



Curso de Engenharia da Computação

Paradigmas de Programação (60h/a)

Introdução à POO

Aula 02

Prof. Lenardo Chaves e Silva, D.Sc. 

lenardo@ufersa.edu.br

terça-feira, 21 de novembro de 2017



Universidade Federal Rural do Semi-Árido (UFERSA)

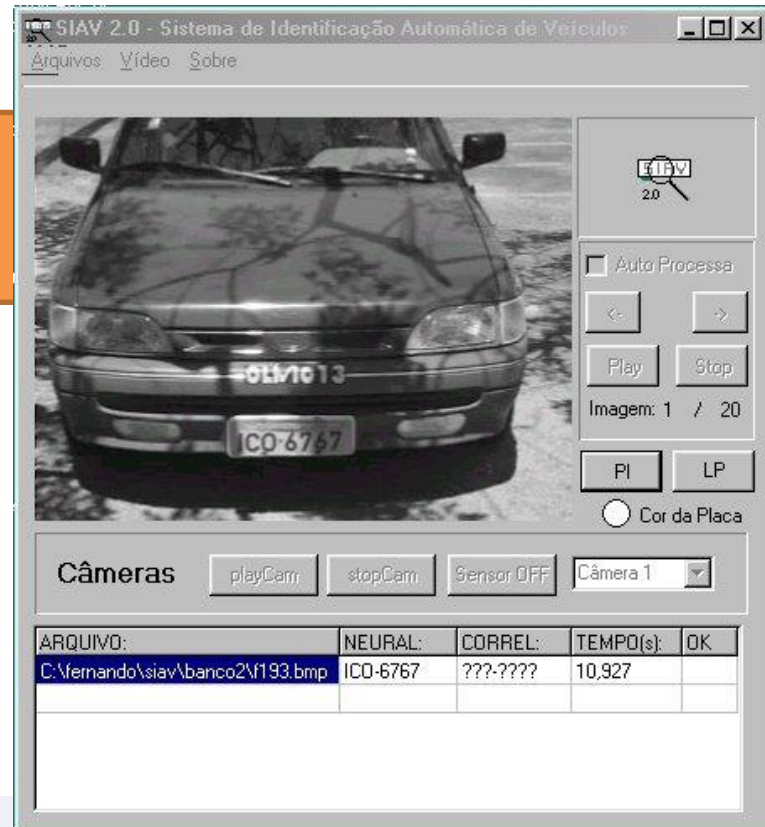
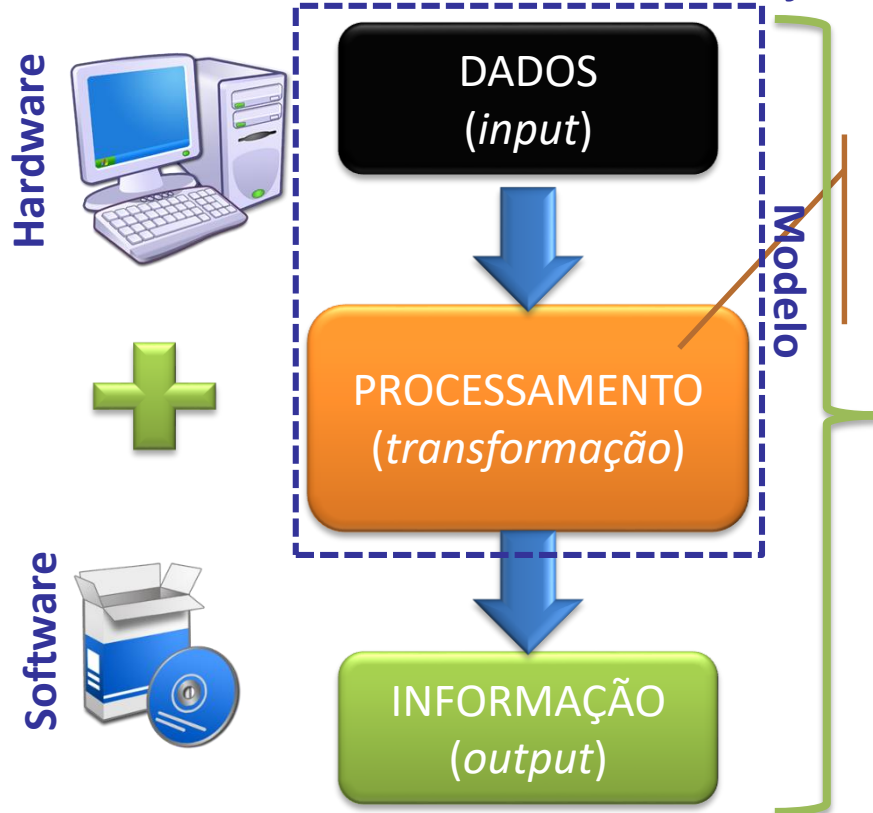
- ☛ Contextualização para POO:
 - ☐ O que é Computação?
 - ☐ Conceito e Características dos Modelos;
 - ☐ Exemplos de Modelos.
- ☛ Introdução à POO:
 - ☐ Conceitos Fundamentais e Básicos;
 - ☐ Importância da POO;
 - ☐ Papéis dos Programadores.
- ☛ Exercícios de Fixação.
- ☛ Referências.

Programação Orientada a Objetos (POO):

- ☛ Lida com LPs de Alto Nível (e.g., C++, Java, Python):
 - ☐ +/- independentes de máquina;
 - ☐ Execução:
 - ☐ Compilação (e.g., C++);
 - ☐ Interpretação (e.g., Python);
 - ☐ Combinação de ambas (e.g., Java).
- ☛ Baseada em Classes → família de objetos:
 - ☐ Objeto = {dados (i.e., atributos), operações (i.e., métodos);
 - Atributos → Particularidades.
 - Métodos → Comportamentos.

O que é Computação?

Processo de transformar dados em informação!



<http://www.lapsi.eletrou.ufersa.br/projetos/siav/siav2.htm>

Modelos: representações simplificadas (i.e., abstrações) de entidades do mundo real (e.g., itens, pessoas, animais, tarefas, conceitos, etc.), independentes de computadores/plataformas computacionais.

☛ Exemplo: ModeloEstabelecimento

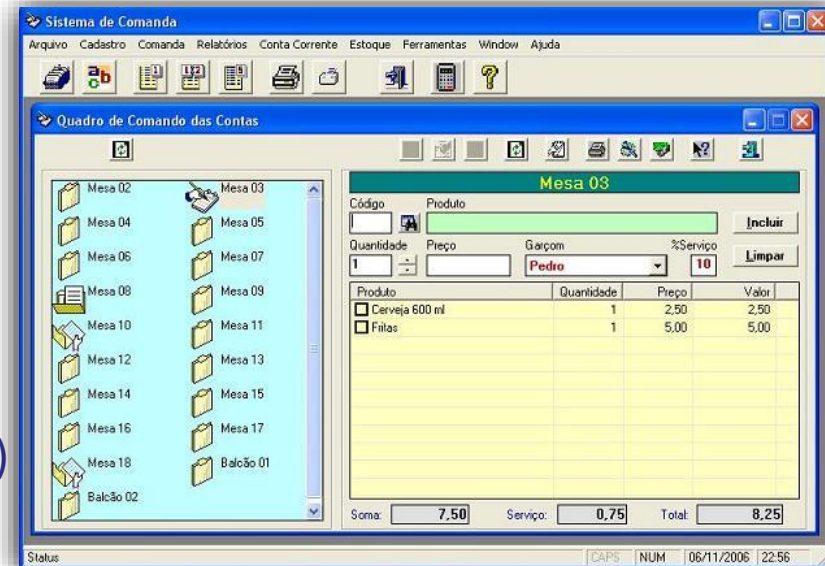
- ☐ Prato (e.g., tipoPrato/peso);
- ☐ Bebida (e.g., tipoBebida, valor, qtd);
- ☐ Mesa (e.g., numMesa, itens, total).

Dados dos Modelos:

- ☐ Os relevantes p/ o propósito da abstração (e.g., numMesa, itens e qtd.)
- ☐ Contraexemplo: nomeCliente.

Operações dos Modelos:

- ☐ Procedimentos para manipular (i.e., processar), em geral, os dados contidos no modelo (e.g., incluirPedido, alterarPedido, fecharPedido).



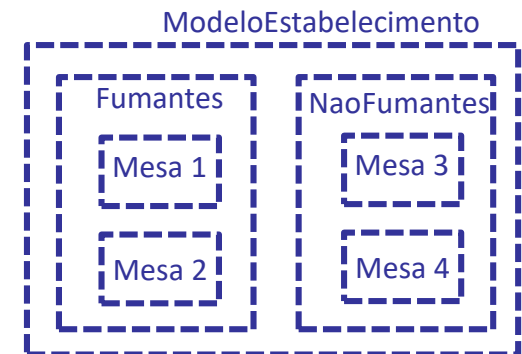
Modelos:

☛ Tipos:

- ☐ Dados e Operações, conjuntamente (**padrão**);
- ☐ Puramente de Dados (pouco uso):
 - Constantes (e.g., constantes matemáticas, restrição de valores).
- ☐ Puramente de Operações (comum):
 - Bibliotecas de operações (e.g., procedimentos, funções);
 - ➔ Manipulação de Geral (e.g., funções matemáticas).
 - Dispensam ➔ armazenamento de dados.

☛ Submodelos:

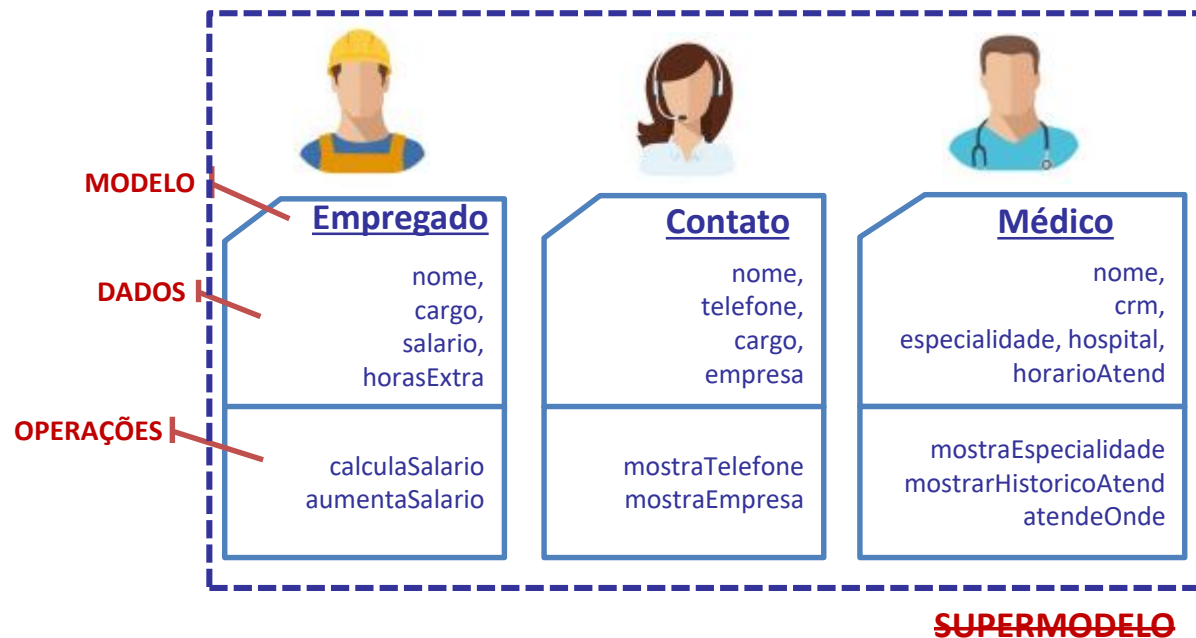
- ☐ Modelos dentro/parte de outros modelos;
Ex.: 1. MesaDoEstabelecimento (*dentro*);
2. Data ➔ {dia, mês, ano} (*parte de*).



Modelos:

☛ Simplificação → dependentes do Contexto de Uso:

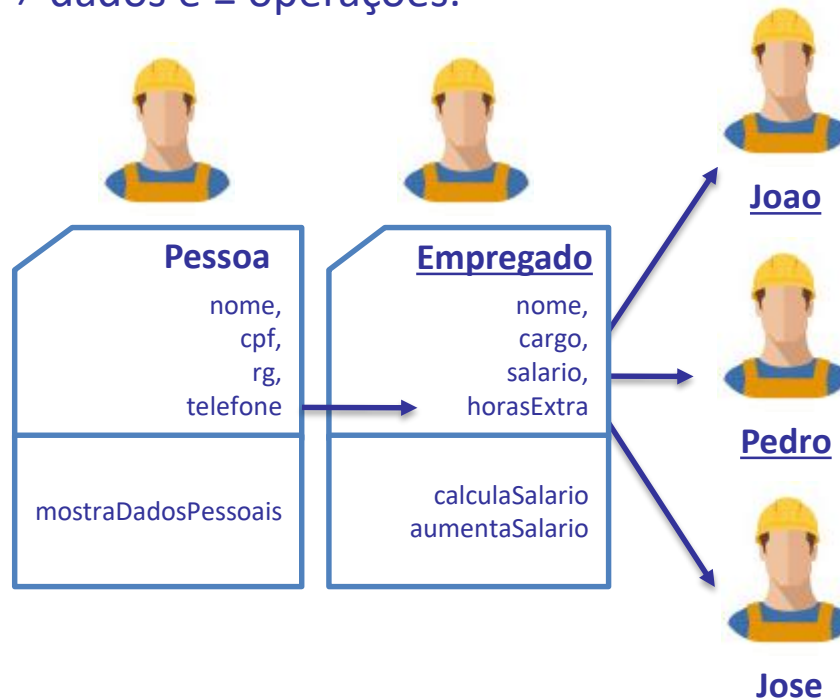
☐ Exemplo: Representação de Pessoa.



Modelos:

➔ Reuso:

- ❑ Modelo (único) → representar diferentes instâncias de uma entidade:
 - \neq dados e \equiv operações.



Programação Orientada a Objetos (POO)



Resumindo, o que é POO?

- ☛ Paradigma de Programação de Computadores;
- ☛ **Origem:** extensão do conceito de Modelo (criação e reuso);
- ☛ Baseado em **classes** e **objetos**:
 - ☐ Representar entidades do mundo real e processar dados;
 - ☐ **Dados:**
 - Tipos de dados nativos/primitivos (i.e., dependentes da LP);
 - Outros dados/modelos (i.e., criados por programadores).
- ☛ **Classes e Objetos vs. Modelos:**
 - ☐ Limitadas às tarefas em um PC e aos recursos da LP;
 - ☐ **Implementação:** armazenamento de dados → tipo e tamanho.
 - **Ex.:** Dados de Pessoa ← Paciente: ficha eletrônica vs. ficha manual.

Resumindo, o que é POO?

☛ Algumas Vantagens:

- ☐ Estruturação adequada dos dados;
- ☐ Simplicidade na execução de operações → dados:
 - **Ex.:** operações *CRUD* (*Create, Retrieve, Update e Delete*).
- ☐ Modularidade → divisão dos papéis por cada modelo;
- ☐ Facilidade na compreensão do funcionamento dos programas:
 - Criação/Manutenção/Evolução do código ← PROGRAMADORES.
- ☐ Modelagem dos Dados e Operações:
 - 👍 **Processamento:** + coeso, + rápido, - acoplado e - suscetível a erros;
 - 👁 **Implementação:** atenção às regras e limitações dos PCs e LPs.

Alguns Conceitos Fundamentais:

☛ **Abstração** → Ocultar detalhes da Implementação (complexidade):

☐ Analogia:



☛ **Encapsulamento** → Capacidade de ocultar ou restringir o acesso direto aos Dados de um Modelo:

☐ Atribuir responsabilidades ao Modelo e não ao Usuário:

▪ Modificar Dados ← Criar Operações.

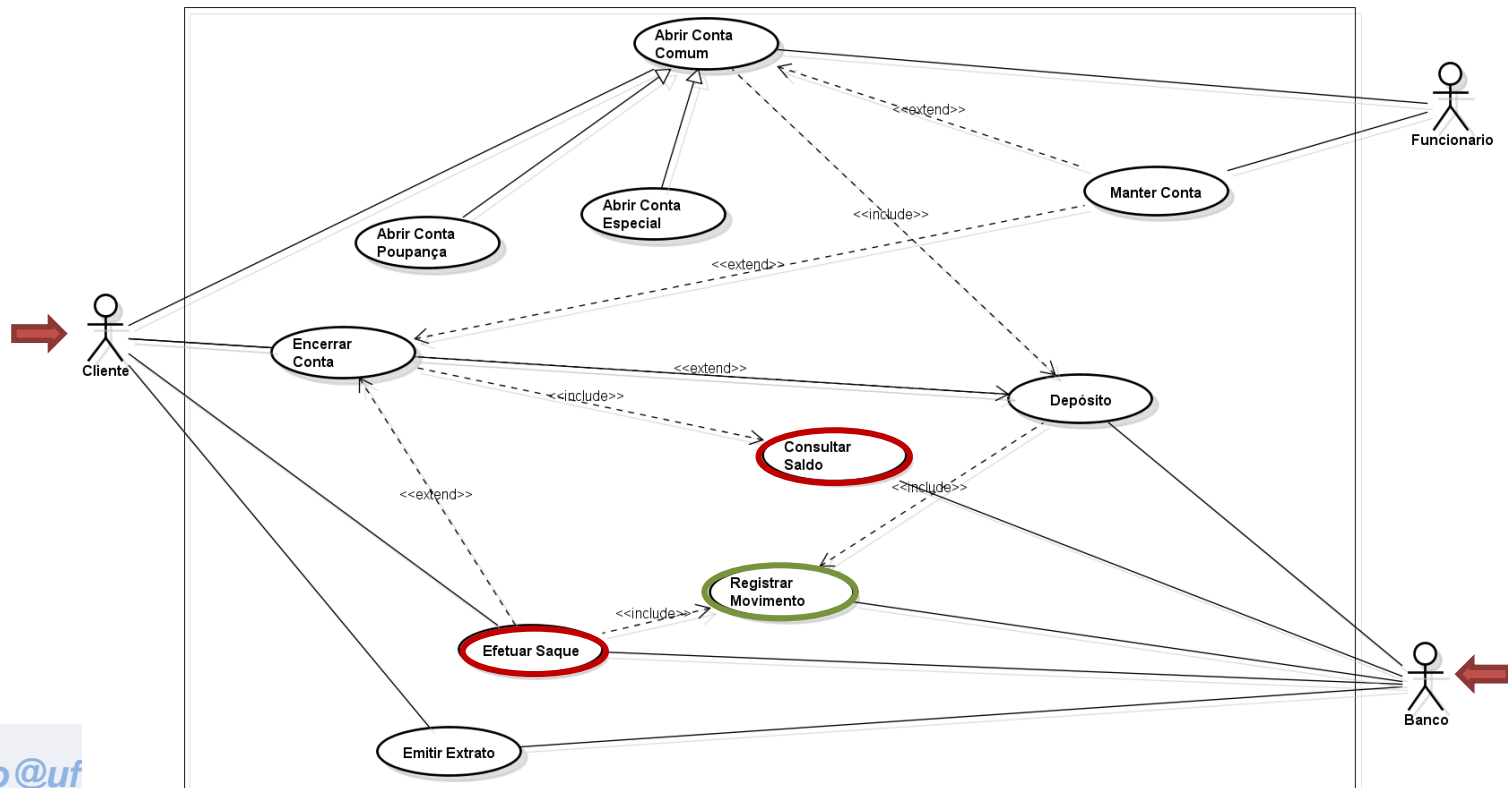
☐ Programas: + compreensíveis, + organizados e - erros;

☐ Boa prática de POO.

Alguns Conceitos Fundamentais:

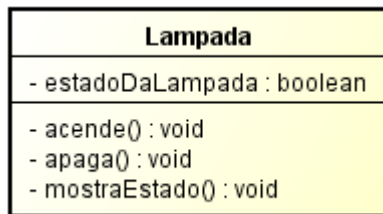
☛ **Encapsulamento** ➔ Restringir o acesso direto aos Dados:

☐ **Exemplo:** Modelo de uma Agência Bancaria.



Exemplos de Modelos:

1. Lâmpada Incandescente:



PSEUDOCÓDIGO

```

modelo Lampada // representa uma lâmpada em uso
início do modelo
    dado estadoDaLâmpada; // indica se está ligada ou não

    operação acende() // acende a lâmpada
        início
            estadoDaLâmpada = aceso;
        fim

    operação apaga() // apaga a lâmpada
        início
            estadoDaLâmpada = apagado;
        fim

    operação mostraEstado() // mostra o estado da lâmpada
        início
            se (estadoDaLâmpada == aceso)
                imprime "A lâmpada está acesa";
            senão
                imprime "A lâmpada está apagada";
            fim
        fim

fim do modelo
    
```

Exemplos de Modelos: 2. Conta Bancária Simples



ContaBancariaSimples

- nomeDoCorrentista : String
- saldo : Double
- contaEspecial : Boolean

- abreConta(nome : String, deposito : Double, especial : Boolean) : void
- abreContaSimples(nome : String) : void
- deposita(valor : Double) : void
- retira(valor : Double) : void
- mostraDados() : void

PSEUDOCÓDIGO

modelo ContaBancariaSimplificada

início do modelo

 dado nomeDoCorrentista, saldo, contaEspecial; // dados da conta

 // Inicializa simultaneamente todos os dados do modelo

 operação abreConta(nome, depósito, especial) // argumentos para esta operação

 início

 // Usa os argumentos passados para inicializar os dados do modelo

 nomeDoCorrentista = nome;

 saldo = depósito;

 contaEspecial = especial;

 fim

 // Inicializa simultaneamente todos os dados do modelo, usando o nome

 // passado como argumento e os outros valores com valores default

 operação abreContaSimples(nome) // argumento para esta operação

 início

 nomeDoCorrentista = nome;

 saldo = 0.00;

 contaEspecial = falso;

 fim

 // Deposita um valor na conta

 operação deposita(valor)

 início

 saldo = saldo + valor;

 fim

 // Retira um valor da conta

 operação retira(valor)

 início

 se (contaEspecial == falso) // A conta não é especial !

 início

 se (valor <= saldo) // se existe saldo suficiente...

 saldo = saldo - valor; // faz a retirada.

 fim

 senão // A conta é especial, pode retirar à vontade !

 saldo = saldo - valor;

 fim

Exemplos de Modelos:

3. Data:



| Data |
|--|
| <ul style="list-style-type: none"> - dia : int - mes : int - ano : int |
| <ul style="list-style-type: none"> - inicializaData(d : int, m : int, a : int) : void - dataValida(d : int, m : int, a : int) : Boolean - mostraData() : String |

PSEUDOCÓDIGO

```

modelo Data
início do modelo
    dado dia,mês,ano; // componentes da data

    // Inicializa simultaneamente todos os dados do modelo
    operação inicializaData(umDia,umMês,umAno) // argumentos para esta operação
    início
        // Somente muda os valores do dia, mês e ano se a data passada for válida
        se dataÉVálida(umDia,umMês,umAno) // Repassa os argumentos para a operação
            início
                dia = umDia;
                mês = umMês;
                ano = umAno;
            fim
        // Se a data passada não for válida, considera os valores sendo zero
        senão
            início
                dia = 0;
                mês = 0;
                ano = 0;
            fim
    fim

    operação dataÉVálida(umDia,umMês,umAno) // argumentos para esta operação
    início
        // Se a data passada for válida, retorna verdadeiro
        se ((dia >= 1) e (dia <= 31) e (mês >= 1) e (mês <= 12))
            retorna verdadeiro;
        // Senão, retorna falso
        senão
            retorna falso;
    fim
    fim
    
```


Exemplos de Modelos:

4. Registro Acadêmico:



RegistroAcademico

- nomeDoAluno : String
 - numeroDeMatricula : int
 - dataDeNascimento : Data
 - eBolsista : Boolean
 - anoDeMatricula : int
-
- inicializaRegistro(nome : String, matricula : int, data : Data, bolsa : Boolean) : void
 - calculaMensalidade() : void
 - mostraRegistro() : void

PSEUDOCÓDIGO

```

modelo RegistroAcademico
início do modelo
    // Dados do registro acadêmico
    dado nomeDoAluno, númeroDeMatricula;
    dado dataDeNascimento, éBolsista, anoDeMatricula;

    // Inicializa simultaneamente todos os dados do modelo, passando argumentos
    operação inicializaRegistro(oNome, aMatricula, aData, temBolsa, qualAno)
    início
        // Usa os argumentos para inicializar os valores no modelo
        nomeDoAluno = oNome;
        númeroDeMatricula = aMatricula;
        dataDeNascimento = aData;
        éBolsista = temBolsa;
        anoDeMatricula = qualAno;
    fim

    operação calculaMensalidade() // calcula e retorna a mensalidade
    início
        mensalidade = 400;
        se (éBolsista) mensalidade = mensalidade / 2;
        retorna mensalidade;
    fim

    operação mostraRegistro() // mostra os dados do registro acadêmico
    início
        imprime "Nome do aluno:";
        imprime nomeDoAluno;
        imprime "Número de Matricula:";
        imprime númeroDeMatricula;
        imprime "Data de Nascimento:";
        dataDeNascimento.mostraData(); // pede à data que se imprima !
        se (éBolsista == verdadeiro) imprime "O aluno é bolsista.";
        senão imprime "O aluno não é bolsista.";
        imprime "Ano de Matricula:";
        imprime anoDeMatricula;
    fim
fim do modelo
    
```


Conceitos Básicos:

- ☛ **Classes:** equivalem aos modelos de entidades do mundo real.
 - ❑ Estruturas POO → implementar Modelos = {dados, operações};
 - ❑ **Nome da Classe:** fácil de associar ao modelo.
 - Ex.: (Contexto de um Banco) ← **Conta** ou **ContaBancaria**?
 - ❑ Implementação: recursos e regras da LP (e.g., Java, C++, Python);
 - ☛ **Objetos:** derivados a partir de uma classe (**ou conjunto**).
 - ❑ Tipo de dado específico;
 - ❑ Instância = materialização da classe:
 - Analogia:** Planta de Casa (classe) ← Casa (objeto).
 - ❑ Manipulação: referência = variável do “tipo” da classe:
 - Analogia:** Número da Casa.
- Ex.: Pessoa (classe) ← Empregado (classe) ← João (objeto).

Conceitos Básicos:

- ☛ **Atributos/Campos:** dados contidos na classe ← encapsulados.
 - ☐ **Nome:** referência ao dado;
 - ☐ **Tipo de Dado:**
 - Nativo → própria LP;
 - Abstrato → classe existente na LP ou classe criada p/ programador.
 - ☐ **Variáveis:** valores contidos na classe → auxiliar métodos.



Joao
Engenheiro
R\$ 6.000
20 h



Pedro
Pedreiro
R\$ 2.000
40 h



Jose
Servente
R\$ 1.000
40 h

Conceitos Básicos:

☛ **Métodos:** operações pertencentes à classe.

❑ Execução: chamados/invocados explicitamente via código:

- Sinônimo: “*Mensagem*”;
- Por quem? **R.** Própria classe ou outras classes.
- **Ex.: RegistroAcademico.**



PSEUDOCÓDIGO

```
operação mostraRegistro() // mostra os dados do registro acadêmico
início
    imprime "Nome do aluno:";
    imprime nomeDoAluno;
    imprime "Número de Matrícula:";
    imprime númeroDeMatrícula;
    imprime "Data de Nascimento:";
    dataDeNascimento.mostraData(); // pede à data que se imprima !
    se (éBolsista == verdadeiro) imprime "O aluno é bolsista.";
    senão imprime "O aluno não é bolsista.";
    imprime "Ano de Matrícula:";
    imprime anoDeMatrícula;
fim
fim do modelo
```

Conceitos Básicos:

☛ **Métodos:** operações pertencentes à classe.

☐ Argumentos (opcional):

- Valores de tipos nativos ou referências à instâncias de classes;
- Declarados ← declaração dos Métodos;
- Não existe restrição da quantidade: **↑ ALERTA!**

➤ **Dividir a responsabilidade com outros métodos!**

☐ Retorno (opcional): valores ou instâncias de classes.

- Sem Retorno / Único Valor / ~~+1 Valor~~.

Quando é importante?

☛ Problemas:

- ☐ ▲ Complexidade:
 - **Contraexemplo:** Equação 2º Grau.

☛ Aplicações:

- ☐ Representar conjunto de dados dependentes ou interligados...
- ☐ Aplicar conjunto de rotinas específicas → Conjunto de dados.
 - **Ex.:** Data e RegistroAcademico/ContaBancaria.

Papéis dos Programadores?

1. Programador de Classes:

- ☐ Criação de novas classes → novos modelos:
 - **Encapsular:** atributos e métodos.
- ☐ Definição do contrato: criador \leftrightarrow usuário:
 - **Relacionar:** métodos? \leftrightarrow atributos?
- ☐ *God-Class* (super-modelos):
 - **Evitar:** sobrecarga (i.e., responsabilidades sob medida).
 - **Prever:** quais aplicações?

2. Programador-Usuário:

- ☐ Utilização de classes existentes → LP/outras programadoras.



Exercícios de Fixação - POO



1. A operação *abreConta* do modelo *ContaBancariaSimplificada* (listagem slide 14) permite que alguém crie uma conta bancária passando como argumento um valor negativo, criando uma conta já em débito. Modifique a operação abre conta para que se alguém passar um saldo inicial negativo, que este seja considerado como zero.
2. Crie um modelo *Livro* que represente os dados básicos de um livro, sem se preocupar com a sua finalidade.
3. Usando o resultado do exercício 2 como base, crie um modelo *LivroDeBiblioteca* que represente os dados básicos de um livro de uma biblioteca, que pode ser emprestado a leitores.

Exercícios de Fixação - POO



4. Escreva um modelo *Empregado* que represente um empregado de uma empresa qualquer. Considere que os dados *nome*, *departamento*, *horasTrabalhadasNoMes* e *salarioPorHora* devam ser representados, e que ao menos as operações *mostraDados* e *calculaSalarioMensal*.
5. Modifique a operação *calculaSalarioMensal* no modelo *Empregado* (exercício 4) para que todos os empregados do departamento “Diretoria” tenham 10% de bônus salarial.
6. Crie um modelo *EquacaoSegundoGrau* que contenha somente uma operação, a que calcula as raízes da equação por meio da fórmula de Bhaskara. Considere que os valores de *a*, *b* e *c* serão passados como parâmetros para tal operação.


$$\Delta = b^2 - 4.a.c$$

$$x = \frac{-b \pm \sqrt{\Delta}}{2a}$$

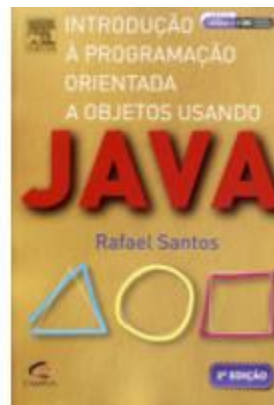
Extras

Modifique a operação *dataÉVálida* do modelo *Data* (slide 15) para que esta considere o valor máximo para o *dia*, dependendo do *mês*. Para fevereiro o valor máximo deve ser calculado em função do ano ser bissexto ou não.

Dica: Anos bissextos (tendo 29 dias em fevereiro) são divisíveis por quatro, a não ser que sejam divisíveis por 100. Anos que podem ser divididos por 400 também são bissextos. Desta forma, 1964 e 2000 são bissextos, mas 1900 não é bissexto. A operação de divisibilidade pode ser implementada pela função módulo, representada pelo sinal %.

-  SANTOS, R. Introdução à programação orientada a objetos usando JAVA. 2. ed. Rio de Janeiro: Campus, 2003. 336p.

Capítulo 1



Próxima Aula...

➤ Aula 03:

- Introdução a Java através de exemplos.



Importante! *Você* é responsável por aprender detalhes adicionais da linguagem Java. O que faremos em aula não é uma cobertura completa da linguagem.