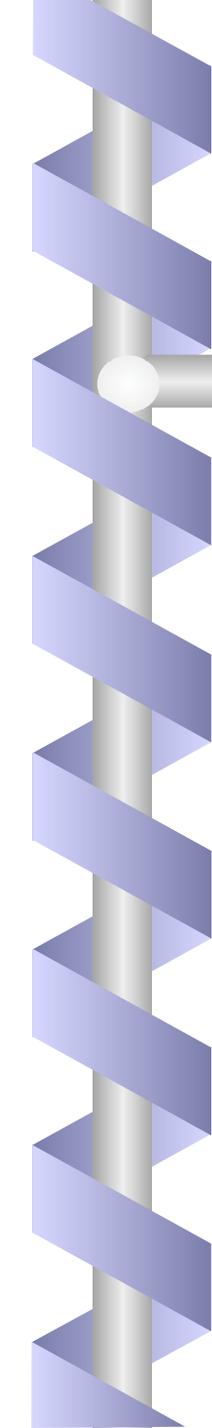
Perguntas

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



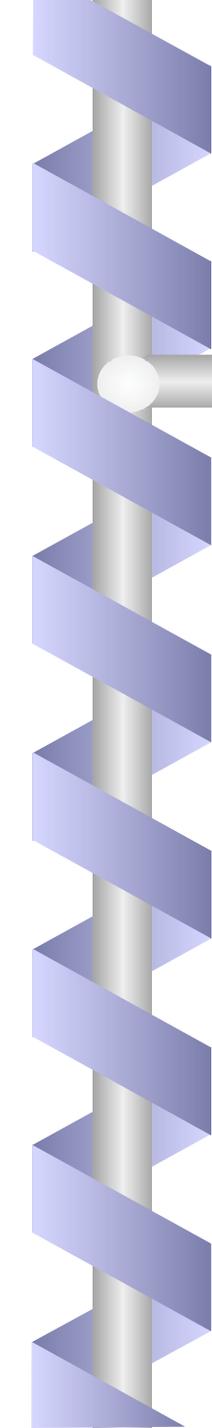
Histórico de OO

- **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**



Histórico de OO

- **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), Python (~1990), Perl 5 (2005)**

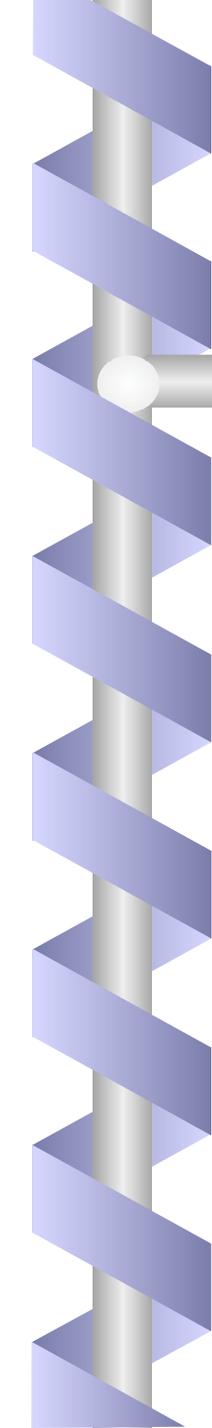


Linguagens orientadas a objetos

- **“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.**

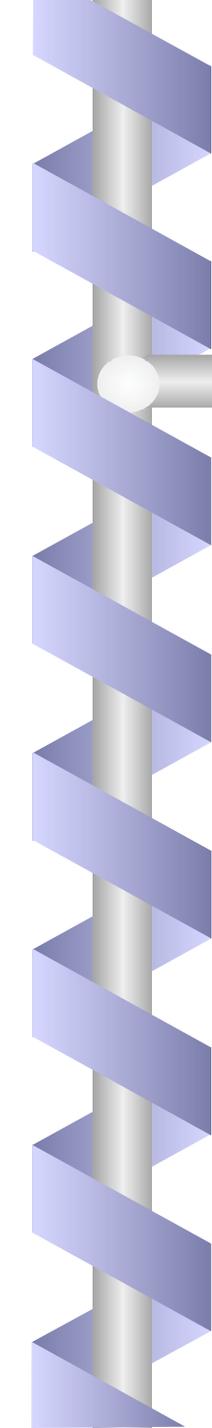
Histórico de OO

- Surgiram vários **métodos de análise e projeto OO**
 - **CRC (*Class Responsibility 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**



Pergunta

- **Porque estudaremos a UML?**



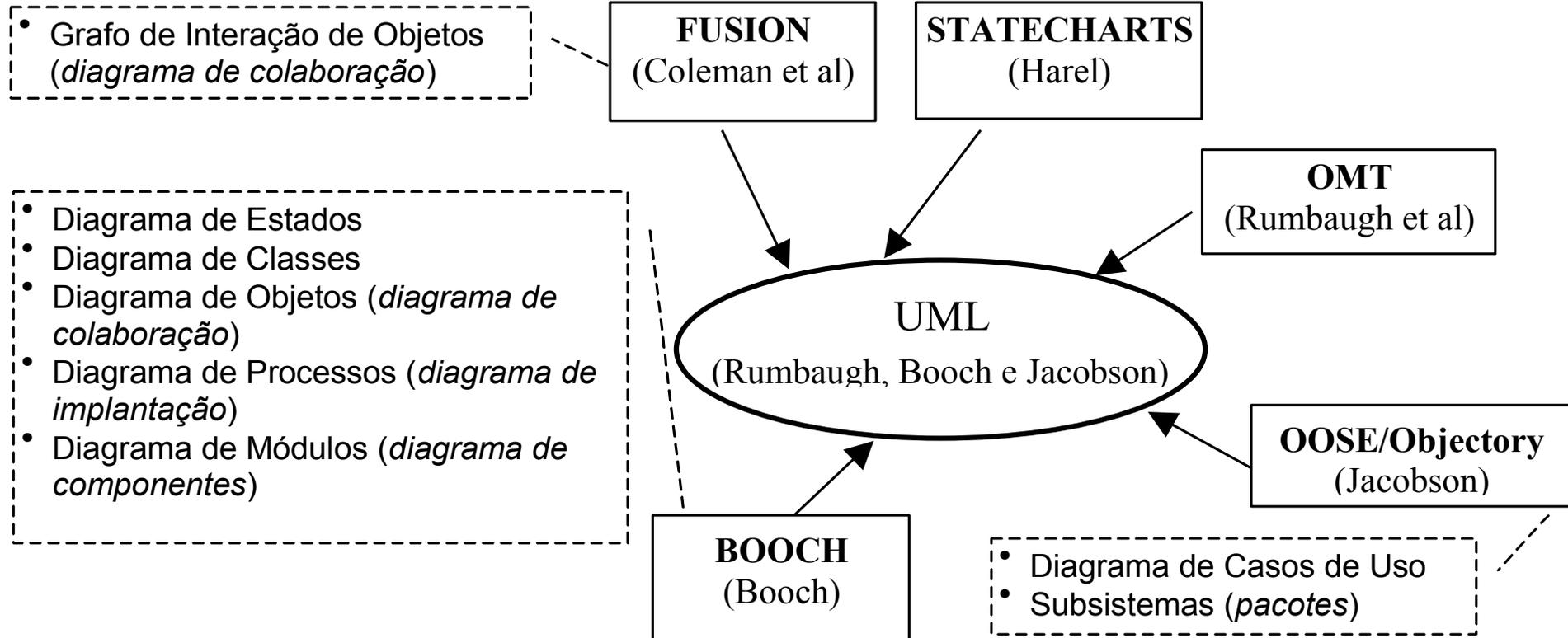
Introdução à UML

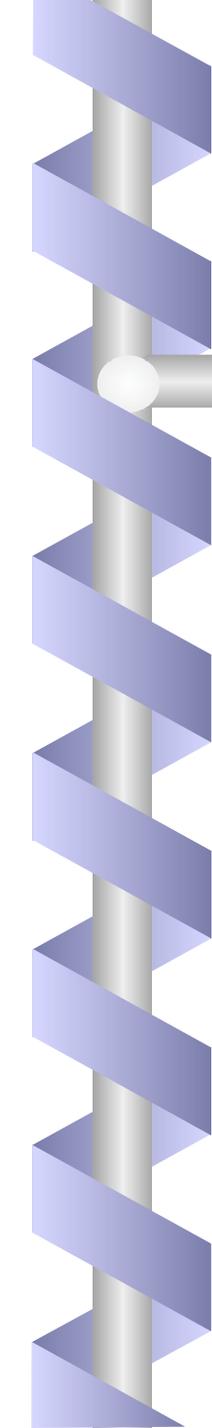
- **UML (*Unified Modelling Language*)**
 - **Linguagem para especificação, construção, visualização e documentação de sistemas.**
 - **Evolução das linguagens para especificação de conceitos de *Booch*, OMT e OOSE e também de outros métodos de especificação de requisitos de software OO ou não.**

Histórico da UML

- **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: UML





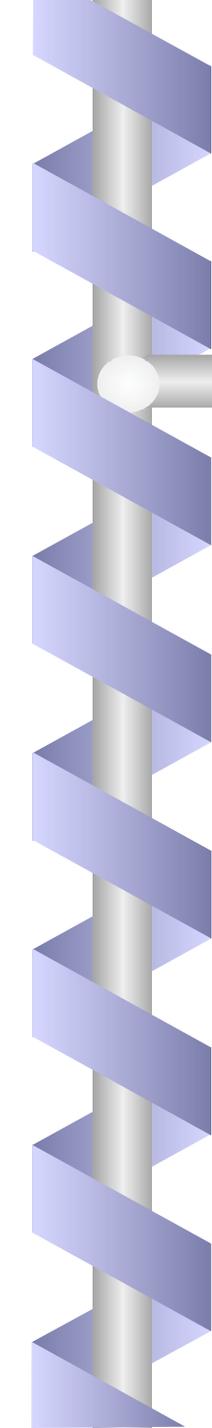
Ferramentas de Apoio

- 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:**

- **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>

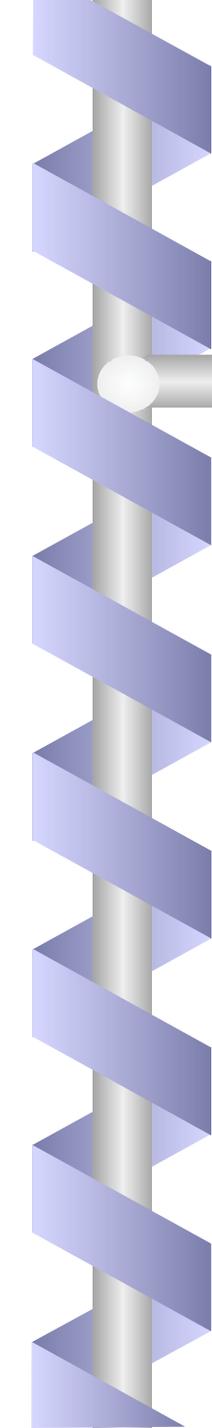


Diagramas Mais Comumente Usados da UML

- **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*)**

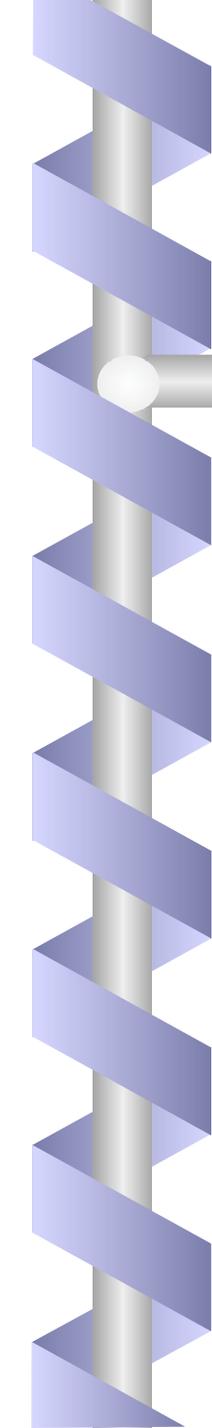
Diagramas

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
	Perfil
	Estrutura Composta



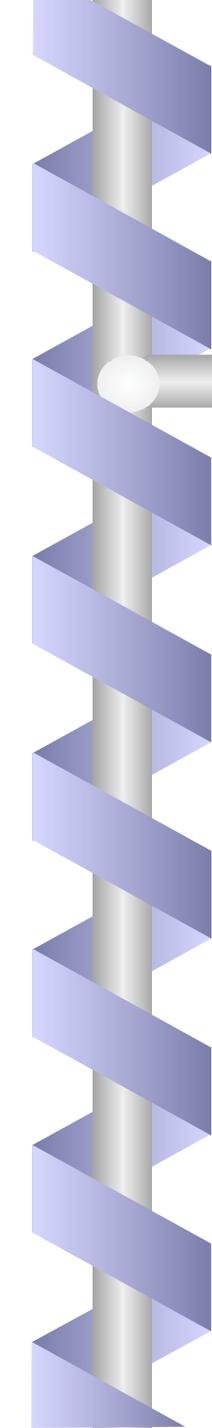
Pergunta

- **UML serve somente para modelar sistemas?**
- **UML somente serve para sistemas OO?**



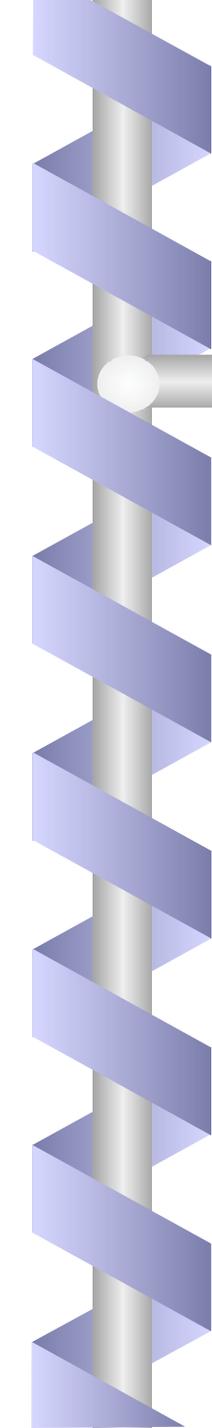
Conceitos Básicos: OO

- **Orientação a Objetos (OO) é uma abordagem de programação que procura explorar nosso lado intuitivo.**



Objetos

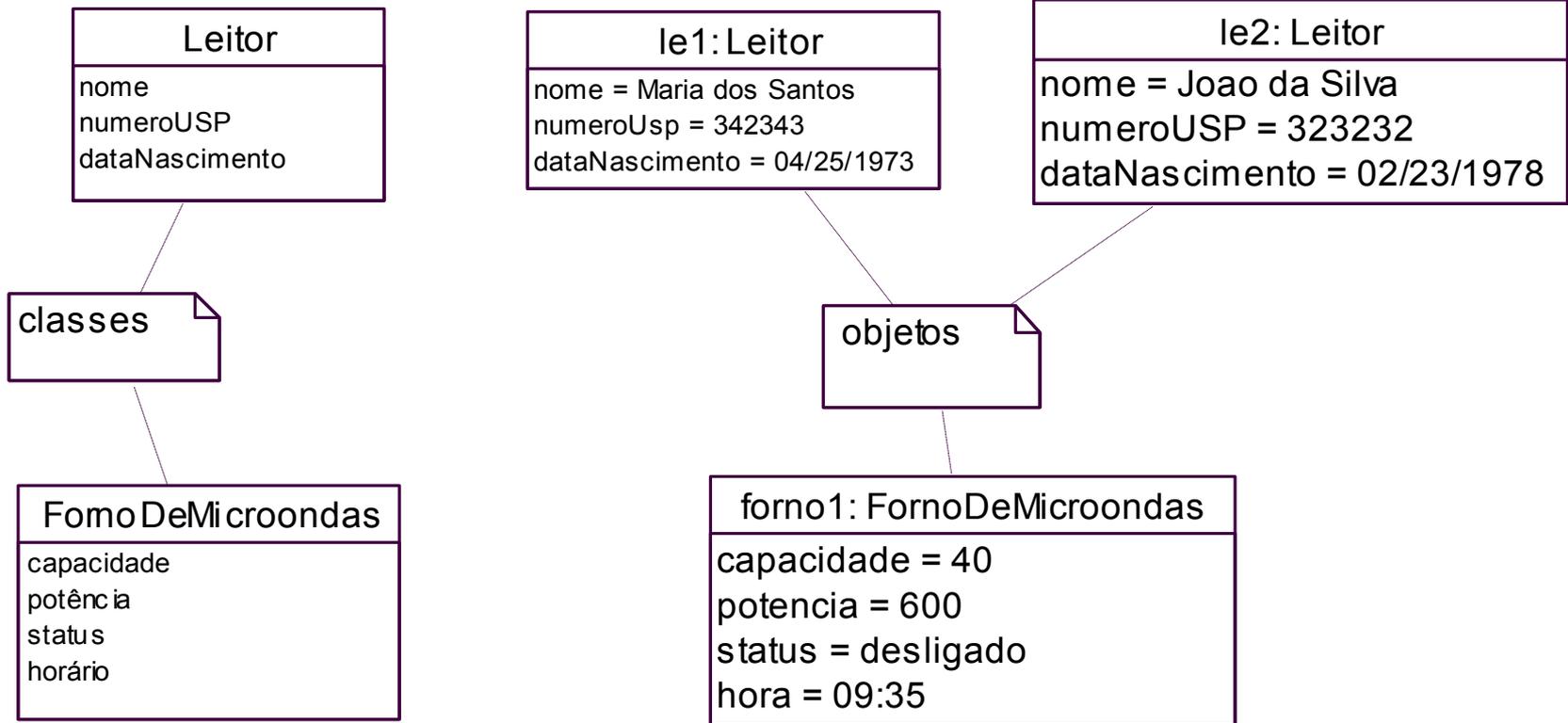
- **Objeto no mundo físico é tipicamente um produtor e consumidor de itens de informação**
- **Objeto no mundo do software:**
 - ***“Qualquer coisa, real ou abstrata, a respeito da qual armazenamos dados e métodos que os manipulam”*** [Martin e Odell, 1995]
- **Objetos da computação são análogos aos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

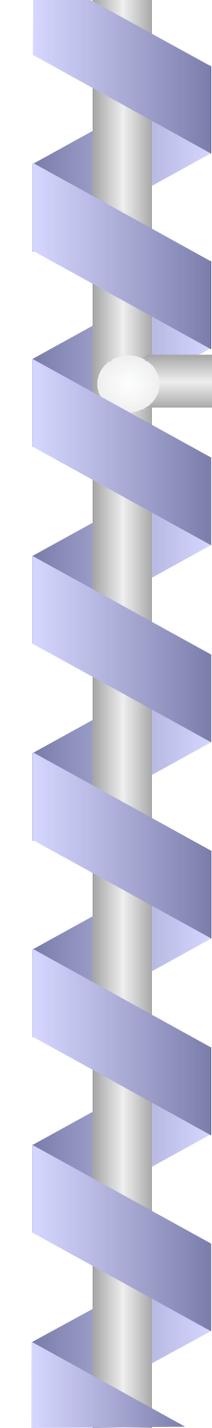


Classes

- **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.**

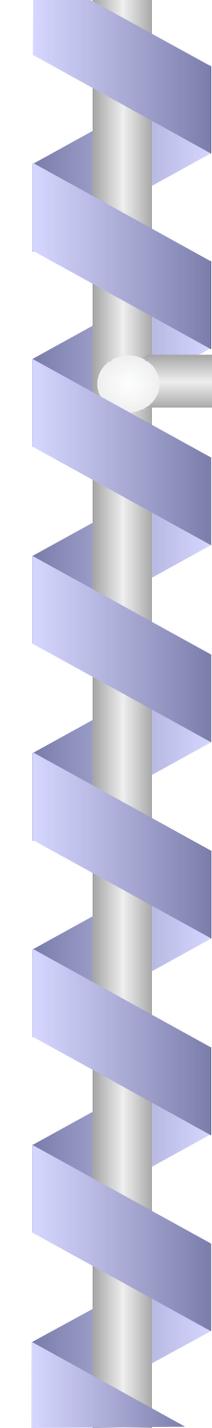
Objetos e Classes





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

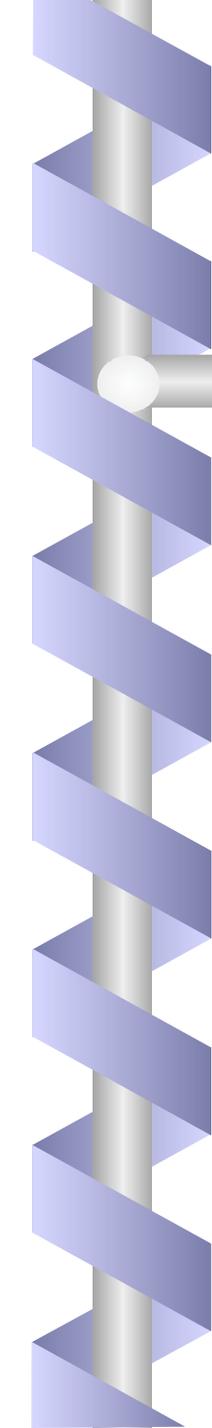
Automóvel
proprietário marca placa ano
registrar transferir_Proprietário mudar_Placa



Atributos

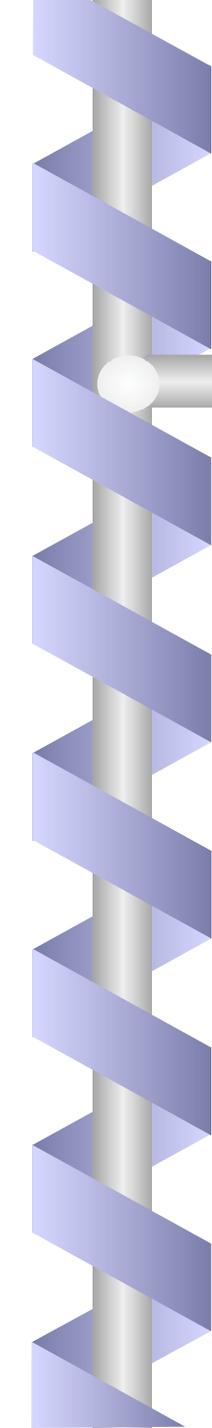


Métodos



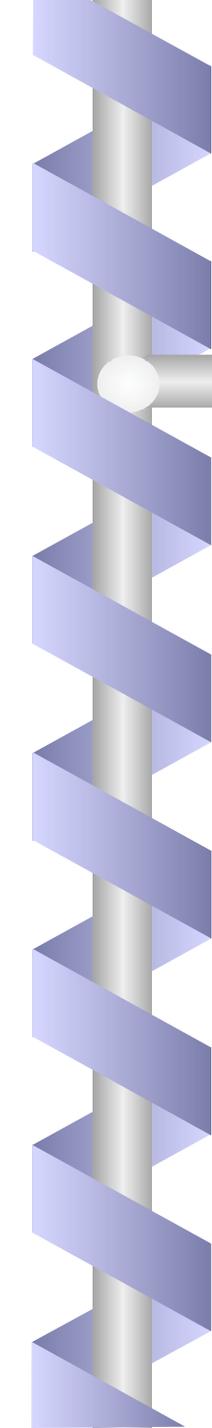
Abstração

- **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**



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 reúso**

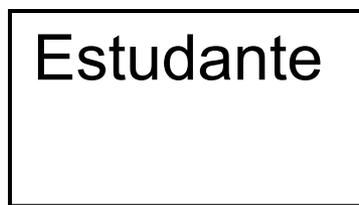


Herança

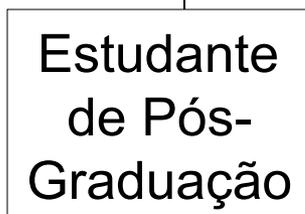
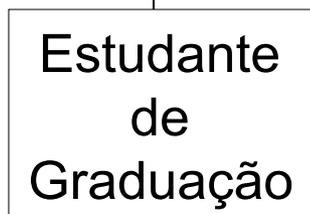
- Mecanismo que permite que características comuns a diversas classes sejam colocadas em uma classe base, ou **superclasse**.
- As propriedades da superclasse não precisam ser repetidas em cada **subclasse**.
- Por exemplo, *JanelaRolante* e *JanelaFixa* são subclasses de *Janela*. Elas herdam as propriedades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

Herança: Generalização/ Especialização

Superclasse →



Subclasse ↘

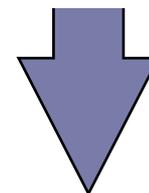


GENERALIZAÇÃO

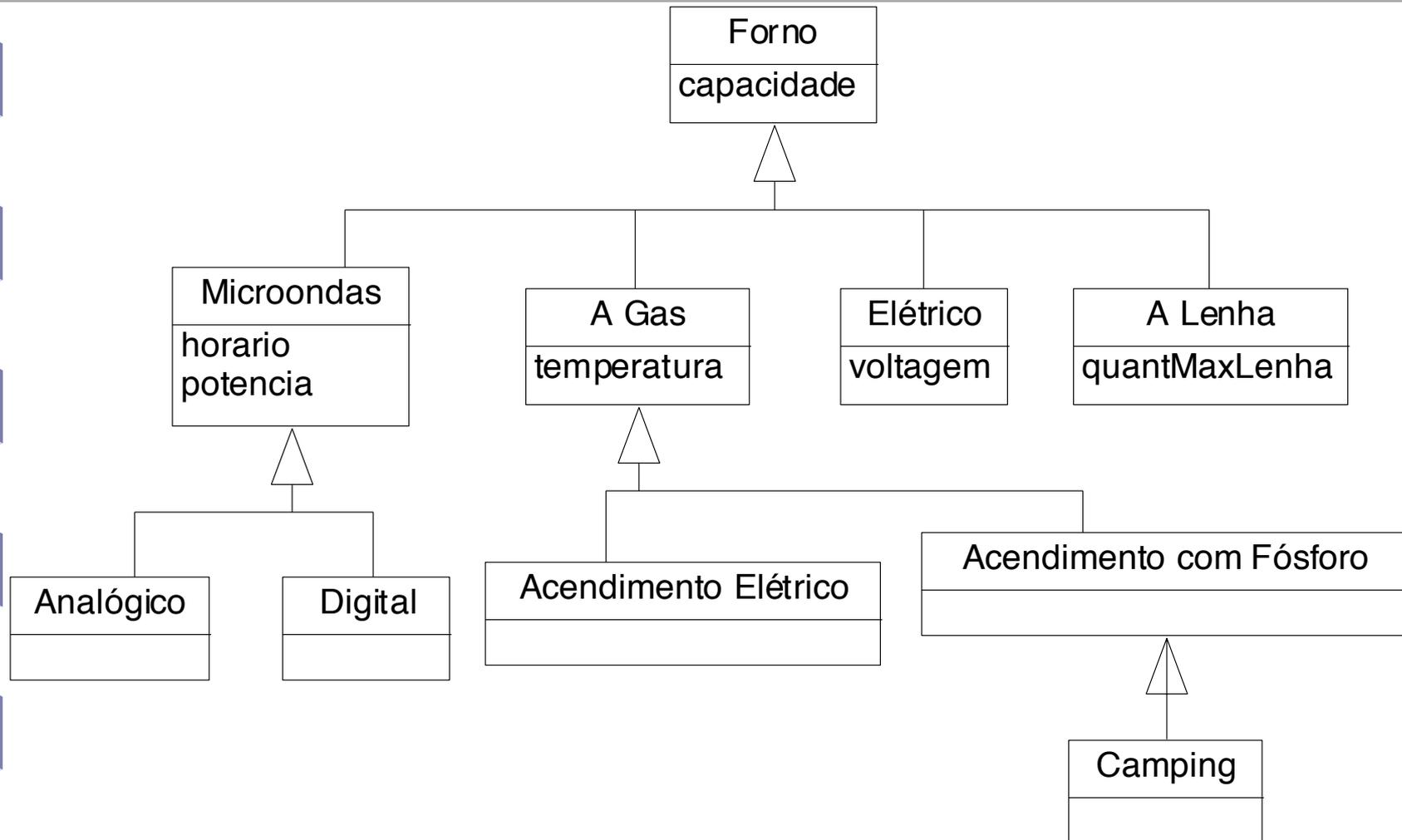


ESPECIALIZAÇÃO

(herança)

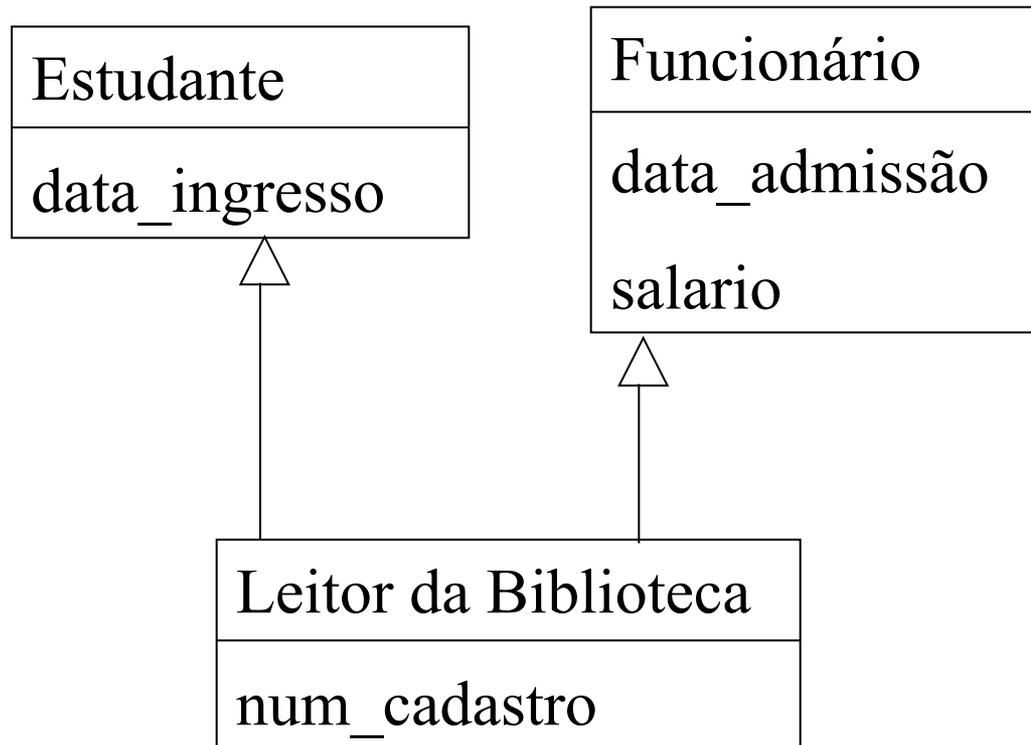


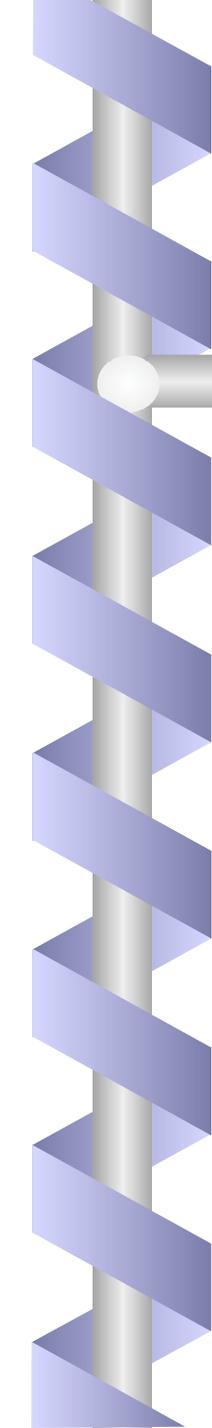
Herança



Herança Múltipla

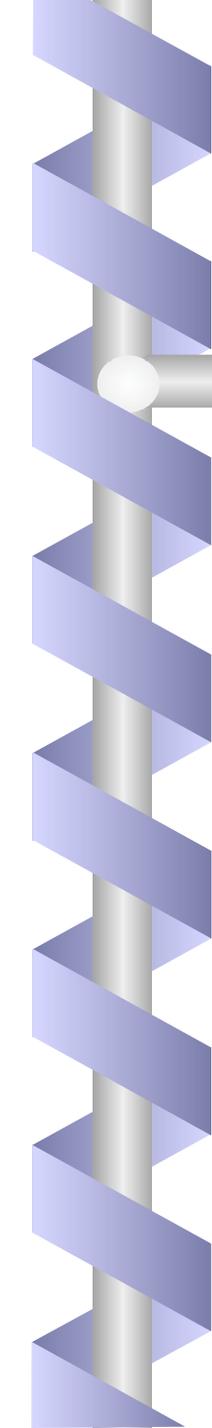
Existe mais de uma superclasse, ou seja, uma classe é declarada como uma subclasse de uma ou mais superclasses.





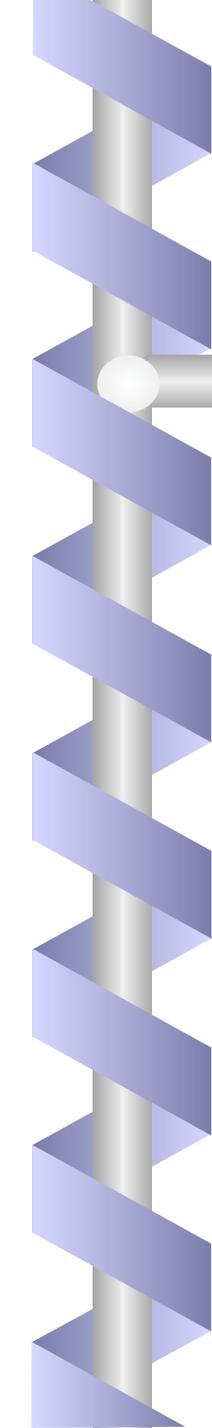
Vantagens de OO

- **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

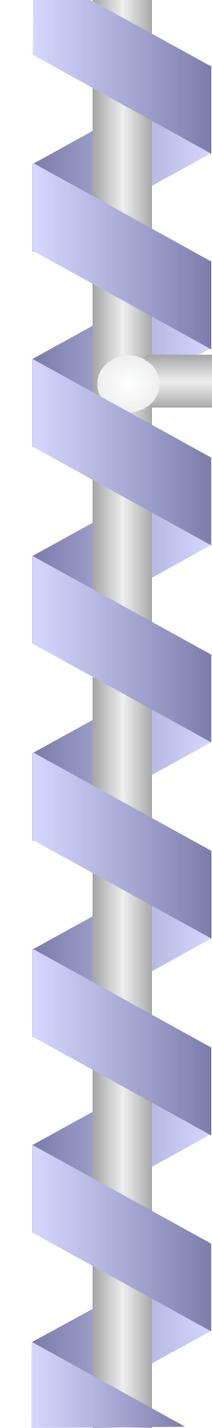
- **reutilização**: o encapsulamento dos métodos e representação dos dados para a construção de classes facilitam o desenvolvimento de software reutilizável, auxiliando na produtividade de sistemas;
- **extensibilidade**: facilidade de estender o software devido a duas razões:
 - herança: novas classes 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.



Material sobre UML

- **<http://www.omg.org/spec/UML/2.5> (*Object Management Group*) Março, 2015**
- **Larman, C.; *Applying UML and Patterns*, 3rd. Ed., Prentice Hall, 2004.**
- **<http://www-01.ibm.com/software/rational/> (**Rational**)**