Introdução à Programação Orientada a Objetos

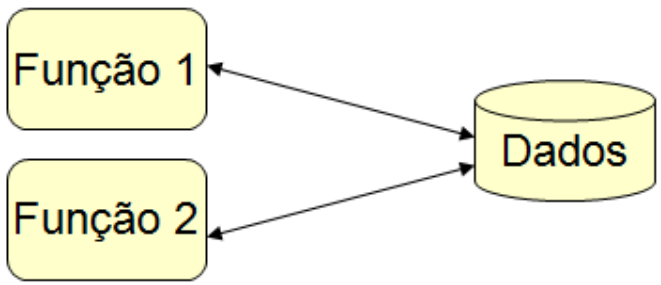
BCC 221 - Programação Orientada a Objectos(POO)

Guillermo Cámara-Chávez

Departamento de Computação - UFOP

Paradigma Imperativo I

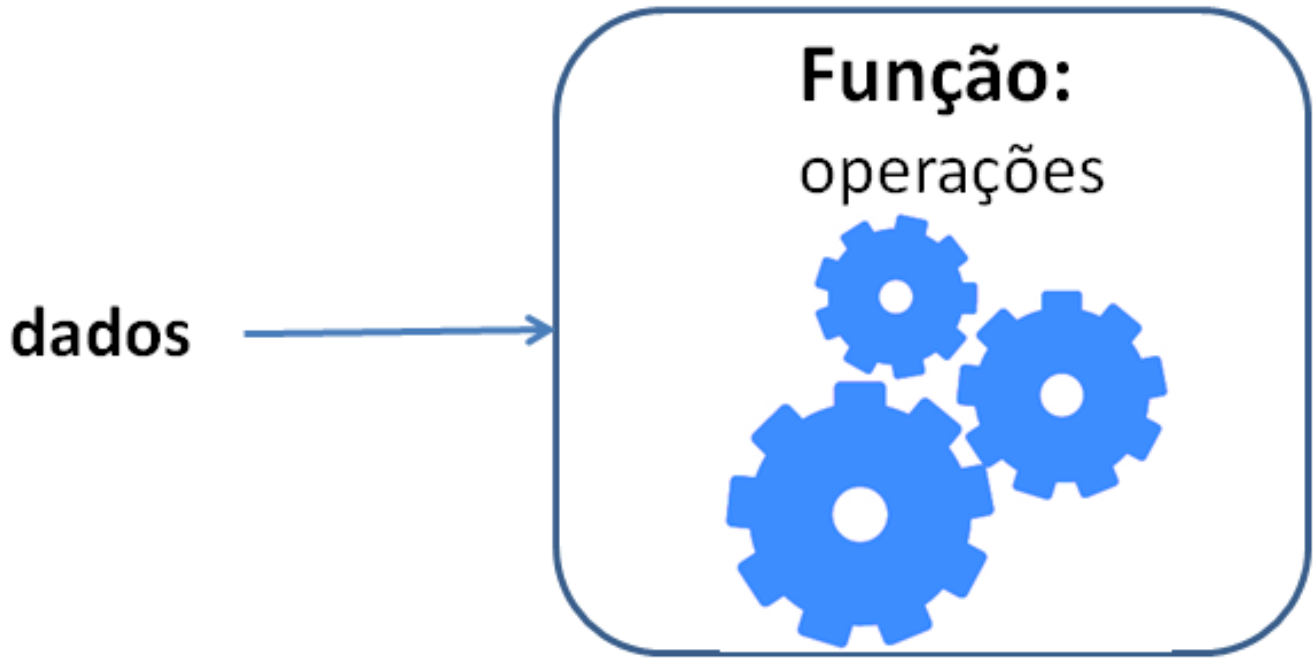
- Linguagens Imperativas:
  - Fazem clara distinção entre funções e dados.
  - Funções, a princípio, são ativas e têm comportamento.
  - Dados são repositórios passivos de informação, afetados por funções.



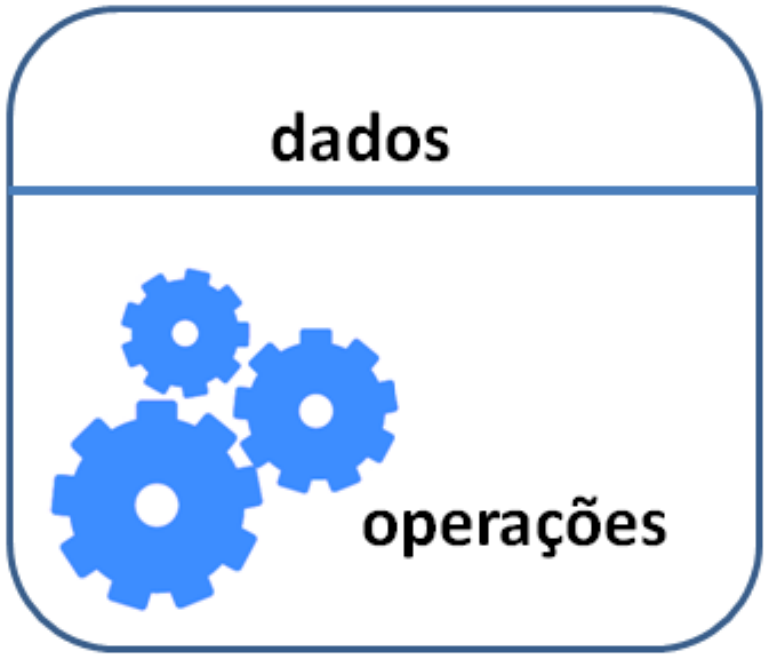
O que é um programa? I

- **Programas processam dados**: valores em uma conta bancária, caracteres entrados por um teclado, pontos de uma imagem, etc;
- O paradigma de programação orientado a objetos considera que os **dados** a serem processados e os **mecanismos de processamento** destes dados devem ser **considerados em conjunto**;
- A **criação de modelos** que representam conjuntamente dados e operações nestes dados é a solução.

O que é um programa? II



O que é um programa? III

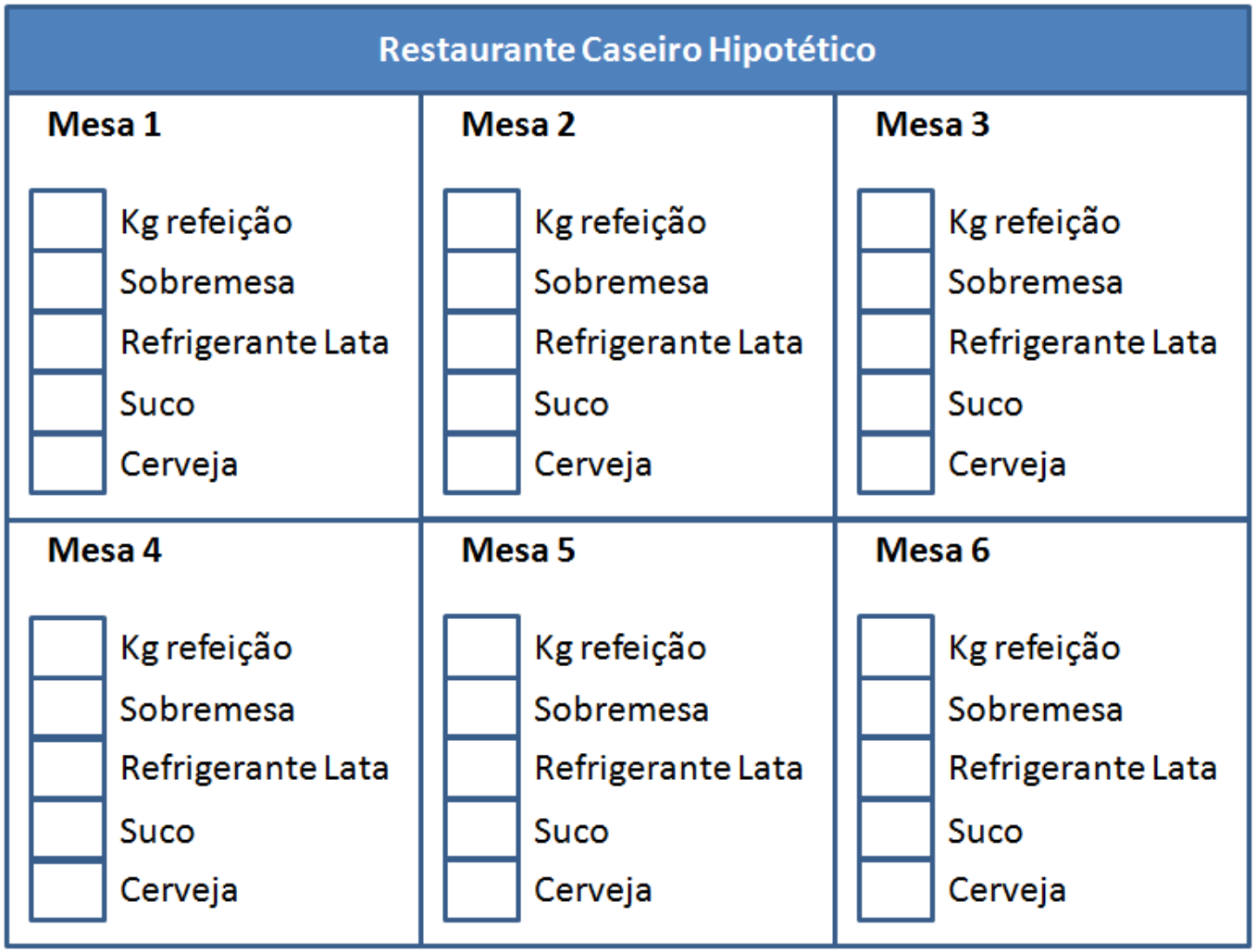


Modelos I

- **Modelos são representações simplificadas de** objetos, pessoas, itens, tarefas, processos, etc. usados comumente por pessoas no seu dia-a-dia independente do uso de computadores.
- Para simplificar o uso de modelos em tarefas comuns, consideramos o **Restaurante Caseiro Hipotético**, que serve refeições por quilo, e onde o gerente que fica na balança e no caixa:
  - anota os pesos dos pratos dos clientes e
  - anota os pedidos que os garçons trazem em uma comanda.

Modelos II

- A figura abaixo mostra a comanda do restaurante caseiro hipotético



Modelos III

- O modelo do restaurante representa **certos dados ou informações**, que no caso são os itens e a quantidade dos pedidos;
- Um modelo comumente contém **operações ou procedimentos associados** a ele, por exemplo:
  - Inclusão de um pedido para uma mesa;
  - Encerramento de um pedido de uma mesa e
  - Apresentação da conta para os clientes.

Modelos IV

- **Modelos podem conter submodelos** e ser parte de outro modelos.

- A **simplificação** inerente aos modelos é, em muitos casos, **necessária**: dependendo do contexto, algumas informações devem ser ocultadas ou ignoradas;

Modelos V

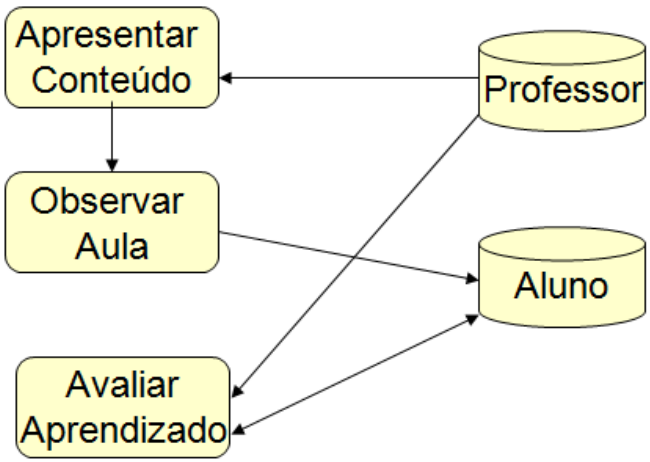
- Em um modelo, a **representação das informações** sobre uma pessoa **pode ser feita de maneira diferente** dependendo do contexto.

contexto	**dados**	**operações**
empregada empresa	nome, cargo, salário, horasExtras	calculaSalario, aumentaSalario
paciente clínica	nome, sexo, idade, altura, peso, historicoConsultas	VerificaObesidade
contato comercial	nome, telefone, cargo, empresa	mostrarTelefone

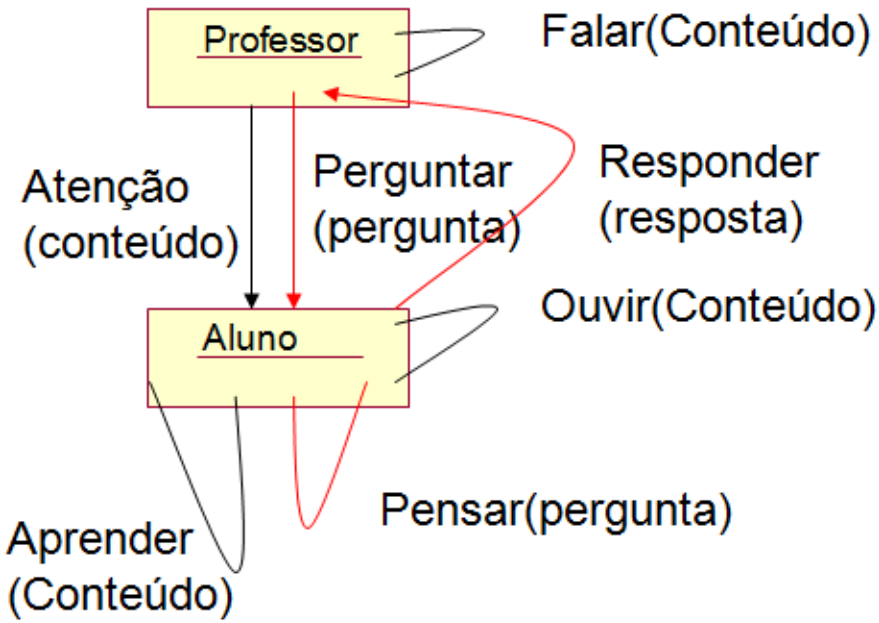
POO vs. Paradigma Imperativo I

Processo de Ensinar

Linguagens Imperativas



POO



O que é POO? I

- Programação Orientada a Objetos é uma metodologia de desenvolvimento de software.
- É um paradigma de programação onde se usam **classes e objetos criados** a partir dos **modelos** descritos anteriormente, para **representar e processar dados**.
- Sua principal vantagem é a proximidade com a forma que os seres humanos visualizam e entendem o mundo ao seu redor.

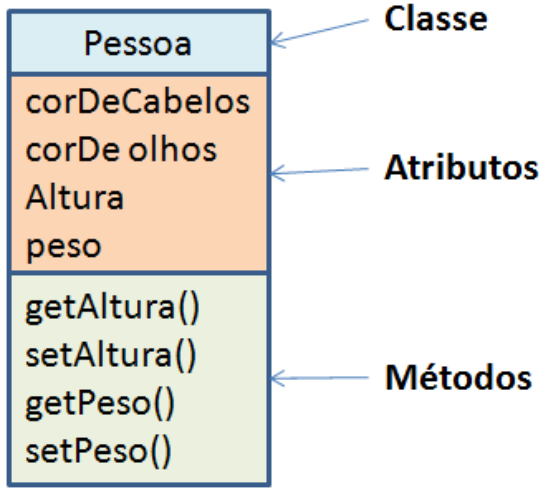
O que é POO? II

- Tem como principal objetivo **facilitar a modelagem** e desenvolvimento de sistemas, através da **interação entre objetos**.
- Os modelos na POO são chamados de classes
- **Classes** são **estruturas** que **contém um determinado modelo**.
  - os **dados** devem ser representados e
  - as **operações** devem ser efetuadas entre esses dados

O que é POO? III

- As **classes** especificam as **características** dos **objetos** materializados a partir delas.
- Tais características são chamados de **atributos**.
- Ao ser materializado um objeto dizemos que criamos uma **Instância** da classe.
- Instâncias distintas de uma mesma classe possuem os mesmos atributos, entretanto com possíveis diferenças de valores.

O que é POO? IV



- Os objetos são diferenciadas pelos seus nomes de referência e não pelo valor de seus atributos.

O que é POO? V

- Em muitas situações os problemas necessitam ser fragmentados de acordo com as **ações** que esse modelo especificará para a representação do problema.

- Objetos possuem comportamentos os quais são representados nas classes através dos **métodos**.

O que é POO? VI

- A **fragmentação** permite que o problema seja **representado** por um **conjunto de métodos**, os quais representam as **ações** do modelo.
- Vantagens de ter vários métodos em um modelo:
  - facilidade em escrever o modelo;
  - utilizações repetidas de um mesmo método;
  - facilidade em dar manutenção;
  - utilização adequada de recursos.

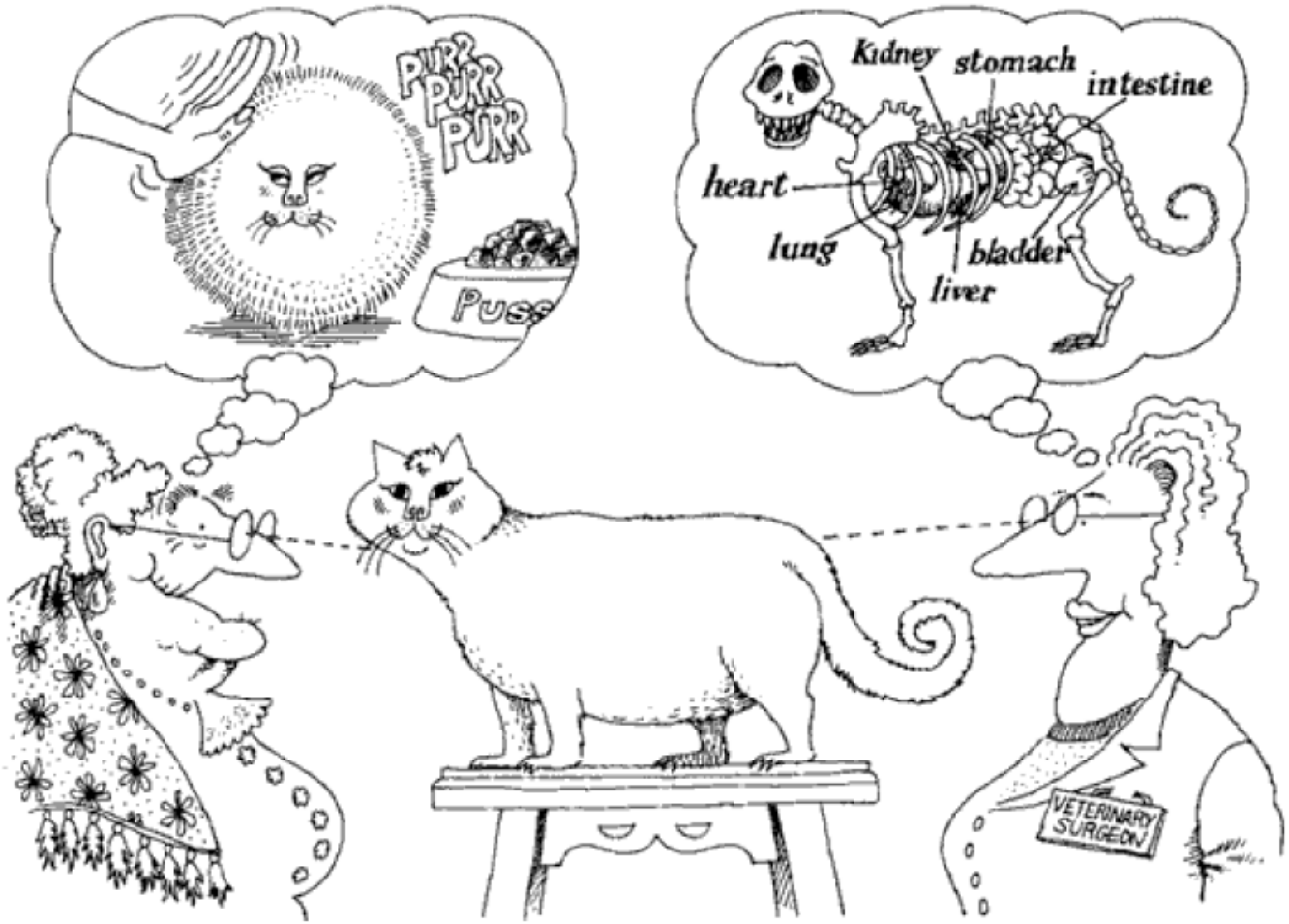
Fundamentos do Modelo de Objetos I

- Encapsulamento
- Modularidade
- Abstração de dados
- Hierarquia

Abstração de conceitos do Domínio I

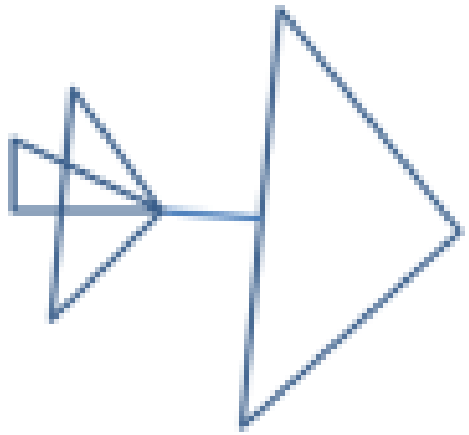
- As pessoas tipicamente tentam **compreender o mundo, construindo modelos mentais** de partes dele.
- Um modelo mental **abstrai** apenas as **características relevantes** de um sistema para seu entendimento.

Abstração de conceitos do Domínio II



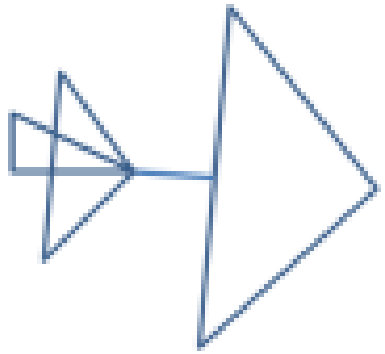
Abstração de conceitos do Domínio III

Concreto: todos os detalhes da realidade



Abstrato: só características importantes

Abstração de conceitos do Domínio IV



Quanto mais **abstrato** mais geral.
A figura ao lado poderia representar qualquer avião.

Quanto mais **concreto**, mais específico
O avião ao lado é tão específico, com tantos detalhes, que só representa ele mesmo

Abstração de conceitos do Domínio V

- As pessoas tipicamente tentam compreender o mundo, construindo modelos mentais de partes dele.
- Um modelo mental abstrai apenas as características relevantes de um sistema para seu entendimento.
- Um Engenheiro Civil quando pensa em um elevador, visualiza ele de um modo diferente que um Analista de Sistema.

Abstração de conceitos do Domínio VI

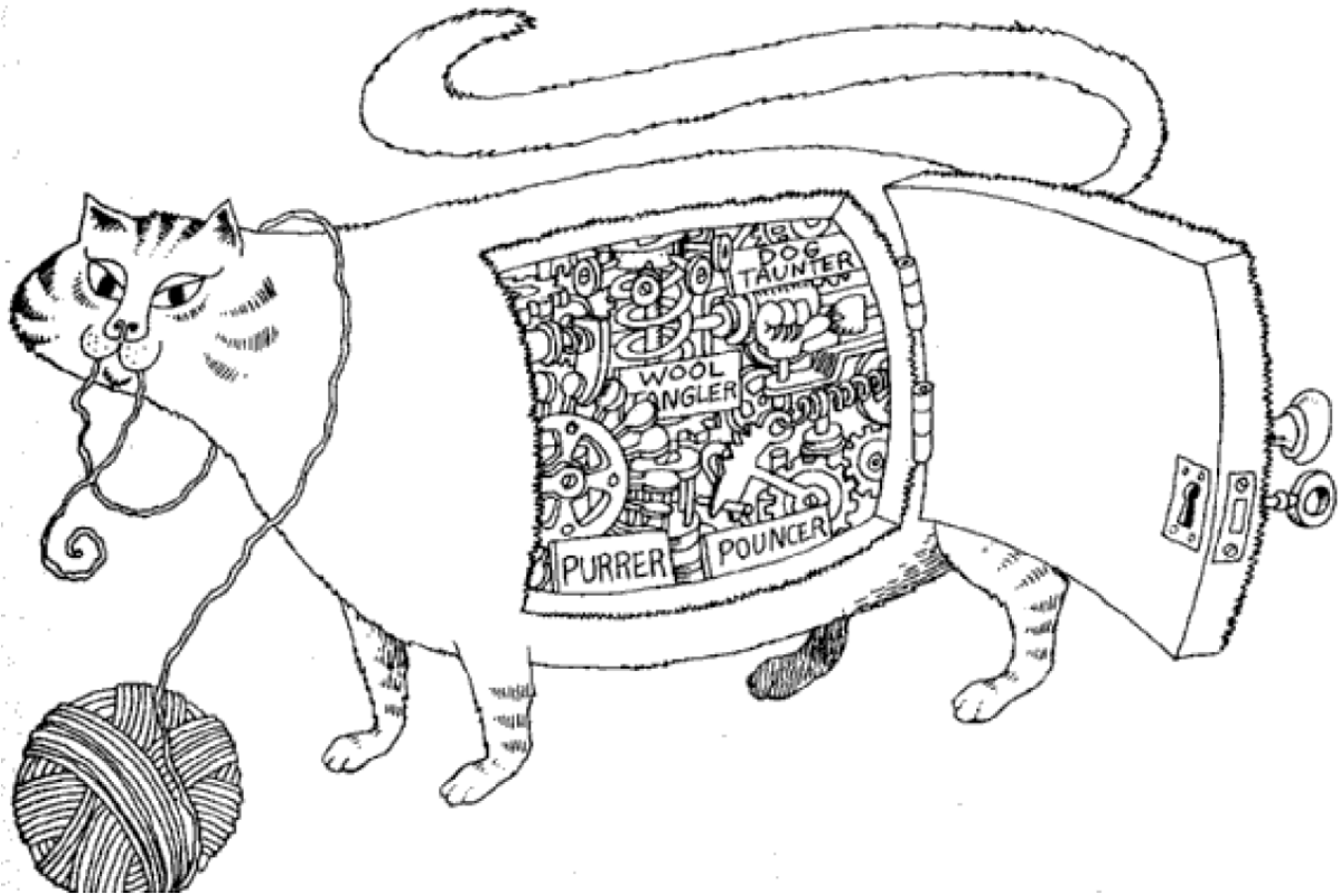
- **Engenheiro**: Qual a Marca? Qual a capacidade? Qual a Velocidade? Indicado para que tipo de edifício? Qual a largura? Qual a Altura?

- **Analista**: Quais as características que devem ser consideradas: estado(subindo, descendo, parado), porta aberta ou fechado, qual andar que o elevador esta, indicar o andar atual no painel. Abrir porta, Fechar porta, subir, descer, parar, atualizar indicador de andar.

Encapsulamento I

- É um dos conceitos básicos de POO
  - A idéia de uma "caixa preta":
  - Não é necessário saber os detalhes de funcionamento interno, mas sim como utilizar.
- Encapsulamento é um dos benefícios mais palpáveis da POO.
- Modelos que encapsulam os dados possibilitam a criação de programas com menos erros e mais clareza.

Encapsulamento II



Encapsulamento III

- Encapsular é esconder como as coisas funcionam por trás de uma interface externa

  Ex: caixa automático

  - Como ele é implementado internamente?
  - Utilizamos através de operações bem conhecidas

- Em muitos casos, será desejável que os dados não possam ser acessados ou usados diretamente, mas somente através das operações.

Modularidade I

Modularidade II

- Um sistema com um módulo que permite ordenar dados
- O múdulo inicialmente foi implementado usando o algoritmo *bubblesort*
- Re-implementação do módulo usando o *quicksort*

Modularidade III

- Propiedade de construção de sistemas através de **módulos**
- A **coesão** de um módulo denota a média da inter-relação entre os seus componentes. Uma classe não deve assumir responsabilidades que não são suas.
- O **acoplamento** entre os módulos é o grau de interdependência entre eles

Ideal: Alta coesão e Baixo acoplamento

Modularidade I

- Modularização é o processo de dividir um todo em partes bem definidas, que podem ser construídas e examinadas separadamente
- Essas partes interagem entre si, fazendo com que o sistema funcione de forma adequada
- Particionar um programa em componentes individuais, pode reduzir a complexidade

Modularidade II

- Critérios para modularidade

  Decomposição: é alcançado quando o modelo de design ajuda a decompor os problemas em outros subproblemas cujas soluções podem ser atingidas separadamente. Este tipo de design é chamada de *top-down*

  Composição: Achar maneiras de desenvolver pedaços de programas que executem tarefas bem definidas e utilizáveis em outros contextos

Modularidade III

Entendimento: ajuda à produção de módulos que podem ser separadamente compreendidos pelos desenvolvedores; no pior caso, o leitor deve ter atenção sobre poucos módulos vizinhos.

Continuidade: um método de desing satisfaz a continuidade se uma pequena mudança na especificação do problema resulta em alterações de um único ou poucos módulos

Proteção: provê a arquitetura de isolamento quando da ocorrência de condições anômalas em tempo de execução, seus efeitos ficam restritos aquele modulo ou pelo menos se propaga a poucos vizinhos.

Hierarquia I

- Quando um sistema tem muitos detalhes, ele pode ser decomposto em uma hierarquia de abstrações
- Permite que detalhes relevantes sejam introduzidos de uma maneira controlada
- É um "ranking" ou uma ordenação de abstrações
- Duas categorias de hierarquias
  - Hierarquia de agregação/decomposição
  - Hierarquia de generalização/especialização

Hierarquia II

Generalizar: remover restrições para obter abstrações mais genéricas

Especializar: buscar características que diferenciem abstrações afins

Agregar: combinar abstrações para obter estruturas e comportamentos mais complexos

Decompor: significa detalhar uma abstração dividindo-a nos seus elementos mais constituintes

Hierarquia III

Conceitos Básicos: Objetos I

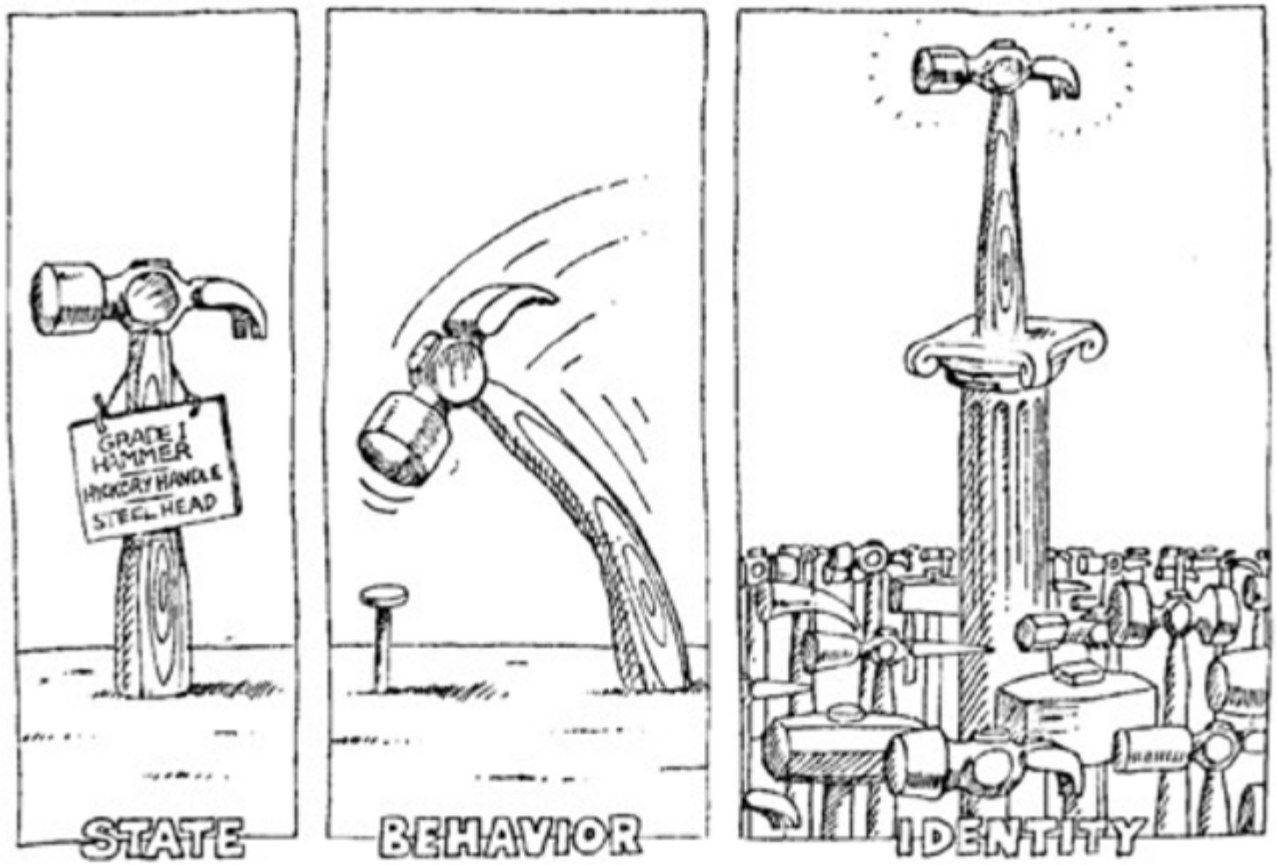
- Encapsula dados (atributos) e funções (membros), ou seja, é uma **abstração de dados e uma abstração de operações** em uma mesma estrutura
- Possui uma interface de acesso bem definida (atributos e métodos internos são invisíveis ao meio externo)
- Um objetos possui estado, comportamento e identidade

Conceitos Básicos: Objetos II

- Estado de um objeto: conjunto de valores associados às propriedades do objeto. Representam:
  - características,
  - atributos, ou
  - qualidades associadas à entindade que representam

- O comportamente: descreve como se **processam as mudanças** do estado em função de sua **interação** com o mundo externo

Conceitos Básicos: Objetos III



An object has state, exhibits some well-defined behavior, and has a unique identity.

Conceitos Básicos: Objetos IV

- Exemplo
  - Um empregado de uma firma pode ser representando por um objeto
  - Todo empregado tem: nome, carteira de identidade, endereço, seção na qual trabalha, salario, etc
  - Pode-se: alterar o salário de um empregado, imprimir seu endereço, etc

Conceitos Básicos: Objetos V

- Exemplo
  - Uma lista pode ser representada como um objeto
  - Todo lista tem: um número de elementos, etc
  - Pode-se: inseriri ou remover elementos, imprimir elementos, etc

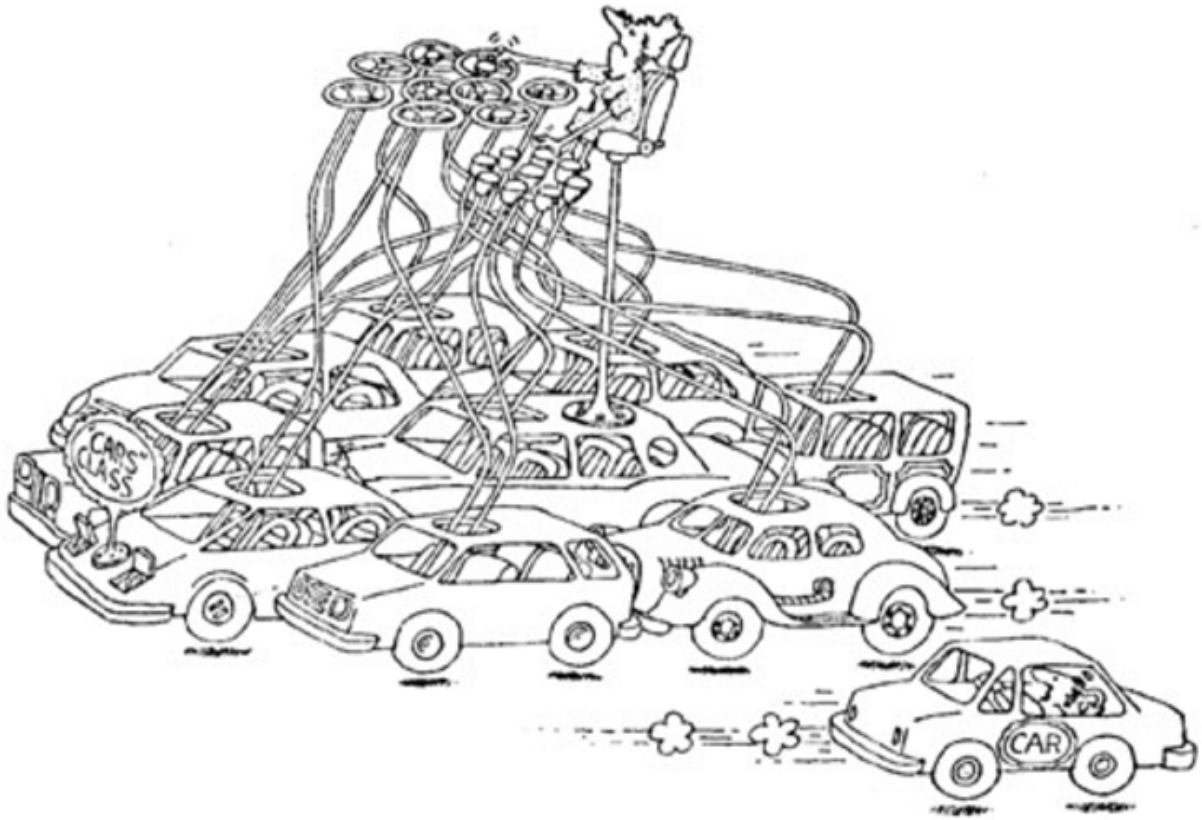
Conceitos Básicos: Classes I

"Classe é um agrupamento de objetos que revelam profundas semelhanças entre si, tanto no aspecto estrutural quanto no funcional".

Exemplo:
Tom, Garfield, Frajola, Manda-chuva e Batatinha. São todos da mesma classe: **Gatos**

Conceitos Básicos: Classes II

Concepts



A class represents a set of objects that share a common structure and a common behavior.

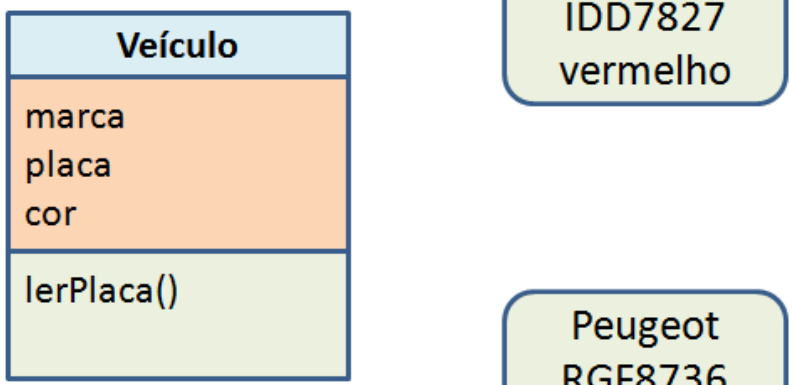
Conceitos Básicos: Classes III

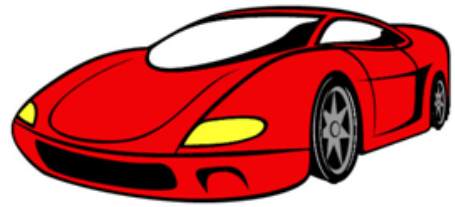
- Uma classe é um tipo:
  - definido pelo usuário
  - contém o molde, a especificação para os objetos (como o tipo inteiro contém o molde para as variáveis declaradas como inteiros)

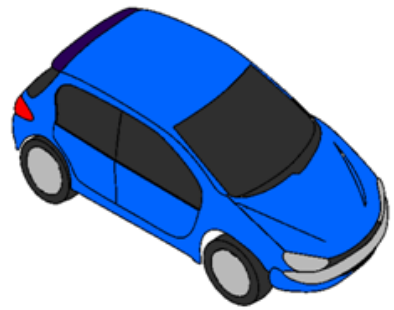
Conceitos Básicos: Classes IV

- A classe envolve, associa, métodos e dados, controlando o acesso a estes
- Defini-la implica em especificar os seus ATRIBUTOS (dados) e seus SERVIÇOS (código)
- Instância: a partir de uma classe pode-se gerar diversas instâncias (objetos)

Conceitos Básicos: Classes V







Conceitos Básicos: Classes VI

- Classes Abstractas
  - Classes que não utilizaremos para instanciar
  - Existem apenas para servir de molde para outras classes
    - Para que outras classes herdem a interface e/ou implementação

Conceitos Básicos: Classes VII

- Classes Concretas
  - Podem ser instanciadas
    - Ou seja, podem criar objetos

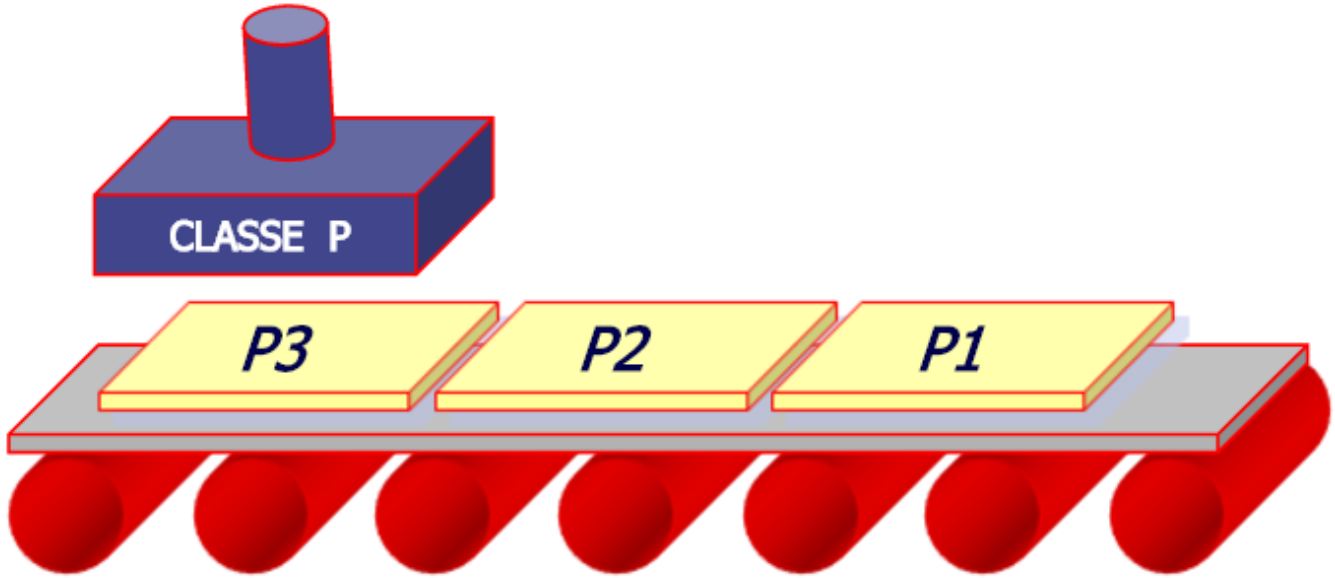
Exemplos de modelos I

- Um objeto Homem, pode ser "moldado" pela classe **pessoa**
- Uma cadeira pela classe **moveis**
- Um fusca pela classe **carro**, etc.

- Desta forma uma classe pode gerar vários objetos, tendo todos eles as mesmas características

Exemplos de modelos II

- Instancias

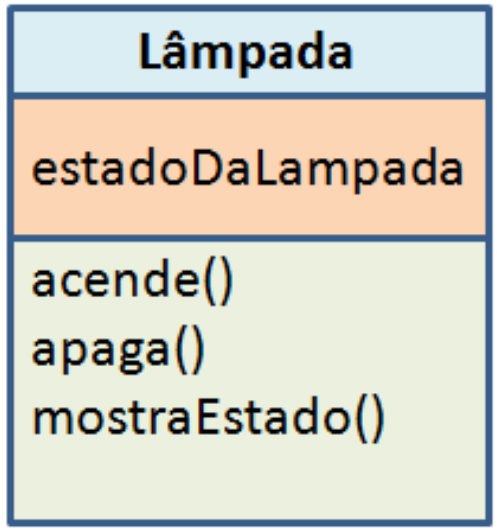


onde:

- P: Classe do objeto P
- $P1$, P_2, $P-3$: Instâncias da classe P (cada classe terá seus proprios dados independentes dos outros.

Exemplos de modelos III

- Exemplo 1: Uma lâmpada incandescente



- Atenção para os nomes de dados e das operações.
- A abrangência do modelo define os dados e operações.

Exemplos de modelos IV

```
Modelo Lampada
Início do modelo
   dado estadoDaLampada;
   operação acende()
   início
         estadoDaLampada = aceso;
   fim
   operação apaga()
   início
         estadoDaLampada = apagado;
   fim
   operação mostraEstado()
   início
         se (estadoDalampada == aceso)
                  imprime "A lâmpada está acesa"
         senão
                  imprime "A lâmpada está apagada";
   fim
Fim do modelo
```

Representa uma lâmpada em uso

Acende a Lâmpada

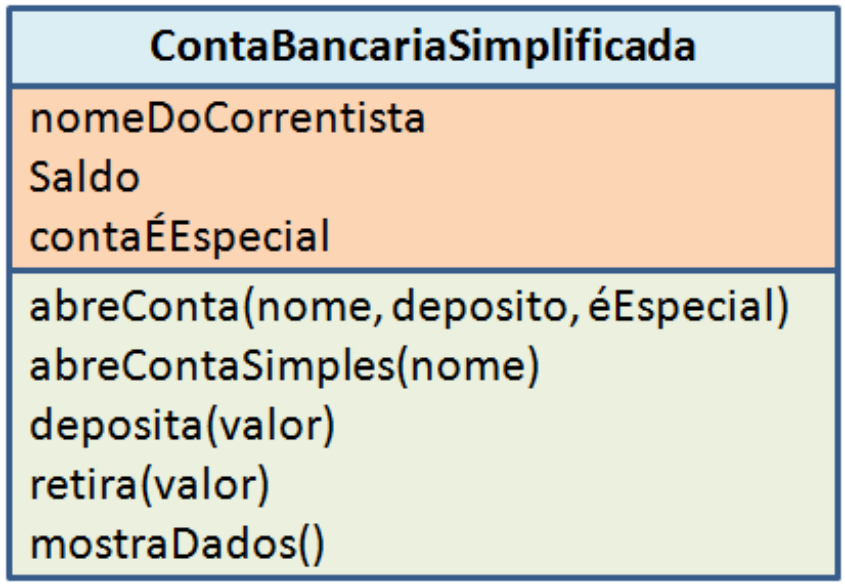
Apaga a Lâmpada

Mostra o estado da lâmpada

13

Exemplos de modelos V

- Exemplo 2: Uma conta bancária simplificada



- Aspectos práticos de contas reais (senhas, taxas) foram deixados de lado.

Exemplos de modelos VI

Modelo ContaBancáriaSimplificada
Inicio do modelo
dado nomeDoCorrentista, saldo, contaÉEspecial ;

Dados da conta

operação abreConta(nome,deposito,especial)

argumentos para a operação

início
nomeDoCorrentista = nome;
saldo = deposito;
contaÉEspecial = especial;
fim

Inicializa simultaneamente todos os dados do modelo

operação abreContaSimples(nome)
início
nomeDoCorrentista = nome;
saldo = 0.00;
contaÉEspecial = false;
fim

Inicializa simultaneamente todos os dados do modelo, usando o nome passado como argumento e os outros valores com valores *default*

Exemplos de modelos VII

```
operação deposita(valor)
início
        saldo = saldo + valor;
fim
operação retira(valor)
início
        se (contaÉEspecial == false)
        inicio
                se (valor <= saldo)
                        saldo = saldo – valor;
        fim
        senão
                saldo = saldo - valor;
fim
operação mostraDados()
inicio
        imprime "o nome do correntista é " + nomeDoCorrentista;
        imprime "o saldo é " + saldo;
        imprime se (contaEEspecial) imprime "A conta é especial"
fim
Fim do modelo
```

Deposita um valor na conta

Retira um valor na conta

A conta não é especial

Existe saldo suficiente

Conta especial, pode retirar a vontade

Mostra os dados imprimindo seus valores

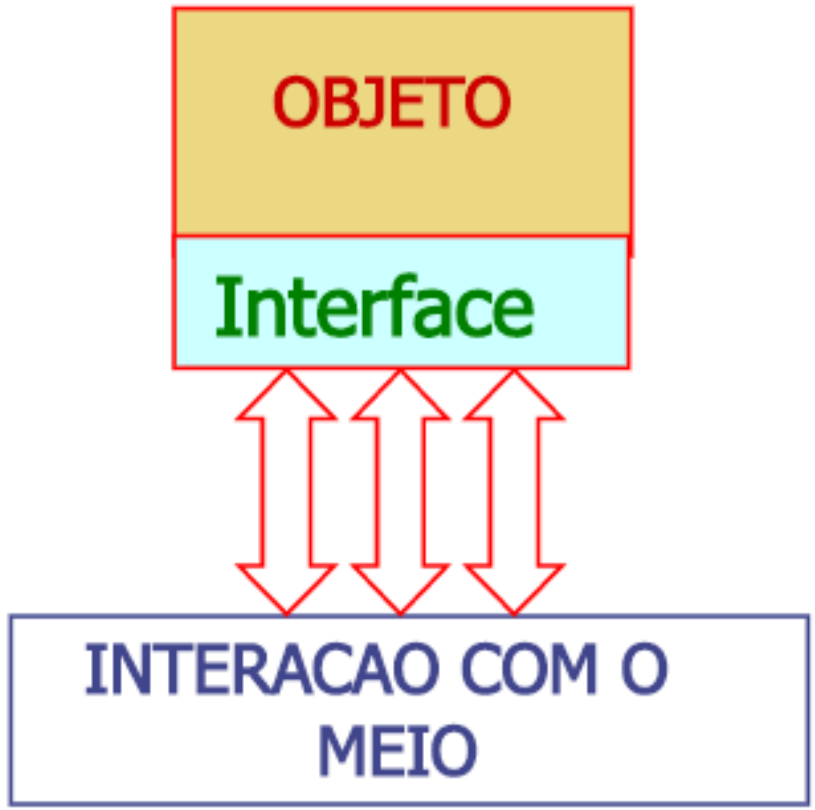
2

Interfaces I

- Através dos métodos, interagimos com o objeto solicitando e/ou alterando seu estado interno
- Esta é a única forma de interagirmos com o interior do objeto
- Dá a ideia de uma "caixa preta",
  - a forma como os dados são trabalhados internamente **não tem muito significado** para o usuário,
  - já o resultado da ação é importante

Interfaces II

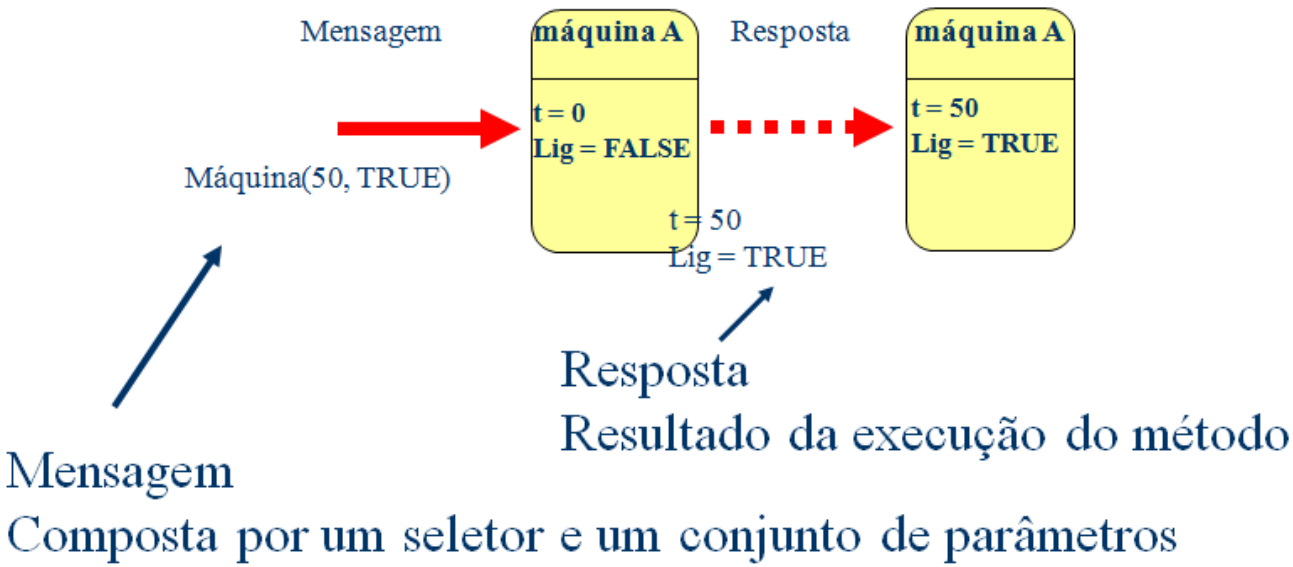
- A **interação** de um objeto com o mundo externo é realizado através de um conjunto de métodos que representa a **interface** do objeto



Interfaces III

Mensagens

- Os objetos são solicitados a executar algo através de **mensagens**
- **Mensagem:** Ativa um método de um objeto



Resumo I

Objeto: é uma entidade fechada que possui dados (características) e métodos (operações ou ações)

Método: é uma sub-rotina interno no objetos que informa como executar uma ação

Classe: é um recipiente para construir objetos. Aqui são definidos os atributos e métodos.

Resumo II

Instância: é um objeto que detém os dados e a referencia para as classes

Mensagem: Perguntar a um objeto alguma coisa é chamado de: "envio de mensagem". Isto é similar a solicitar uma sub-rotina.

Herança I

- Objetos definidos pela mesma classe, pode apresentar um comportamento especifico
- Observe: Se temos a classe : Moveis

- Esta classe pode instancia uma **cadeira** e uma **mesa**, porém a propriedade: **Num de gavetas** não se aplica aos dois objetos. Apenas a mesa

- Como fazer para instanciar objetos com atributos diferentes de uma mesma classe?

Herança II

- Uma forma bastate utilizada na POO está relacionada com o conceito de **herança**
- Classes mais especificas são modeladas partindo de uma classe mais genérica e assim "herdando" todo as suas características.

Herança III

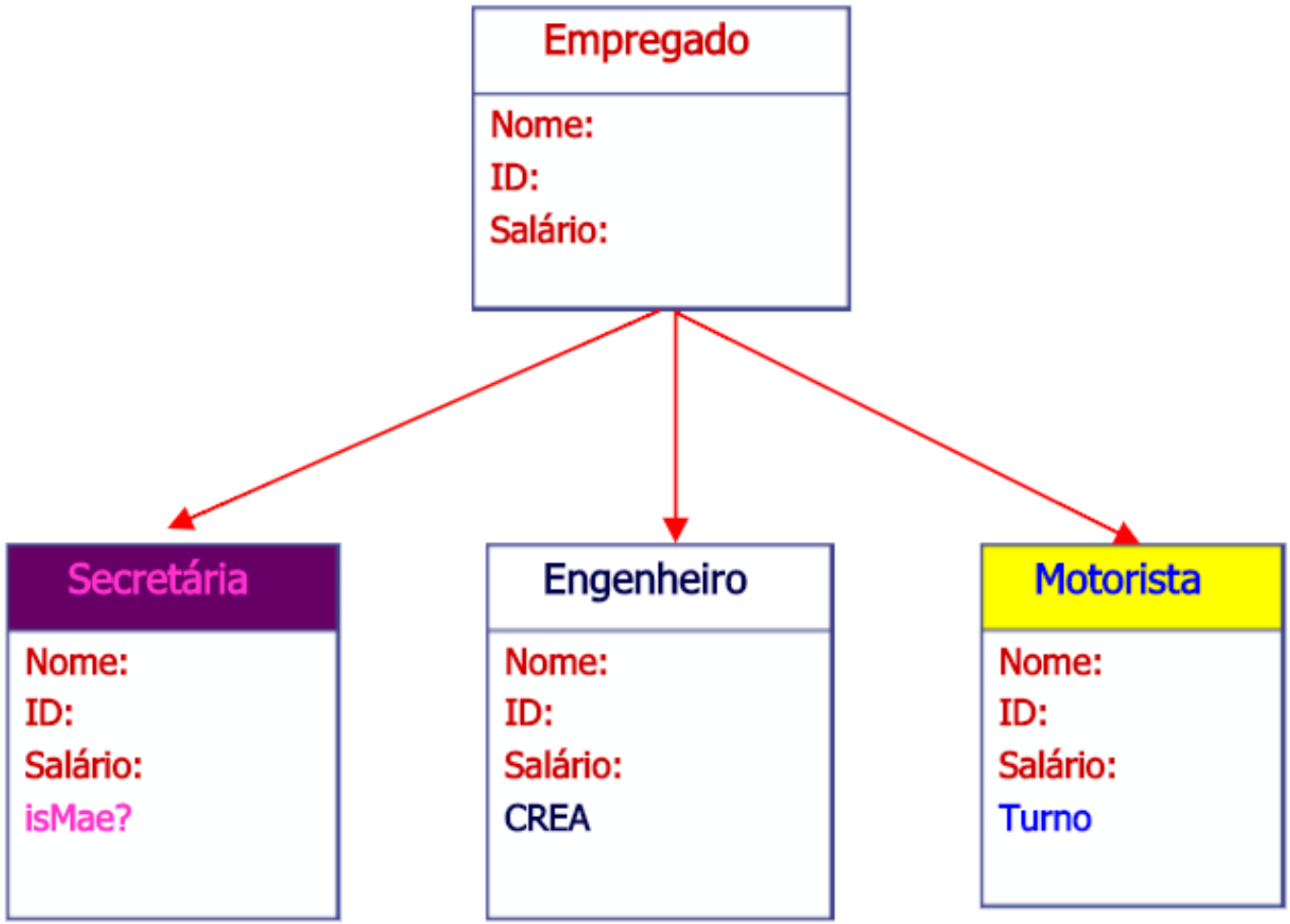
- **Herança** é um mecanismo simples e poderoso que permite que uma nova classe seja descrita a partir de outra classe já existente, herdando as suas propriedades.

- No nosso exemplo, teríamos a classe: Moveis (mais genérica) e as sub-classes: CMesa e CCadeira (cada uma com suas particularidades, podendo assim gerar seus objetos que "herdam" as características mais gerais da classe moveis.

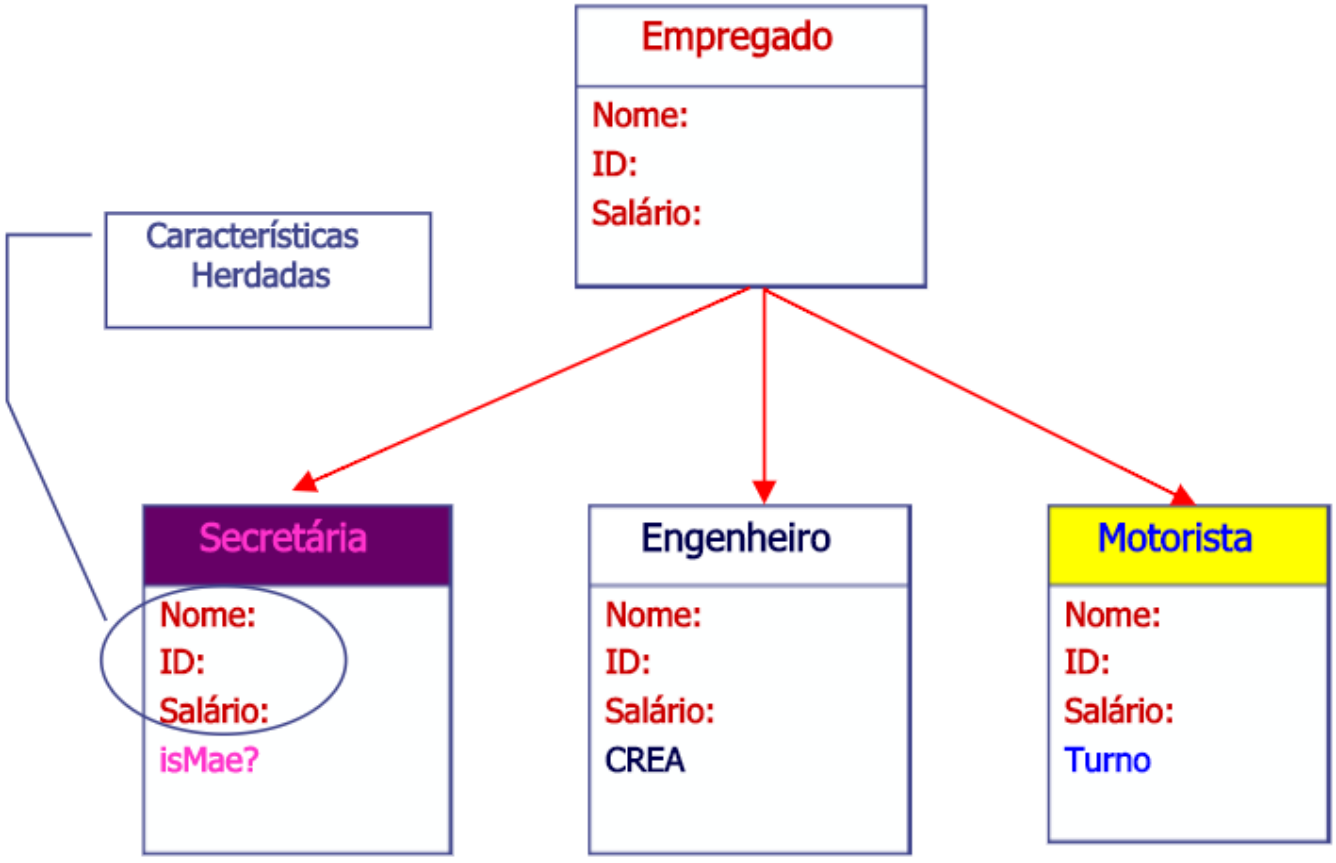
Herança IV

- Diz-se que uma subclasse herda os métodos e as variáveis de instância de uma superclasse.
  - A subclasse deve ser capaz de adicionar novos métodos e variáveis, além de redefinir os métodos herdados
  - Uma subclasse é uma especialização de uma superclasse
  - Uma instância da subclasse têm todas as propriedades de uma instância da superclasse, mais suas características particulares, e nunca menos

Herança V


Empregado
Nome:
ID:
Salário:
Secretária
Nome:
ID:
Salário:
isMae?
Engenheiro
Nome:
ID:
Salário:
CREA
Motorista
Nome:
ID:
Salário:
Turno

Herança VI


Empregado
Nome:
ID:
Salário:
Características Herdadas
Secretária
Nome:
ID:
Salário:
isMae?
Engenheiro
Nome:
ID:
Salário:
CREA
Motorista
Nome:
ID:
Salário:
Turno

FIM