ACH2002

Orientação a Objetos

Delano Medeiros Beder

EACH – USP

O que é abstração ?

Abstração

- Abstração - mecanismo utilizado na análise de um domínio
- O indivíduo observa a realidade e dela abstrai:
  - entidades
  - ações
  - comportamentos
  - relacionamentos
- São considerados somente os elementos essenciais para uma aplicação
- Todos os aspectos julgados irrelevantes são excluídos

Abstração

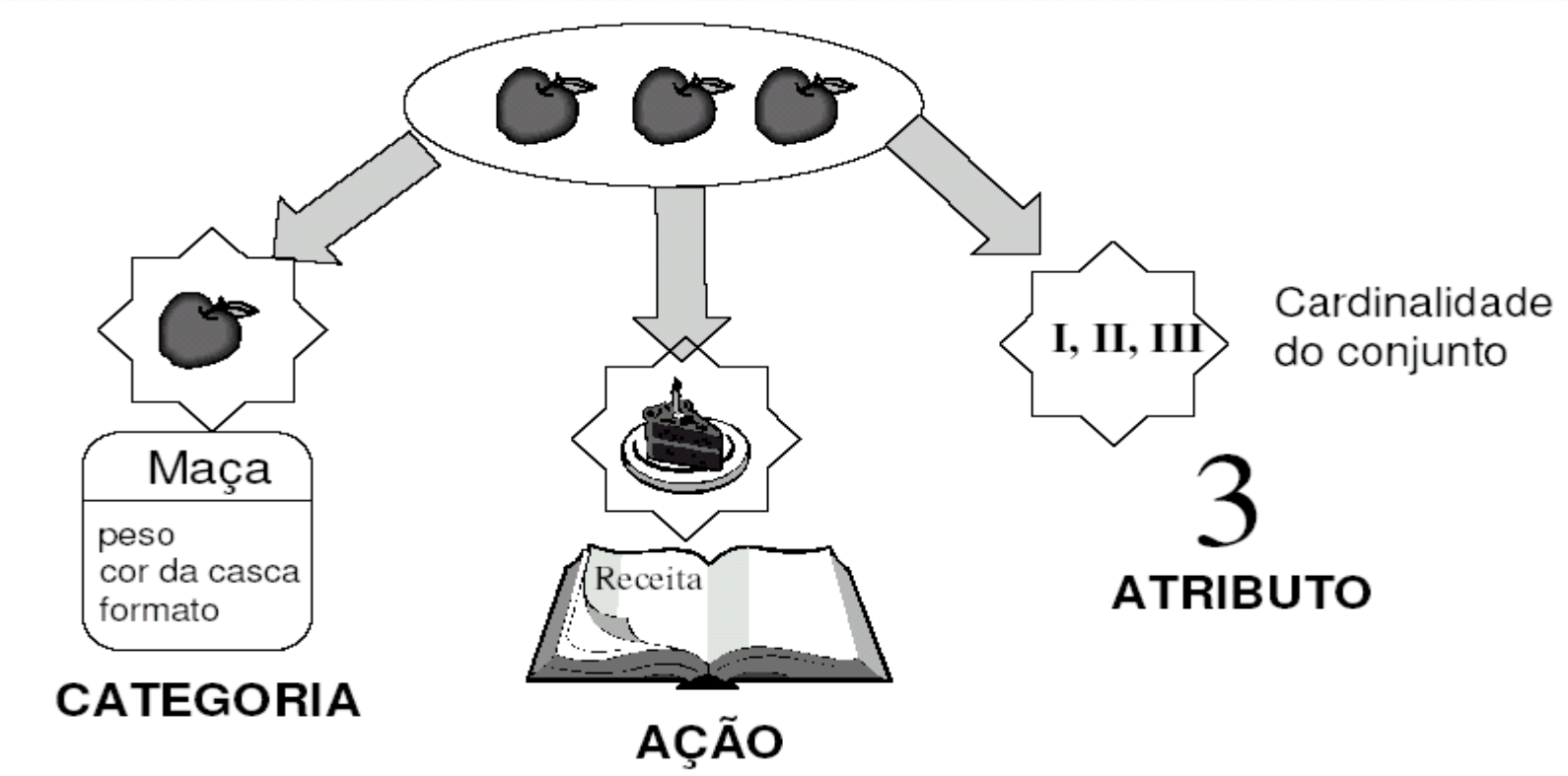
- A abstração consiste em focar em aspectos inerentes e essenciais de uma entidade
  - em desenvolvimento de software: focar em O QUE é e o QUE FAZ um objeto, e não como faz
  - evita comprometimento prematuro com detalhes

Paradigma OO

- Objeto:
  - abstração de uma entidade real, cujas características e comportamento são conhecidos
  - se apresenta a outras entidades por meio de uma interface bem definida
- Visão Interna: define a estrutura e o comportamento do objeto, ou seja, define dados e métodos (e suas implementações)
- Visão Externa: interface que define como o objeto é visto por outros objetos
- Mensagem: comunicação entre objetos

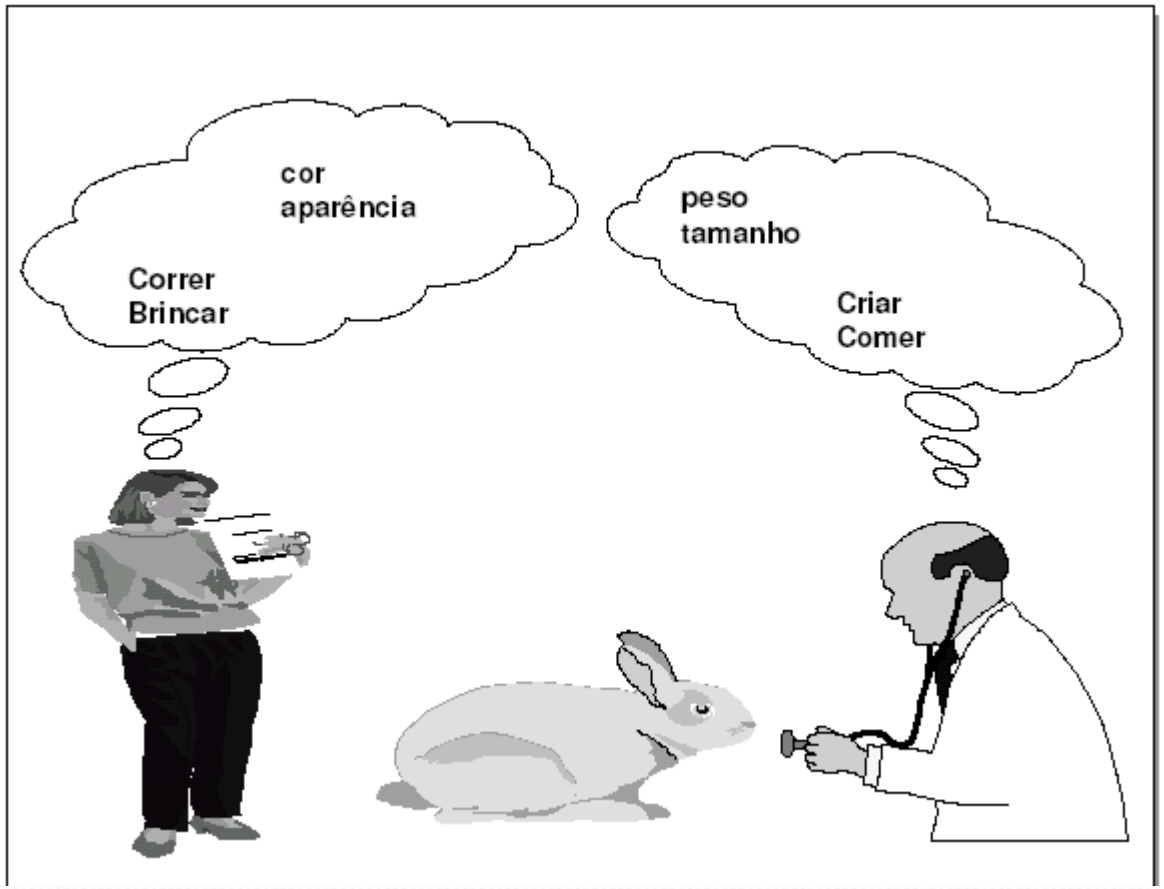
Abstração

- Diferentes abstrações a partir de um mesmo objeto do mundo real



Abstração

- Depende do ponto de vista



Tipos Abstratos de Dados

- Combinação de dados e operações (sobre eles) em um elemento único
  - Outras unidades de programação (chamadas de clientes) podem ter permissão para criar variáveis do tipo abstrato de dados

- Ocultamento da informação e implementação
  - A representação de objetos do tipo não é visível pelos clientes que usam o tipo, de modo que as únicas operações diretas sobre esses objetos são aquelas oferecidas na definição do tipo.

Tipos Abstratos de Dados

- Ocultamento da informação e implementação
  - Permite que possa alterar a implementação do tipo (contanto que mantenha as mesmas operações) sem afetar as unidades de programa que fazem uso dele
  - Aumenta a confiabilidade, pois nenhuma outra unidade de programa pode mudar, acidentalmente ou intencionalmente, as representações do tipo, aumentando a integridade de tais objetos

Orientação a Objetos

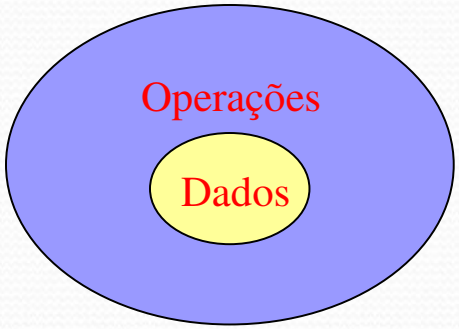
- Pode ser visto como uma extensão do conceito de Tipos Abstratos de Dados
  - Combinação de dados e operações (sobre eles) em um elemento único
- Classe: definição do Tipo Abstrato de Dados
- Objeto: cada instância derivada da classe
- Representa em software entidades que encontramos no mundo real

Orientação a objetos

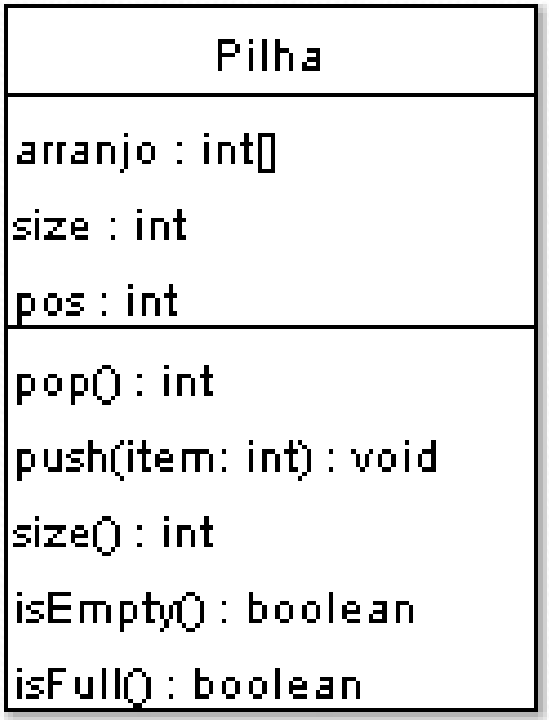
- Encapsulamento
- Ocultamento da informação e implementação (Abstração)
- Retenção de estado
- Identidade de objeto
- Mensagens
- Classes/Objetos
- Herança
- Polimorfismo/Vinculação tardia (*late binding*)
- Relacionamento entre classes/objetos

Encapsulamento

- Paradigmas convencionais separam dados e procedimentos
- O objeto contém tantos os dados quanto as operações:
  - Dados: atributos
  - Implementação das operações: métodos



Encapsulamento



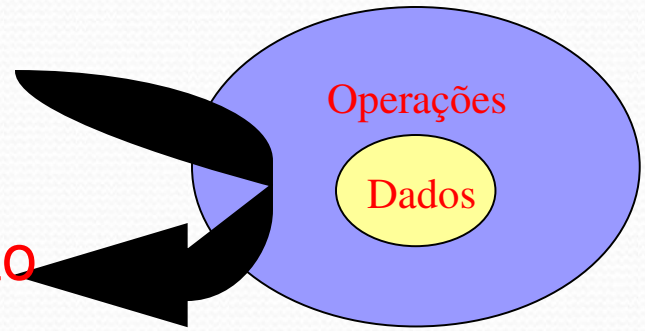
Dados (atributos)

Operações (métodos)

Diagrama de Classes UML

Encapsulamento e Ocultamento de Informações

- Utilização de encapsulamento para restringir a visibilidade externa de certos detalhes de informações (dados) ou implementações (operações), os quais são internos à estrutura de encapsulamento
- Não é possível chegar as dados diretamente
- Cliente não tem conhecimento acerca de como as operações são implementadas



Encapsulamento e Ocultamento de Informações

- **Encapsulamento**: Dados e os Métodos são "empacotados" em um objeto, de forma que objetos externos somente tenham acesso àquilo que for permitido pelo objeto
- O encapsulamento facilita o ocultamento de informações, separando aspectos externos de um objeto, que são acessíveis para outros objetos, dos detalhes internos de estrutura (dados) e implementação, que ficam "escondidos" de outros objetos
  - restringe a visibilidade do objeto, mas facilita reuso
  - atributos e métodos empacotados sob um só nome e podem ser reusados como uma especificação ou componente de programa

Encapsulamento e Ocultamento de Informações

- As estruturas de dados ou atributos das classes ficam encapsulados, ou seja, só podem ser manipuladas pelos próprios métodos das classes.
- A idéia por trás do encapsulamento é a de que a utilização dos objetos não deve depender de sua implementação interna, e sim de sua interface.

Retenção de Estado

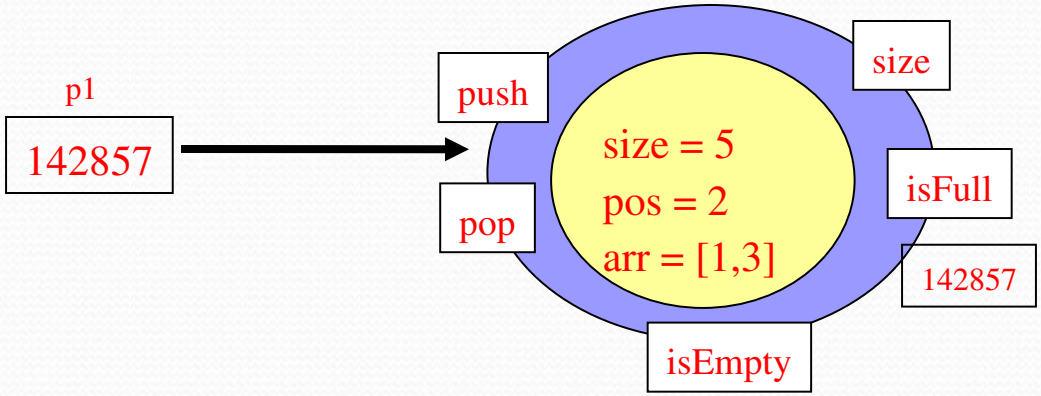
- Habilidade de um objeto reter seu estado
  - estado ≅ conjunto de valores de seus atributos
- Um objeto é ciente de seu passado (operações que foram executadas previamente)
- O estado influencia o comportamento do objeto
  - método pop() retorna o último item inserido através do método push()
  - método size() retorna valores diferentes antes e após a execução de um push() ou pop()

Identidade de Objeto

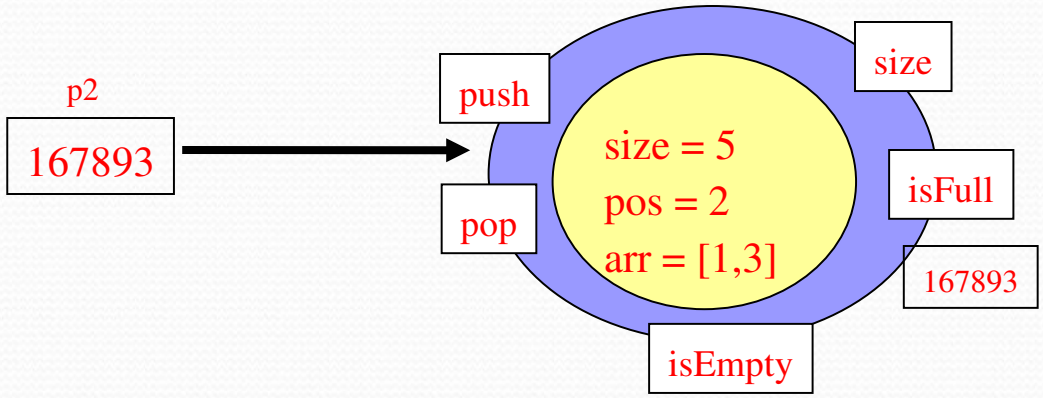
- Propriedade pela qual cada objeto (independente de sua classe ou seu estado) pode ser identificado e tratado como uma entidade distinta de software
  - O mesmo identificador permanece com o objeto por toda sua vida
  - Dois objetos nunca podem ter o mesmo identificador
  - Dois objetos podem ter o mesmo “estado”, porém suas identidades são distintas

Referência para objetos

```
Pilha p1 = new Pilha(5);
p1.push(1);
p1.push(3);
```

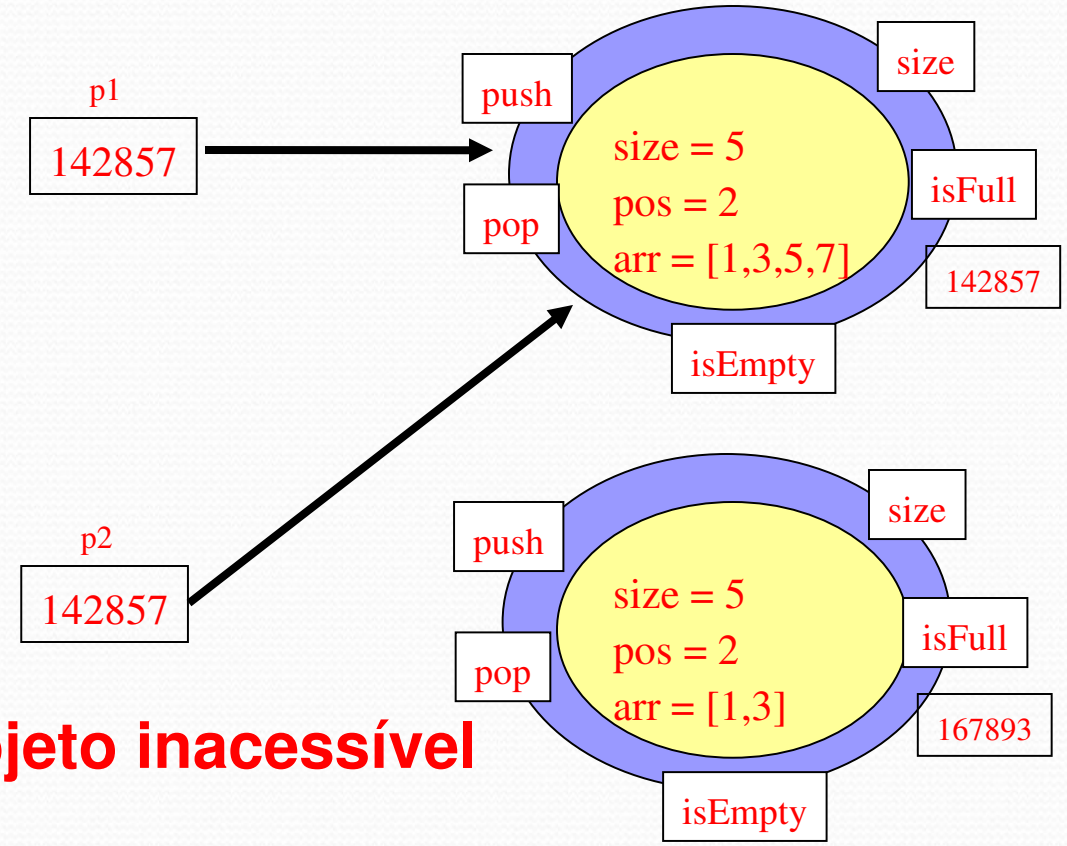



```
Pilha p2 = new Pilha(5);
p2.push(1);
p2.push(3);
```



Referência para objetos

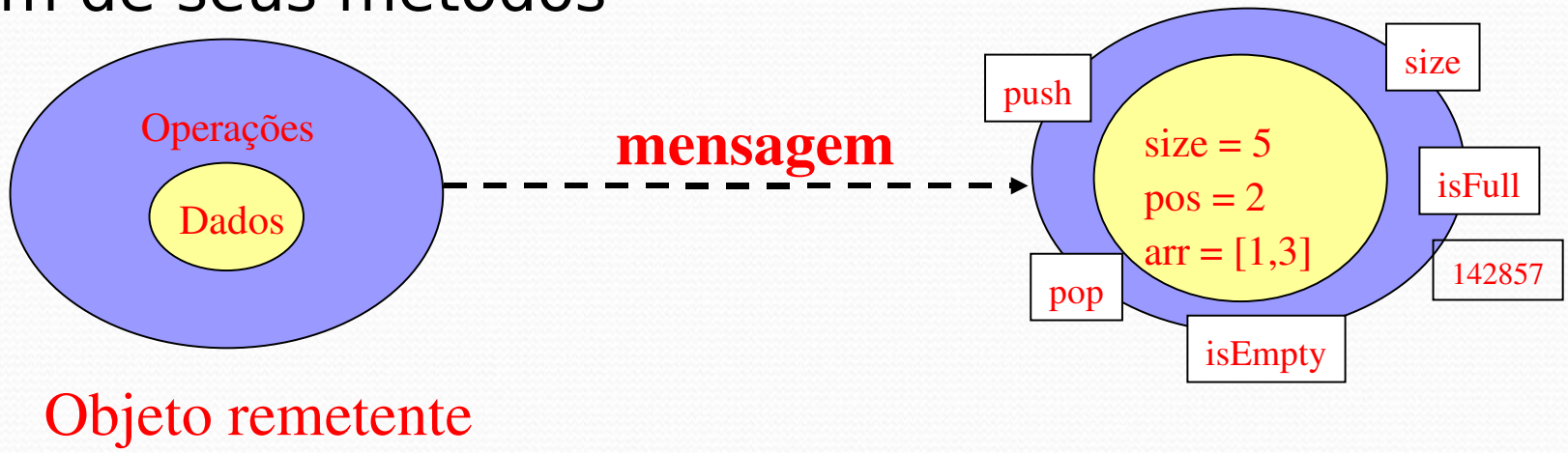
```
p2 = p1;
p1.push(5);
p2.push(7);
```



Objeto inacessível

Mensagens

- Um objeto pode se comunicar com outros através da troca de mensagens
- Uma mensagem é o veículo pelo qual um objeto remetente **obj1** transmite a um objeto destinatário **obj2** um pedido para o **obj2** aplicar um de seus métodos



Classes / Objetos

- Classe: um conjunto de objetos com a mesma estrutura e o mesmo comportamento
  - Pessoa, Automóvel, Carro e Pilha são exemplos de classes
  - João e Maria são dois objetos da classe Pessoa
- A classe é que você projeta e programa
- O objeto é o que você cria (a partir de uma classe) em tempo de execução

Classes / Objetos

- Todo objeto é criado a partir (é uma instância) de uma classe
- Dados (atributos) são associados a cada objeto
- Classes podem conter atributos gerais (e operações) independentes das diversas instâncias
  - Métodos e atributos estáticos (C++ e Java)

Objetos

- Exemplo: um Estudante e uma Disciplina podem ser considerados objetos, pois qualquer um deles pode ser definido em termos de um conjunto de atributos e operações
- Objetos podem se relacionar um com o outro
  - um Estudante pode Cursar uma Disciplina
  - o relacionamento Cursar define uma conexão específica entre Estudante e Disciplina

Objetos

Objeto = Atributos + Métodos + Encapsulamento

dados

funcionalidade (comportamento)

- Outros exemplos de objetos
  - um computador
  - uma tela de um aplicativo do computador
  - uma reserva de livro
  - um livro
  - um processo de identificação de acessos ilegais numa rede de computadores
  - um acesso ilegal

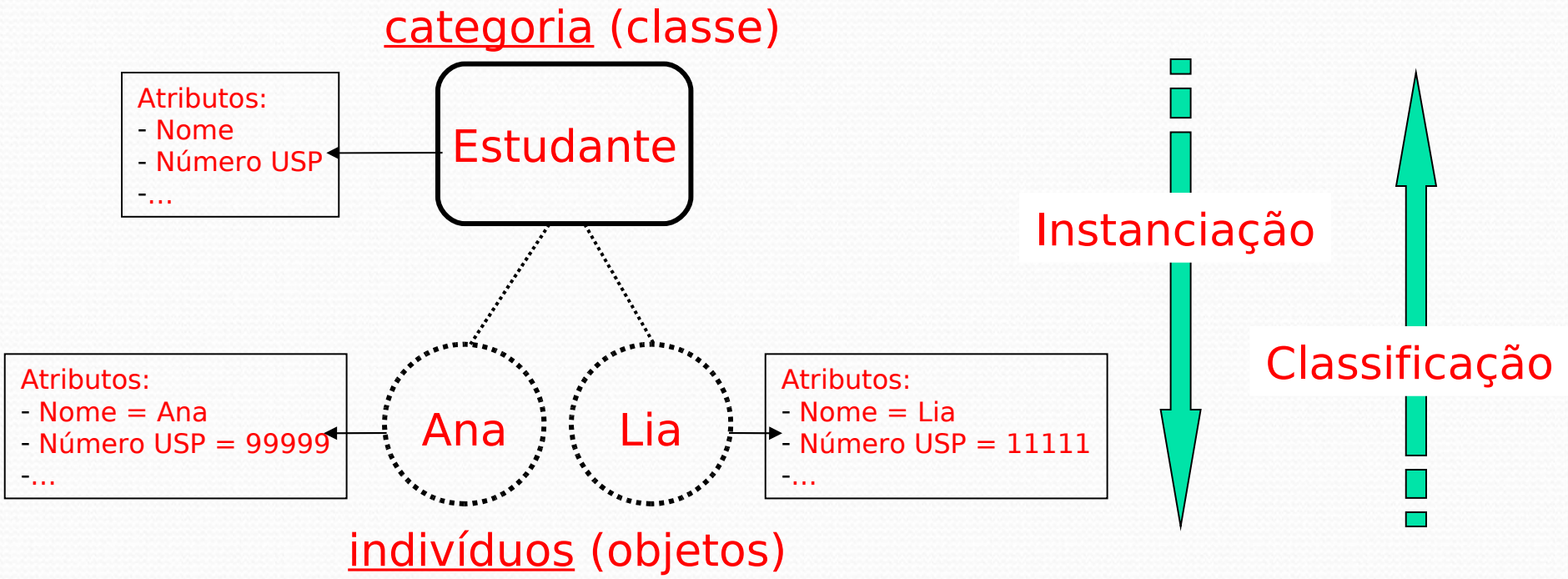
Atributos

- Representam um conjunto de informações, ou seja, elementos de dados que caracterizam um objeto
- Exemplos:
  - objeto Estudante
    - atributos: nome, nro USP, ano de ingresso, data de nascimento, ...
  - objeto Disciplina
    - atributos: nome, código, nro de créditos, pré-requisitos, equivalências, ...
  - objeto Reserva de Vôo
    - atributos: número da reserva, número do vôo, nome passageiro, data validade da reserva, prioridade, ...

Métodos

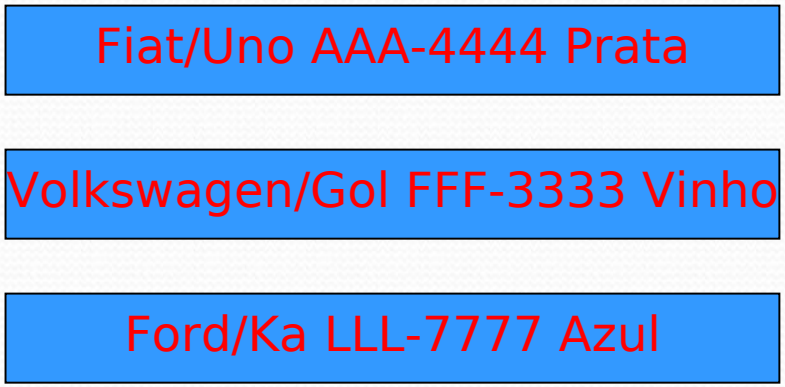
- Quando um objeto é mapeado dentro do domínio do software, os processos que podem alterar seus atributos (dados) são denominados Operações ou Métodos
- Métodos são invocados por Mensagens
- Cada objeto possui seu próprio conjunto de métodos
- Definições:
  - procedimentos definidos e declarados que atuam sobre o objeto
  - descrição de uma sequência de ações a serem executadas pelo objeto
  - especificação de COMO o objeto deve FAZER alguma coisa
  - são intrínsecos ao objeto e não podem ser separados dele

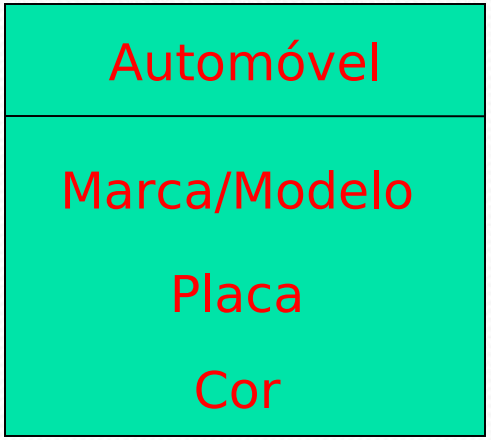
Abstração de Classificação/Instanciação



Classificação e Classes

- A categorização dos objetos em grupos e/ou classes, baseada em propriedades comuns, é um dos princípios mais importantes do paradigma orientado a objetos.





Objetos

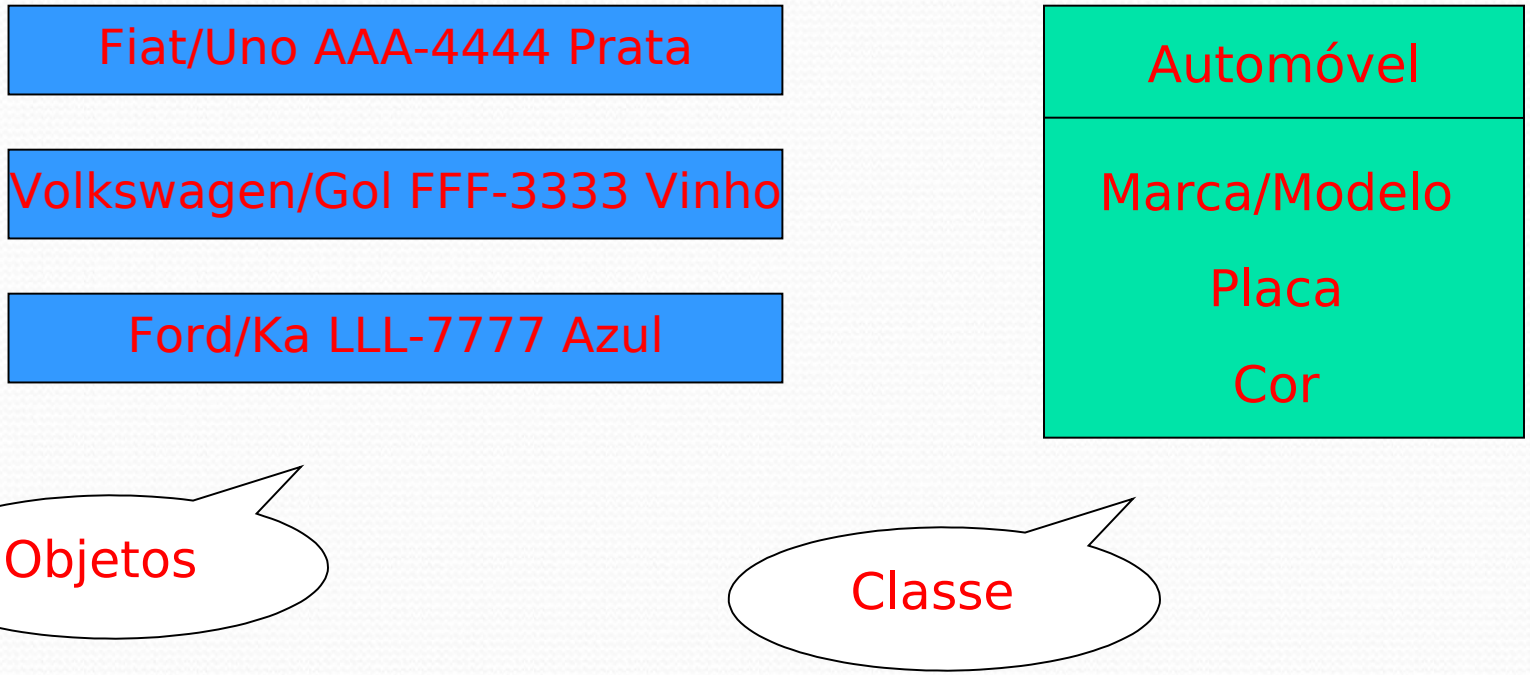
Classe

Classificação e Classes

- Classificação: categorização em Classes
  - quando modelamos um domínio, observamos várias entidades similares que podem ser abstraídas em um conceito único que reflete essa similaridade
  - operação de abstração das Similaridades
- Objeto: abstração de uma entidade do mundo real, através de atributos e operações
- Classe: abstração de um conjunto de objetos similares do mundo real

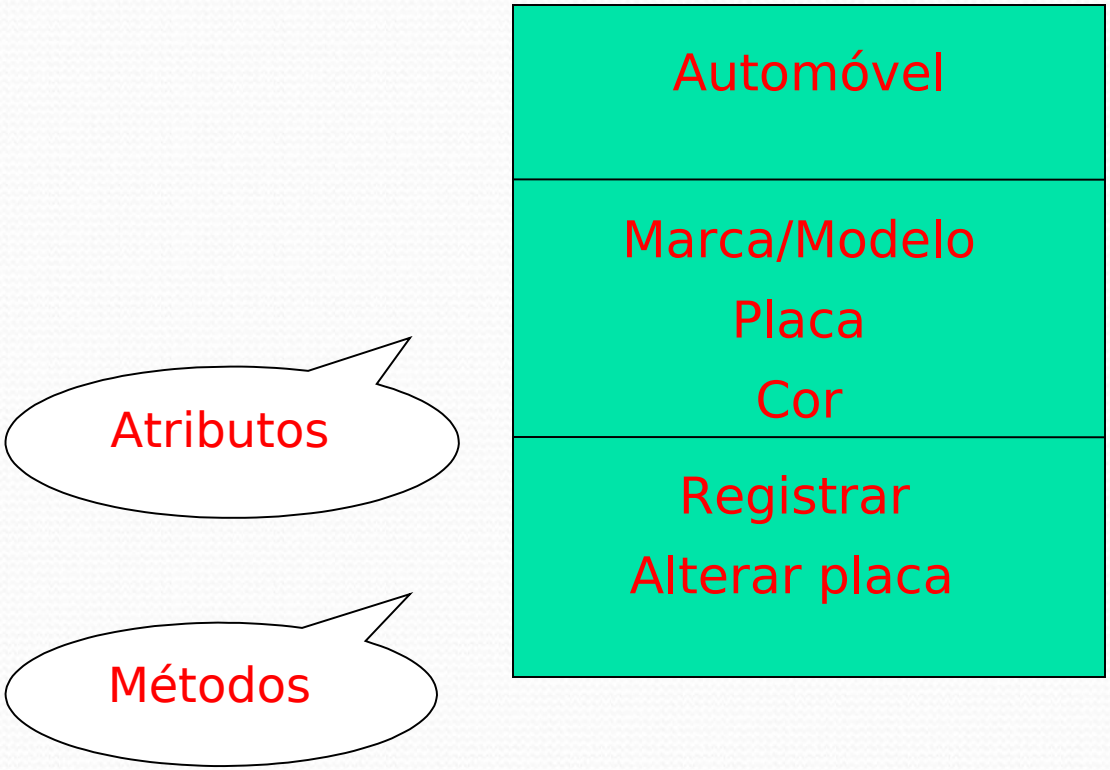
Classificação e Classes

- Cada instância de uma classe tem seus próprios valores para os atributos especificados na classe
- Todo objeto é uma Instância de uma Classe



Classificação e Classes

- Instâncias de uma mesma classe compartilham os mesmos nomes de atributos e os mesmos métodos



Interface

- Objetos se comunicam através de Mensagens, que invocam os Métodos apropriados
- Parte Privada do Objeto (Visão Interna)
  - métodos (e suas implementações)
  - atributos
- Parte Compartilhada do Objeto ou Interface (Visão Externa)
  - corresponde às operações ou métodos que podem ser invocados por outros objetos
  - agrupa um conjunto de mensagens que o objeto pode responder

Modificadores de Visibilidade

- Podem estar presentes tanto para atributos como para métodos
- Em princípio, três categorias de visibilidade podem ser definidas:
  - público: o atributo ou método de um objeto dessa classe pode ser acessado por qualquer outro objeto (visibilidade externa total)
  - privativo: o atributo ou método de um objeto dessa classe não pode ser acessado por nenhum outro objeto (nenhuma visibilidade externa)
  - protegido: o atributo ou método de um objeto dessa classe pode ser acessado apenas por objetos de classes que sejam derivadas dessa por meio do mecanismo de herança

Herança

- O que torna a computação orientada a objetos única é o conceito de herança
- Mecanismo que permite definir uma nova classe (subclasse) a partir de uma classe já existente (superclasse)

Herança

- A subclasse herda as características da superclasse:
  - os atributos e os métodos da superclasse passam a ser também atributos e métodos da subclasse
  - a subclasse pode adicionar novos atributos e métodos, e reescrever métodos herdados
- Portanto: herança é a habilidade de um objeto derivar seus atributos (dados) e métodos (funcionalidade) automaticamente de outro objeto

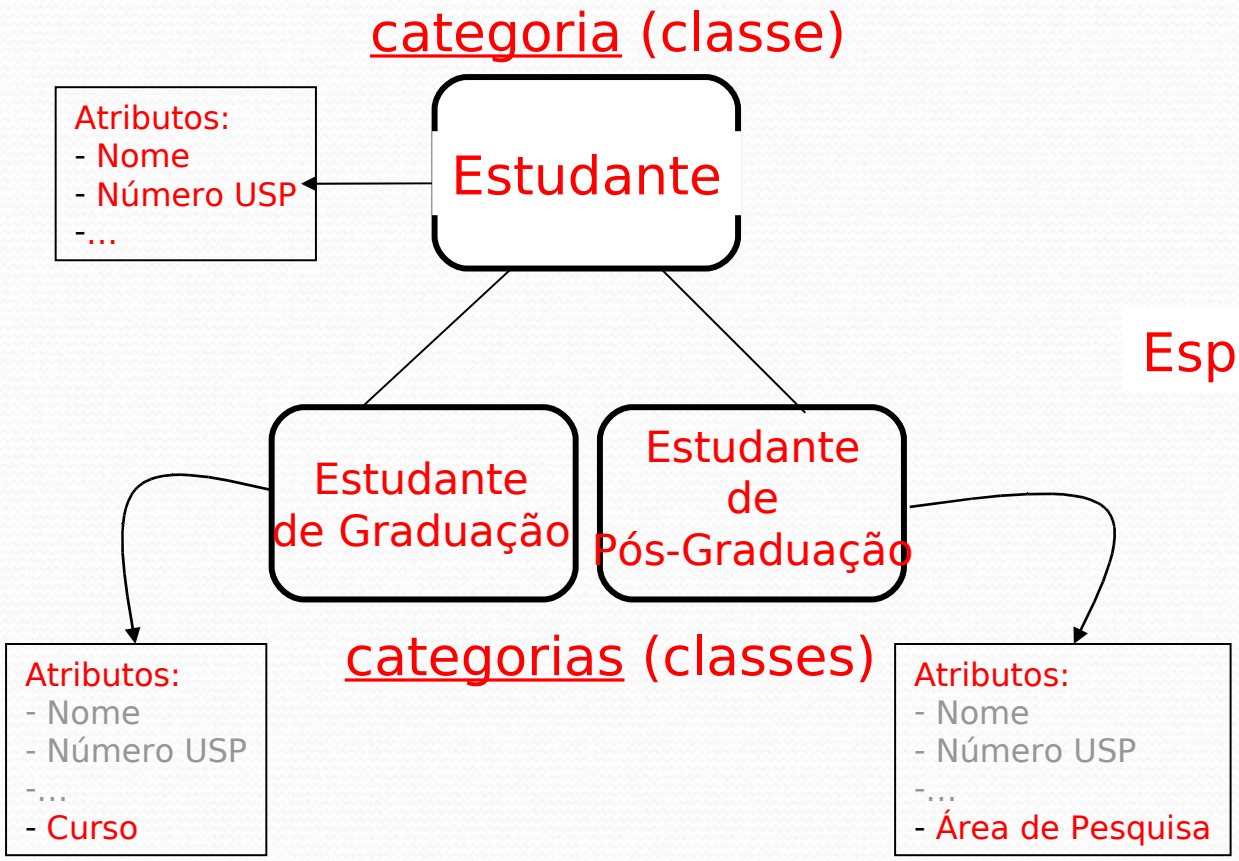
Herança

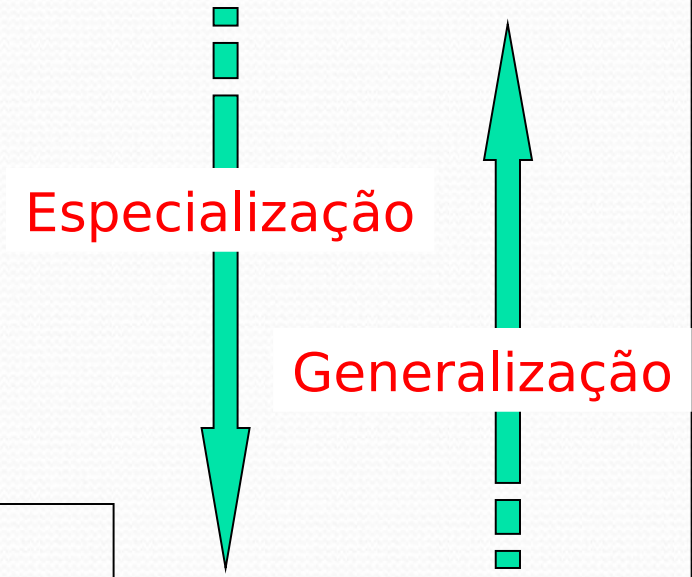
- Permite modelar uma hierarquia entre classes: classes mais especializadas (subclasses) herdam propriedades da classe mais geral (superclasse)
- Cria uma nova classe inserindo somente as diferenças desta para sua superclasse
- Identifica-se a possibilidade de herança por meio da seguinte expressão típica: “é um tipo de”

Herança

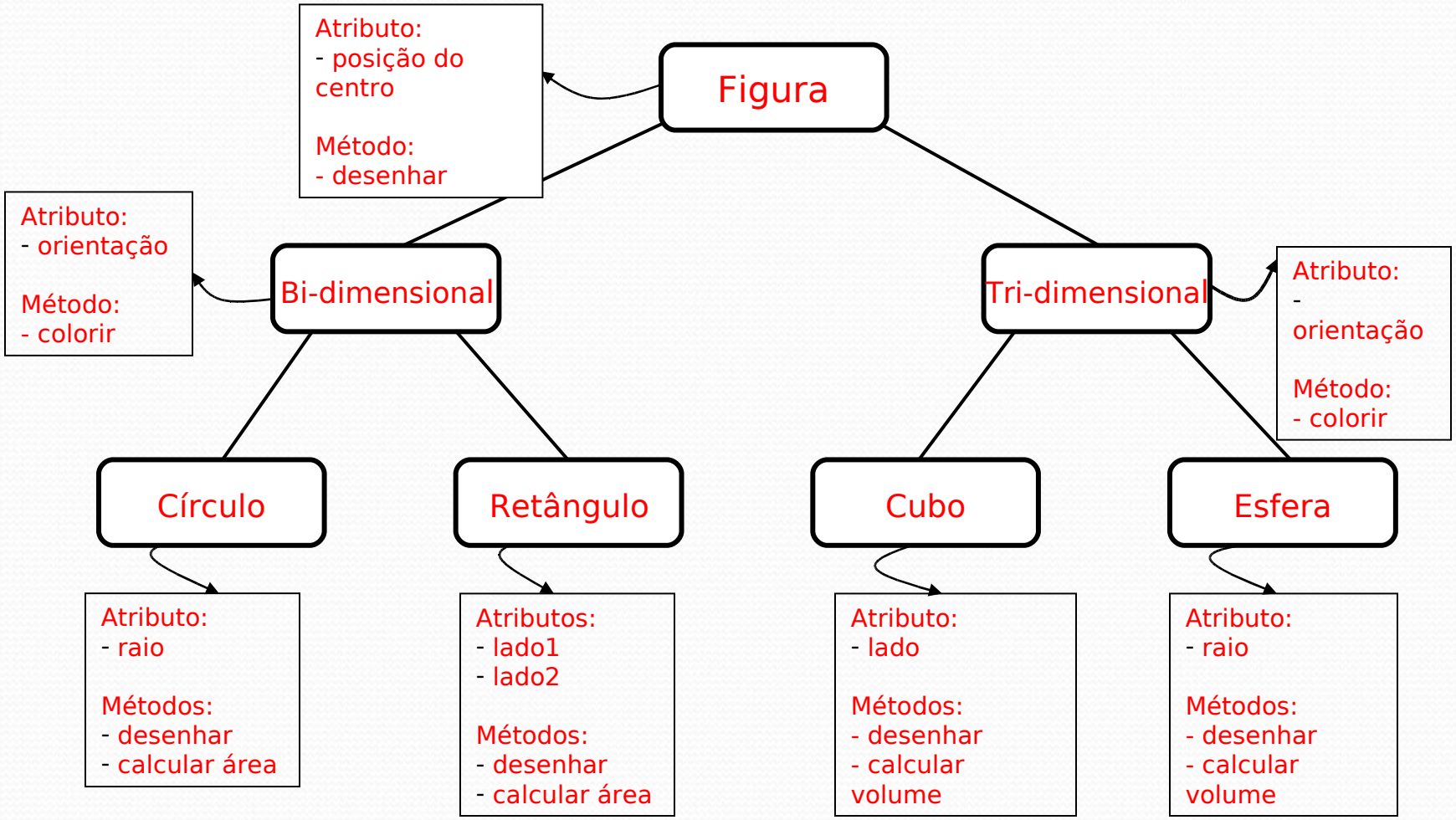
- Herança: mecanismo para derivar novas classes a partir das classes já existentes
  - Ex: Classe Carro é derivada da Classe Automóvel
- Uma classe derivada herda a representação dos atributos e operações públicas da classe base
  - adicionar novas operações
  - estender a representação dos atributos
  - sobrepor a implementação de operações já herdadas

Abstração de Generalização/ Especialização



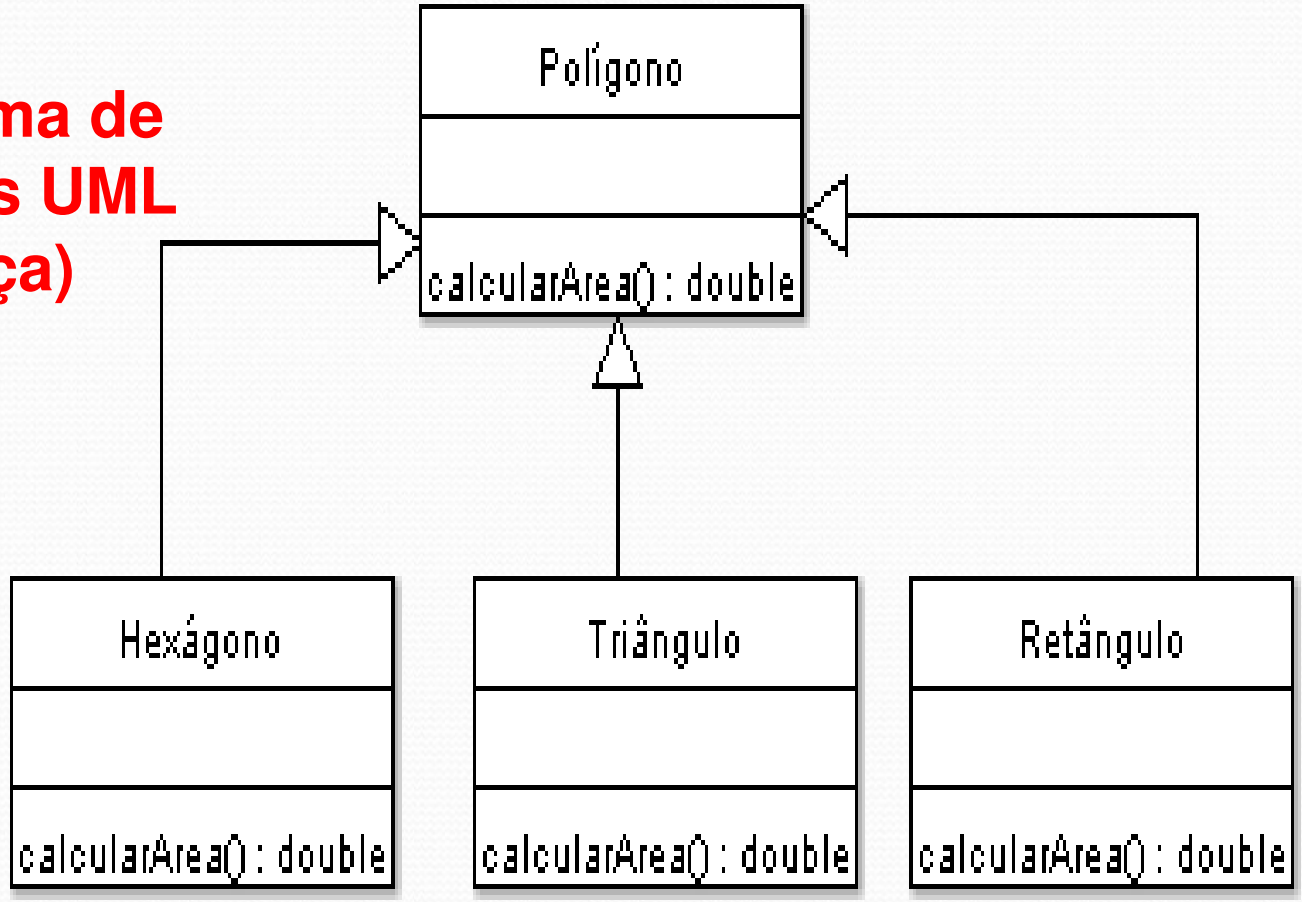


Hierarquia de Especialização



Herança

Diagrama de Classes UML (Herança)



Herança

- Construção de forma incrementada de software: (Reutilização de Software)

- Primeiro, construa classes para lidar com o caso mais geral
- Em seguida, a fim de tratar com os casos especiais, acrescente classes mais especializadas - herdadas da primeira classe

Herança

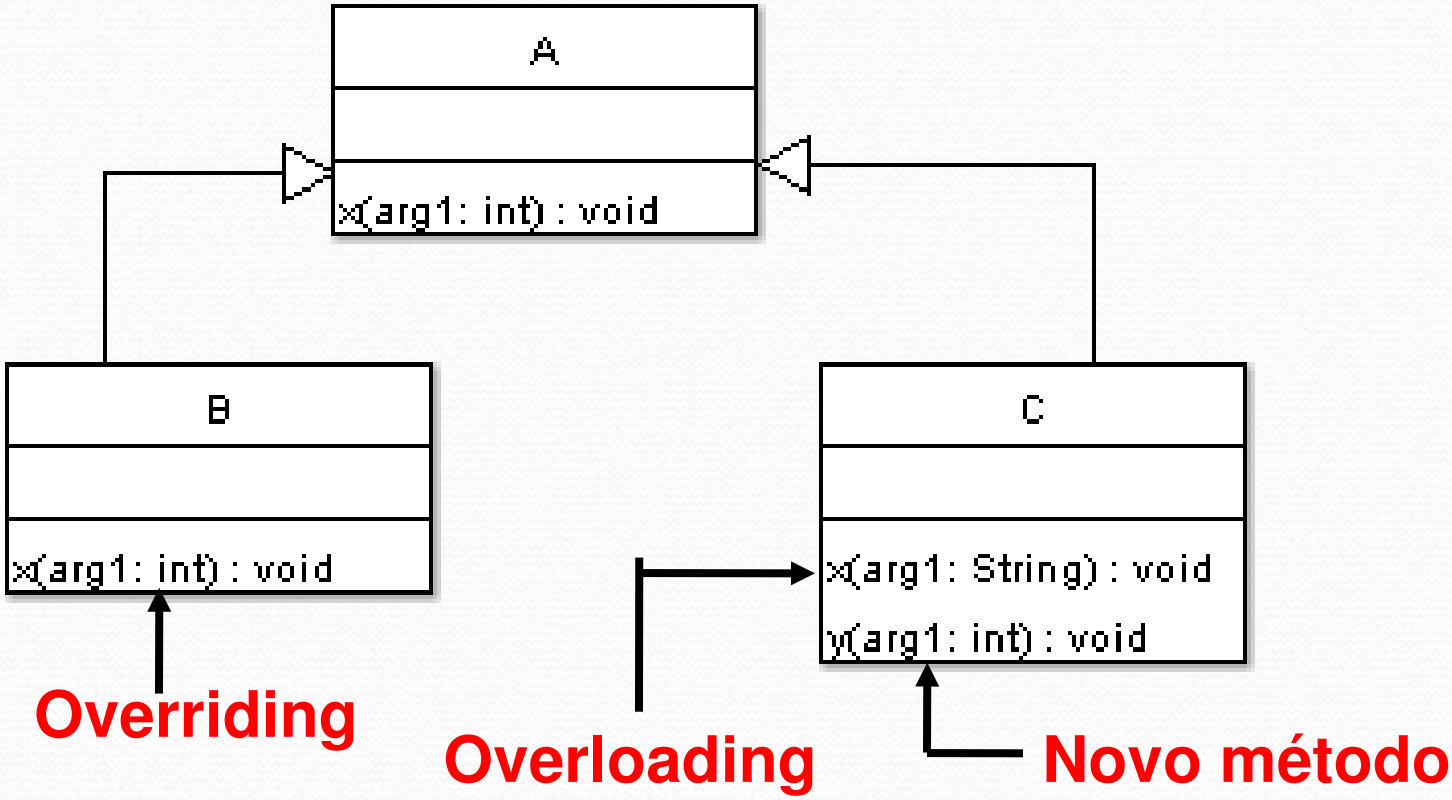
- Capacidade de Substituição
  - Deve ser capaz de substituir uma classe **Derivada** dentro de qualquer programa que exija uma classe **Base** e tudo deve funcionar bem
  - Basicamente, isso significa que, se você escrever um programa supondo que tem um **Polígono**, então pode usar livremente qualquer subtipo de **Polígono (Triângulo, Retângulo ou Hexágono)**.

Sobrecarga *(overloading)*
Redefinição *(overriding)*

- Dar um mesmo nome a mais de um método é **sobrecarga**
  - Desde que a assinatura seja **diferente**
  - **assinatura** ↔ **parâmetros + retorno**

- A redefinição de um método em classes diferentes, dentro de uma estrutura de herança é conhecida como ***overriding***
  - Necessita ter a mesma **assinatura**

Overloading & Overriding



Overloading

A a;

...

if usuário diz **OK**

then a = new B(); // Capacidade de Substituição

else a = new C(); // Capacidade de Substituição

...

a.x(10); // OK.

a.x("Aula"); // **Erro de compilação**: x(String) não está

// definida na classe A

Polimorfismo

- “O que possui várias formas”
- Propriedade de se usar o mesmo nome para métodos diferentes, implementados em diferentes níveis de uma hierarquia de classes
- Para cada classe, tem-se um comportamento específico para o método

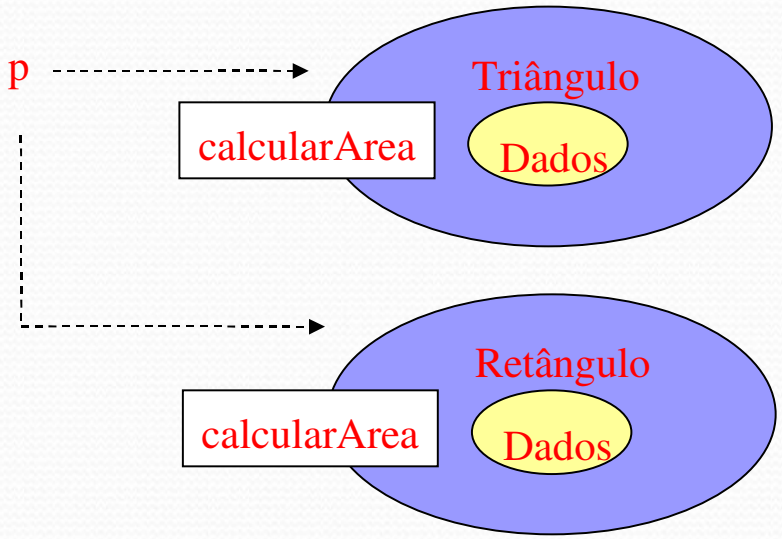
Polimorfismo / Vinculação Tardia

- Habilidade pela qual uma única operação pode ser definido em mais de uma classe e assumir implementações diferentes em cada uma dessas classes
  - *Overriding* de operações

- Vinculação tardia (late binding) é a técnica pela qual a operação a ser executada é determinada somente em tempo de execução
  - Java (implementado diretamente)
  - Palavra chave **virtual** (Linguagem C++)

Polimorfismo / Vinculação Tardia

```
Polígono p;
...
if usuário diz OK
then p = new Triângulo();
else p = new Retângulo();
...
p.calcularArea(); // aqui p pode referir-se a um objeto
                  // Triângulo ou a um objeto Retângulo
                  // Capacidade de Substituição
```



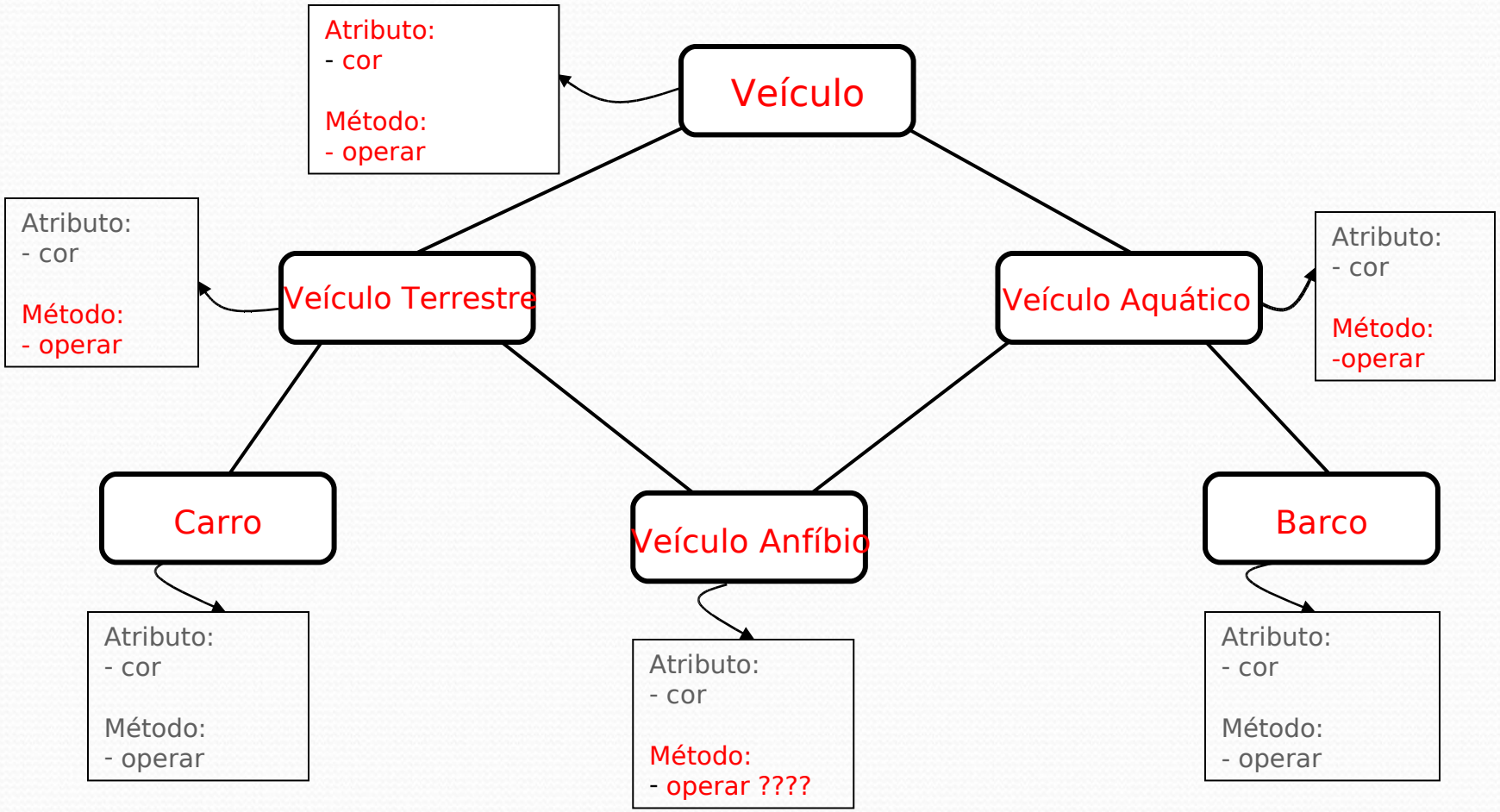
Polimorfismo

- Permite a cada objeto responder a um formato de mensagem da maneira apropriada à classe (ou subclasse) da qual foi instanciado
- Uma mesma operação pode apresentar comportamentos diferentes em classes (ou subclasses) distintas
- Uma operação pode ter diferentes implementações, isto é, mais de um método pode implementá-la

Herança Múltipla

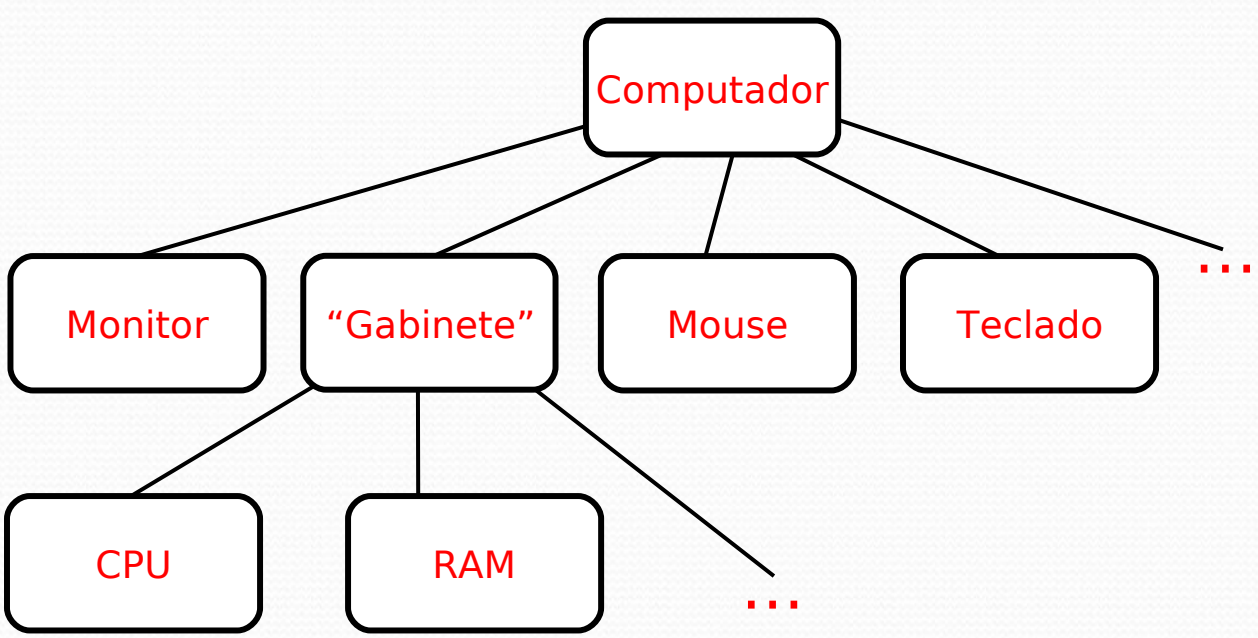
- Uma classe pode herdar características de mais de uma classe, ou seja, pode ter mais de uma superclasse
  - a subclasse herda todos os atributos e métodos de todas as suas superclasses
  - atributos/métodos de um mesmo ancestral que "alcancem" a subclasse por mais de um caminho na hierarquia são herdados apenas uma vez (são o mesmo atributo/método). Ex. a seguir: atributo cor na hierarquia de Veículo
  - conflitos em definições paralelas na hierarquia podem gerar ambiguidades (ex. a seguir: método operar na hierarquia de Veículo)

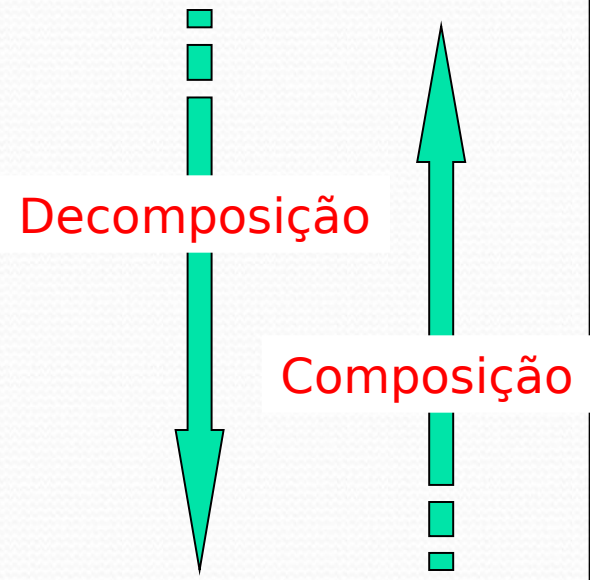
Herança Múltipla



Agregação e Composição

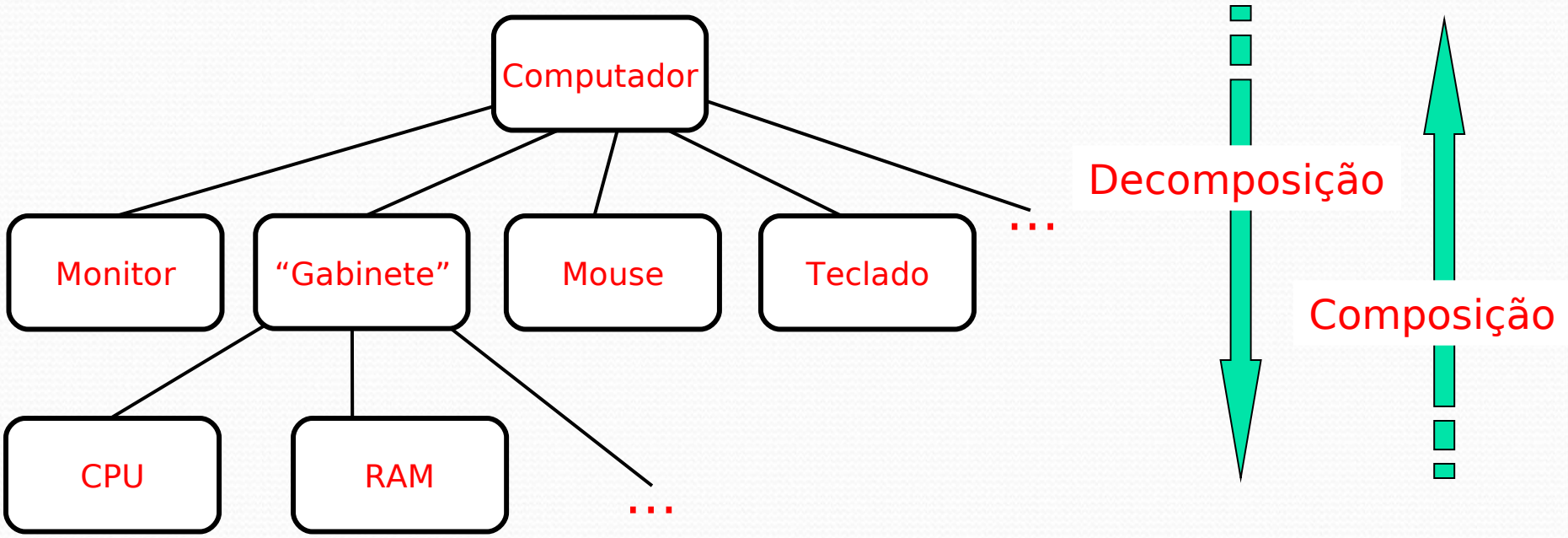
- Uma classe pode ser <u>composta</u> por outras classes, consideradas <u>partes</u> ou <u>componentes</u>





Agregação e Composição

- A composição é frequentemente referida por relacionamento “todo-parte”, no qual o agregado (“todo”) é composto de partes



Relacionamento: Agregação

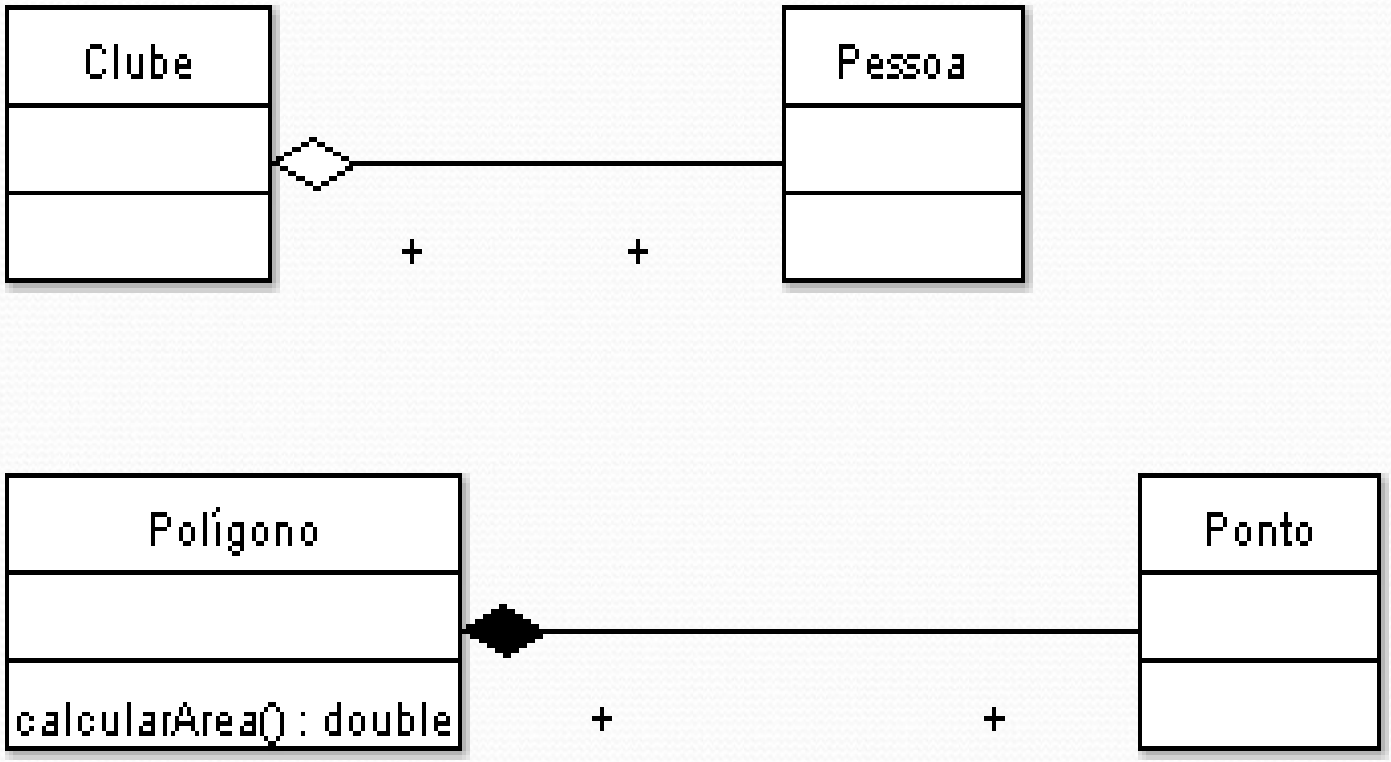
- Capturam relacionamentos do tipo “todo-parte” entre objetos
- Não existe herança entre objetos participando de uma agregação
- Agregações reduzem a complexidade ao permitir tratar vários objetos como um único

Relacionamento: Composição

- Nesse tipo de agregação, os objetos da classe “parte” não podem viver quando o objeto “todo” é destruído

- Não compartilhamento: os objetos da classe “parte” são componentes de apenas um objeto “todo”

Agregação X Composição



Diagramas de Classes UML (Agregação e Composição)

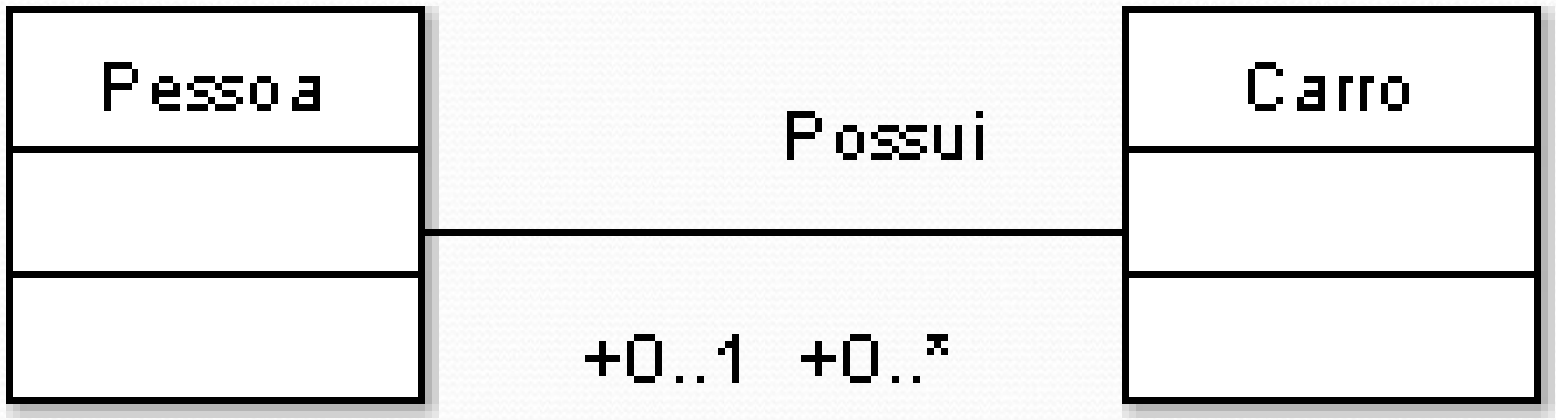
Herança e Agregação/Composição

- Herança é uma relação “é um”
  - Carro **é um** Veículo

- Agregação/Composição é uma relação “é parte de um”
  - Pneu **é parte de um** Carro
  - Motor **é parte de um** Carro

Associação

- Uma associação nos permite capturar relacionamentos básicos que existem entre conjuntos de objetos



Diagramas de Classes UML (Associação)