PROGRAMAÇÃO ORIENTADA A OBJETOS

Prof. Edkallenn

Sistemas de Informação - Uninorte

Programação Orientada a Objetos

- **Prof. Edkallenn Lima**
- edkallenn@yahoo.com.br (somente para dúvidas)
- **Blogs**:
  - http://professored.wordpress.com (Computador de Papel – O conteúdo da forma)
  - http://professored.tumblr.com/ (Pensamentos Incompletos)
  - http://umcientistaporquinzena.tumblr.com/ (Um cientista por quinzena)
  - http://eulinoslivros.tumblr.com/ (Eu li nos livros)
  - http://linabiblia.tumblr.com/ (Eu li na Bíblia)
- **YouTube**:
  - https://www.youtube.com/user/edkallenn
  - https://www.youtube.com/channel/UC-pD2gnahhxUDVuTAA0DHoQ
- **Redes Sociais**:
  - http://www.facebook.com/edkallenn
  - http://twitter.com/edkallenn ou @edkallenn
  - https://plus.google.com/u/0/113248995006035389558/posts
  - Instagram: http://instagram.com/edkallenn ou @edkallenn
  - Foursquare: https://pt.foursquare.com/edkallenn
  - Pinterest: https://br.pinterest.com/edkallenn/
- **Telefones**:
  - 68 98401-2103 (CLARO e Whatsapp) e 68 3212-1211.

Os exercícios devem ser enviados SEMPRE para o e-mail:
edkevan@gmail.com
ou para o e-mail:
edkallenn.lima@uninorteac.com.br

Apresentação

- Nome
- Trabalha? Onde? O que você faz?
- Experiência com informática?
- É programador ou já teve contato com alguma linguagem de programação?

Carga Horária

- Semanal: 3 horas
- Expositivas: 30 horas
- Atividades Supervisionadas: 30 horas
- Total: 60 HORAS

- Muito conteúdo
- Pouco Tempo

Objetivo Geral

- Preparar o aluno para a prática do desenvolvimento de aplicações orientadas a objetos, habilitando o aluno a desenvolver sistemas sob a perspectiva da orientação a objetos.

Objetivos Específicos

- Compreender os conceitos básicos da Programação Orientada a Objetos;
- Apresentar o processo de desenvolvimento orientado a objetos
- Modelar e implementar em uma linguagem de programação orientada a objetos, problemas de pequena complexidade
- Adquirir domínio básico de uma linguagem de programação orientada a objetos através da aplicação prática dos conceitos aprendidos.

Ao final...

- você será um desenvolvedor diferenciado e mais preparado para usar o paradigma orientado a objetos da melhor forma possível
- O foco será explicar da melhor forma possível todos os conceitos deste paradigma de programação.

Procedimento/Metodologia

- Aulas teóricas expositivas;
- Elaboração e correção de exercícios em sala de aula;
- Debates
- Resolução de exercícios
- Exposição da aplicabilidade disciplina na profissão;
- Estímulo ao desenvolvimento de aplicações utilizando os conceitos da Orientação a Objetos

Ementa

- Conceitos básicos da Orientação a Objetos;
- Classes
- Membros de Classe
- Atributos e Métodos
- Encapsulamento
- Construtores e Sobrecarga
- Introdução à Linguagem Orientada a Objeto
- Estruturas de decisão e controle – condicionais
- Estruturas de decisão e controle – repetição
- Herança / Composição
- Polimorfismo
- Classes Abstratas e Interfaces
- Pacote de classes
- Tratamento de exceções
- Enum
- Generics.

Avaliação

Prova = **70%**

Trabalho = **30%**

N1 e N2

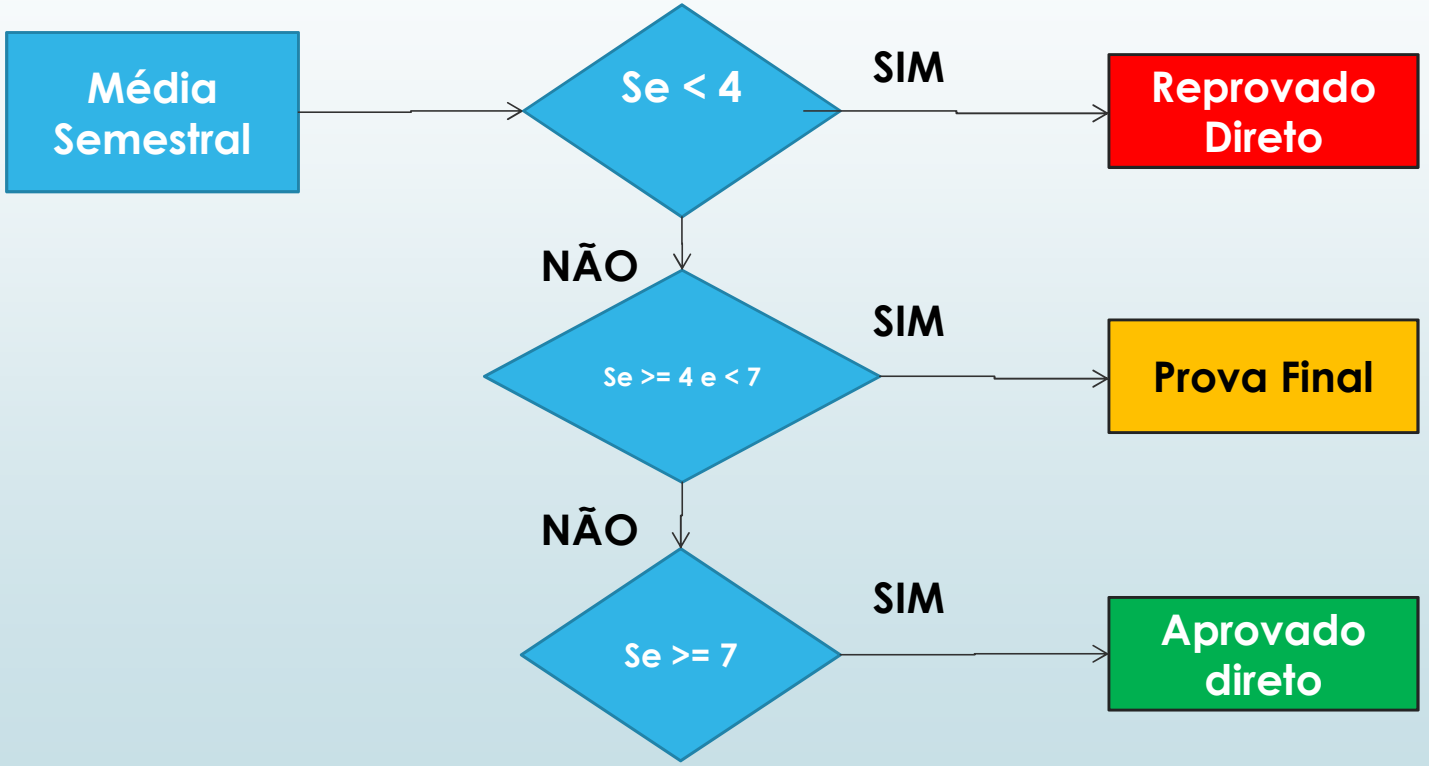
Avaliação/POO

- 40% Trabalhos e exercícios + 60%Prova
- Data da nova N1: Cf. calendário
- Data da N2: Cf. calendário
- Data da 2ª chamada: Cf. calendário
- Final: Cf. calendário
- Calendário normal da faculdade

Segunda Chamada

- Os alunos que não comparecem às provas bimestrais estão automaticamente inseridos na **segunda chamada.**
- **IMPORTANTE:** A prova de segunda chamada deve ser composta por **10 questões objetivas/subjetivas** com conteúdo do bimestre que o aluno perdeu.
- **Os 4,0 pontos de trabalho devem ser levados para a segunda chamada.**

Critérios para a aprovação



Critérios para aprovação

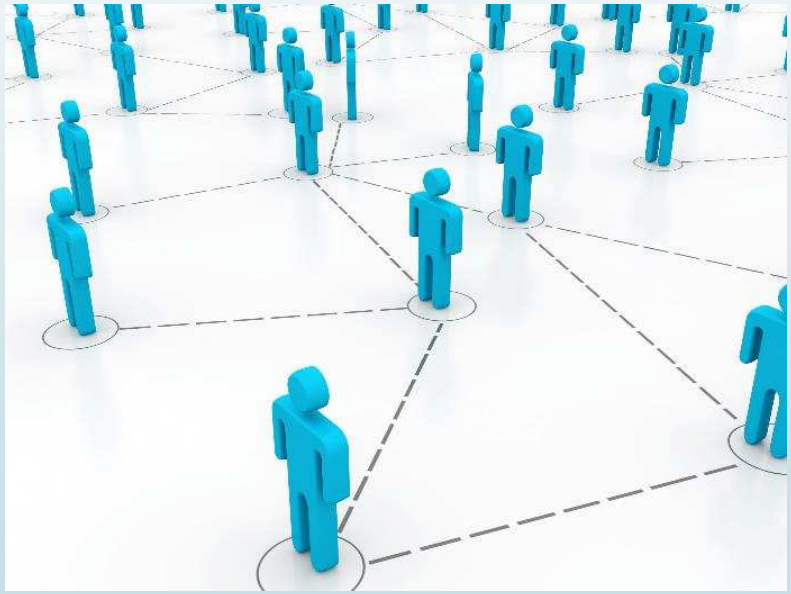
- Para saber a nota que o aluno precisa tirar na prova final basta equiparar a média semestral a 10.

Exemplo:

- Média 1.º bimestre: 7,0
- Média 2.º bimestre: 5,0
- Média Semestral: 6,0
- 6,0 equiparado a 10,0 → 4,0

Compartilhamento de informações

- Dropbox (fazer lista com e-mails)
- Portal
- Blog do professor (Computador de papel)
- Mailing-list da turma (ou grupo/comunidade nas redes sociais)

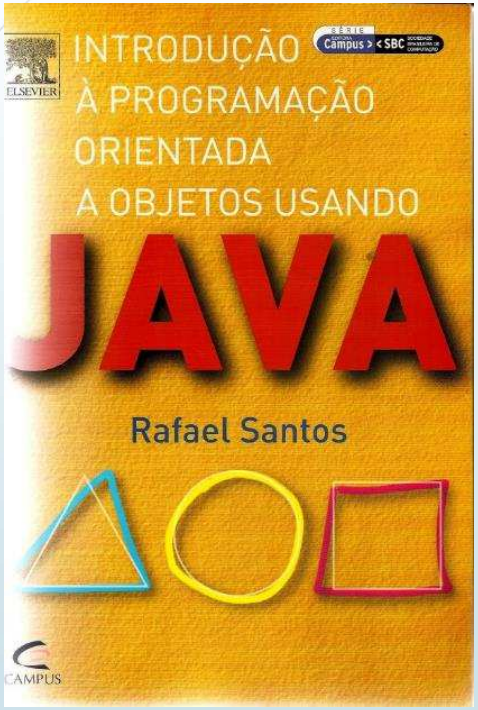


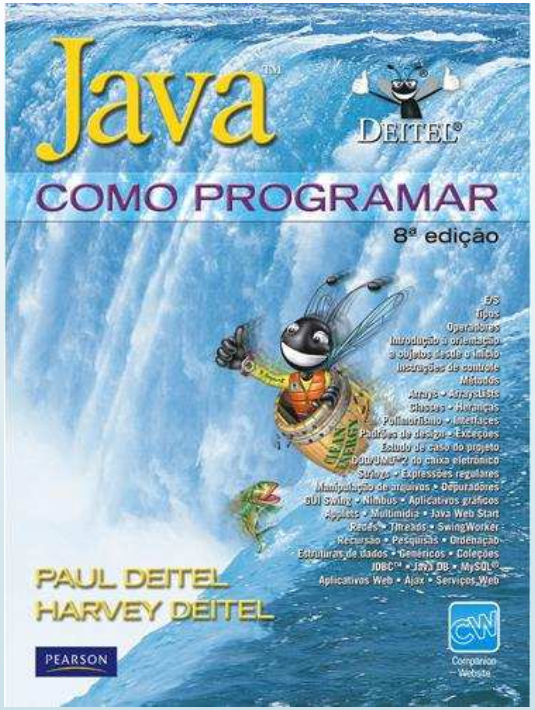
Referências Básicas

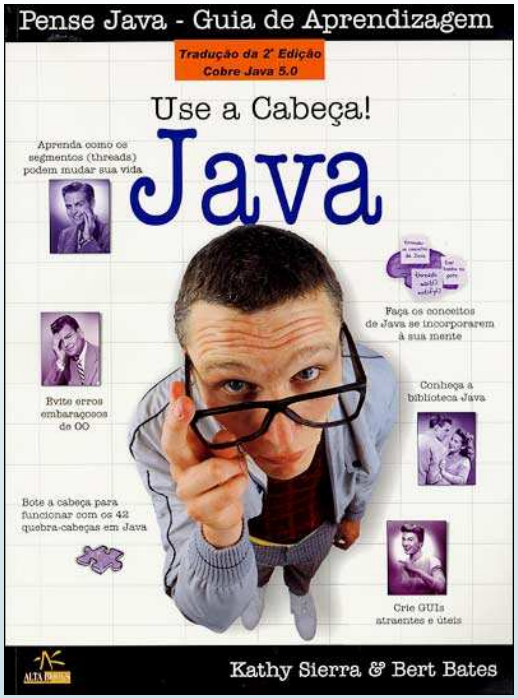
Livros

- DEITEL, Harvey M.; DEITEL, Paul J. Java, como programar. 8. ed. São Paulo: Pearson Education do Brasil, 2010.
- KATHY SIERRA & BERT BATES. Use a Cabeça! Java. Alta Books. 2007
- Cornell, Gary / Horstmann, Cay S. Core Java - Vol. 1 - Fundamentos - 8ª Ed. São Paulo. Pearson Education. 2010
- Santos. Rafael. Introdução à Programação Orientada a Objetos usando Java. 2ª ed. Rio de Janeiro. Elsevier. 2013
- Coelho, Alex. Java com Orientação a Objetos. Rio de Janeiro. Editora Ciência Moderna Ltda. 2012.
- Sintes, Anthony. Aprenda Programação Orientada a Objetos em 21 dias. São Paulo: Pearson Education do Brasil, 2002.
- ANICHE, Mauricio. Orientação a Objetos e SOLID para ninjas. Editora Casa
- do Código, 166 p.
- CARVALHO, Thiago Leite e. Orientação a objetos: Aprenda seus conceitos e suas aplicabilidades de forma efetiva, editora Casa do Código, 238 p.
- LARMAN, Craig. Utilizando UML e padrões: Uma introdução a análise e ao projeto orientados. Porto Alegre: Bookman, 2007.
- C# e orientação a objetos. Apostila K19 treinamentos. 2013.
- Use a Cabeça C# - 2ª Ed. Stellma, Andrew. Greene, Jennifer. Alta Books. 2010.
- C# Como programar. Deitel & Deitel. Makron Books. 2002.

Livros


INTRODUÇÃO
À PROGRAMAÇÃO
ORIENTADA
A OBJETOS USANDO
JAVA
Rafael Santos
ELSEVIER
CAMPUS

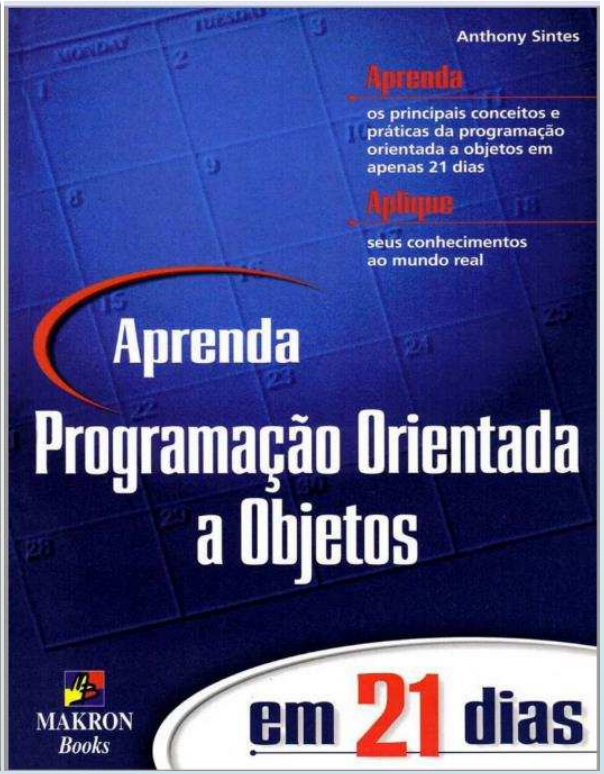

Java
DEITEL
COMO PROGRAMAR
8ª edição
PAUL DEITEL
HARVEY DEITEL
PEARSON

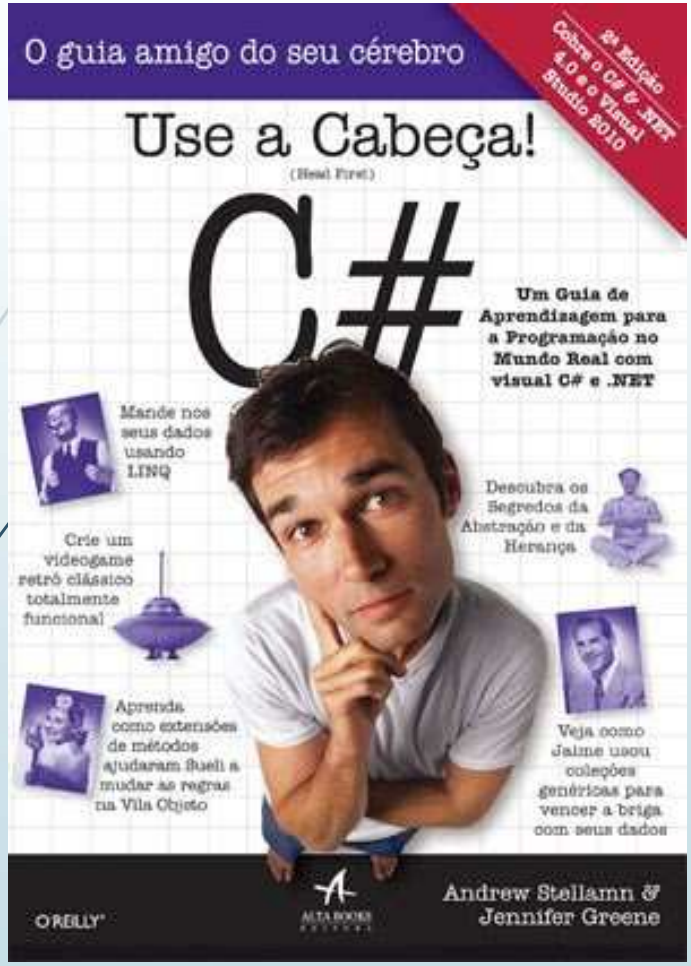

Pense Java - Guia de Aprendizagem
Tradução da 2ª Edição
Cobre Java 5.0
Use a Cabeça!
Java
Kathy Sierra & Bert Bates

Livros


Cay S. Horstmann · Gary Cornell
Core Java
8ª edição
Java
Sun
Volume I — Fundamentos
PEARSON


Alex Coelho
JAVA
com Orientação a Objetos
EDITORA CIÊNCIA MODERNA


Anthony Sintes
Aprenda
os principais conceitos e práticas da programação orientada a objetos em apenas 21 dias
Aplique
seus conhecimentos ao mundo real
Aprenda
Programação Orientada a Objetos
em 21 dias
MAKRON Books


O guia amigo do seu cérebro
2ª Edição
Cobre o C# 4.0 e o Visual Studio 2010
Use a Cabeça!
(Head First)
C#
Um Guia de Aprendizagem para a Programação no Mundo Real com visual C# e .NET
Mande nos seus dados usando LINQ
Descubra os Segredos da Abstração e da Herança
Crie um videogame retrô clássico totalmente funcional
Aprenda como extensões de métodos ajudaram Bueli a mudar as regras na Vila Objeto
Veja como Jaime usou coleções genéricas para vencer a briga com seus dados
O'REILLY
Andrew Stellamn & Jennifer Greene


C# COMO PROGRAMAR
Apresentando .NET e Web Services
MAKRON Books
PEARSON Education
DEITEL™
DEITEL
DEITEL
LISTFIELD
NIETO
YAEGER
ZLATKINA

Internet

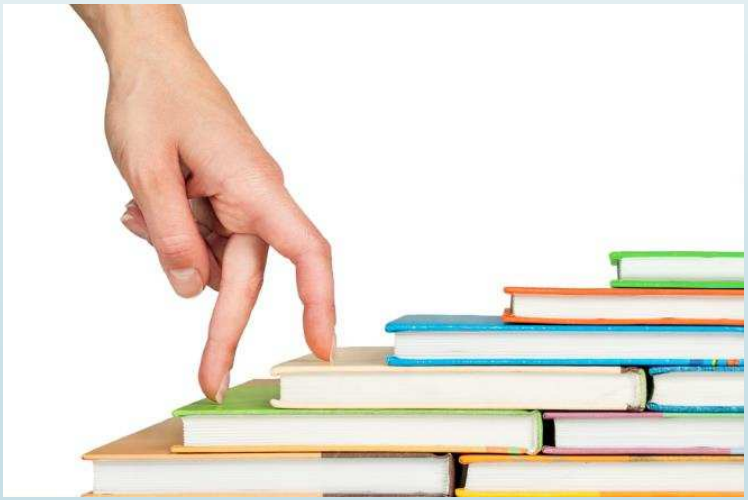
http://jcp.org/en/home/index
http://www.devmedia.com.br/java
http://www.sobresites.com/java/usuarios.htm

Stackoverflow.com

http://www.guj.com.br/
http://javafree.uol.com.br
http://soujava.org.br/
http://www.java.net/

Importante

- Empenho
- Estudo
- Dedicação
- Base sólida = futuro promissor

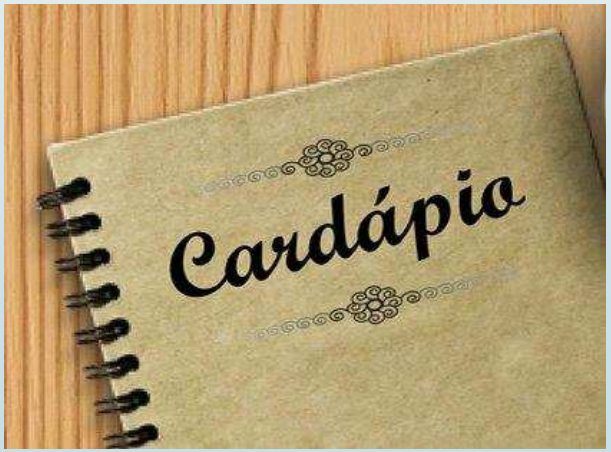


Proposta de Trabalho

- Nivelamento de conceitos perdidos e/ou não aprendidos
- Enfoque teórico e prático com uso muitos exercícios, realizados em sala e em casa para fixação
- Desenvolvimento de pequenos projetos

Conteúdo de OO

- Introdução à orientação a objetos
- Conceitos de orientação a objetos
- Classes e objetos
- Atributos e métodos
- Abstração e encapsulamento
- Interfaces e classes abstratas
- Relacionamento entre objetos: composição, associação, dependência e herança
- Herança, dynamic binding e polimorfismo
- Type casting
- Construtores
- Modelagem/Abstração
- Exceções
- Reutilização de Código



Linguagens utilizadas para expor os conceitos

- Java
- C#

==========

- Nenhuma linguagem em si será o alvo deste curso, pois os conceitos aqui explicados podem ser aplicados em qualquer uma que implemente tal paradigma.

==========

- Para alguns conceitos ficarem mais claros, é necessário demonstrá-los em uma linguagem de programação que segue o paradigma orientado a objeto.
- Neste caso, nada melhor que usar Java e C# como linguagens de exemplo, pois são as mais utilizadas no mercado no momento.

Ferramentas

- Java SDK
- Sublime ou Visual Code
- Eclipse (ou Netbeans)
- Visual Studio (Visual C#)
- TODOS GRATUITOS

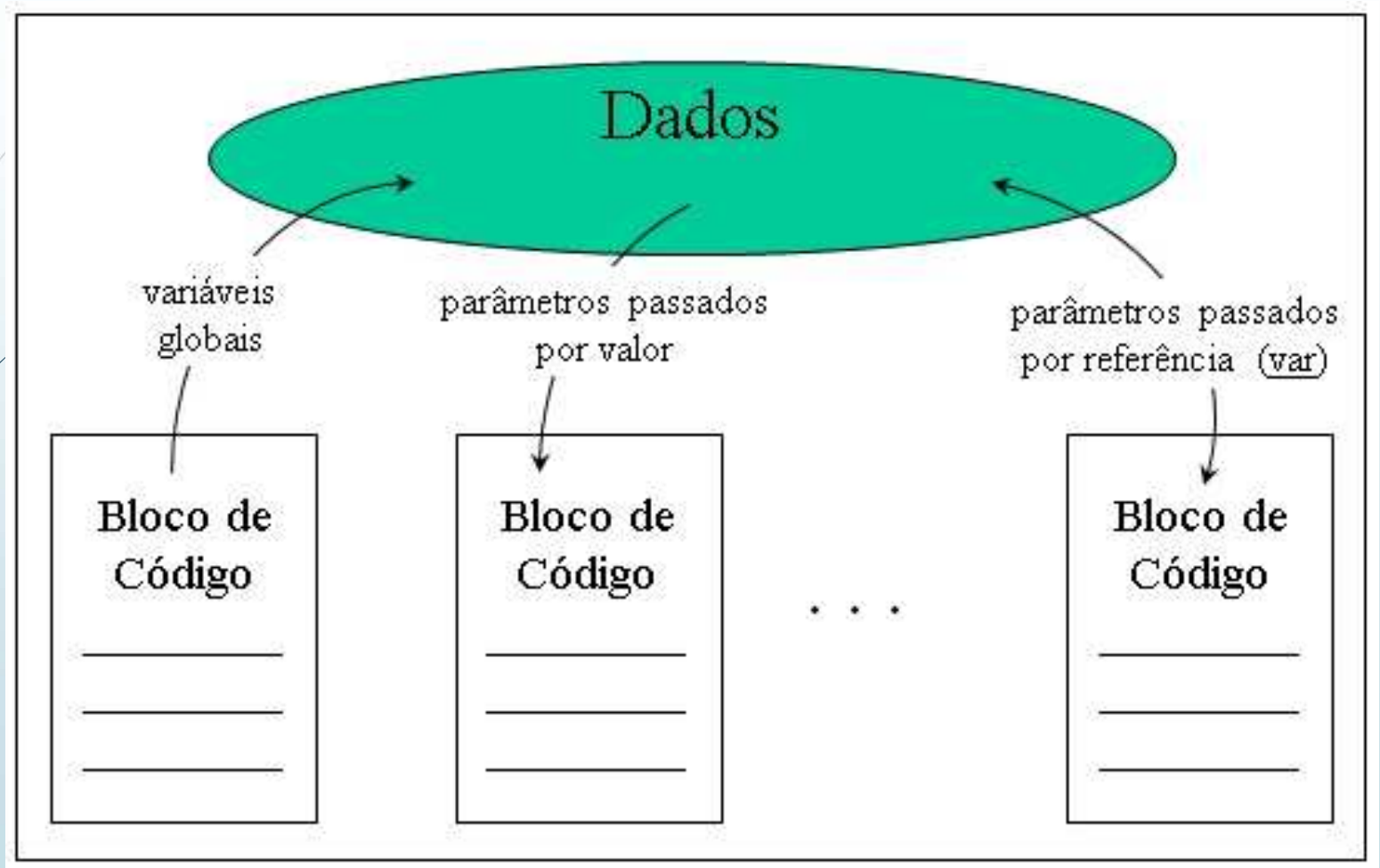
Então...

- Falar:
  - Nome
  - Expectativas na matéria
  - Expectativas no Curso (imaginadas/atendidas)
  - Dificuldade com Algoritmos?
  - Dificuldade com Programação

Programação Orientada a Objetos

- Programação Estruturada
- Base:
  - Sequência: Uma tarefa é executada após a outra, linearmente.
  - Decisão: A partir de um teste lógico, determinado trecho de código é executado, ou não.
  - Iteração: A partir de um teste lógico, determinado trecho de código é repetido por um número finito de vezes.
  - **Vantagens**
    - É fácil de entender. Muito usada em cursos introdutórios de programação.
    - Execução mais rápida.
  - **Desvantagens**
    - Baixa reutilização de código
    - Códigos mais confusos: Dados misturados com comportamento

Programação Orientada a Objetos



Programação Orientada a Objetos

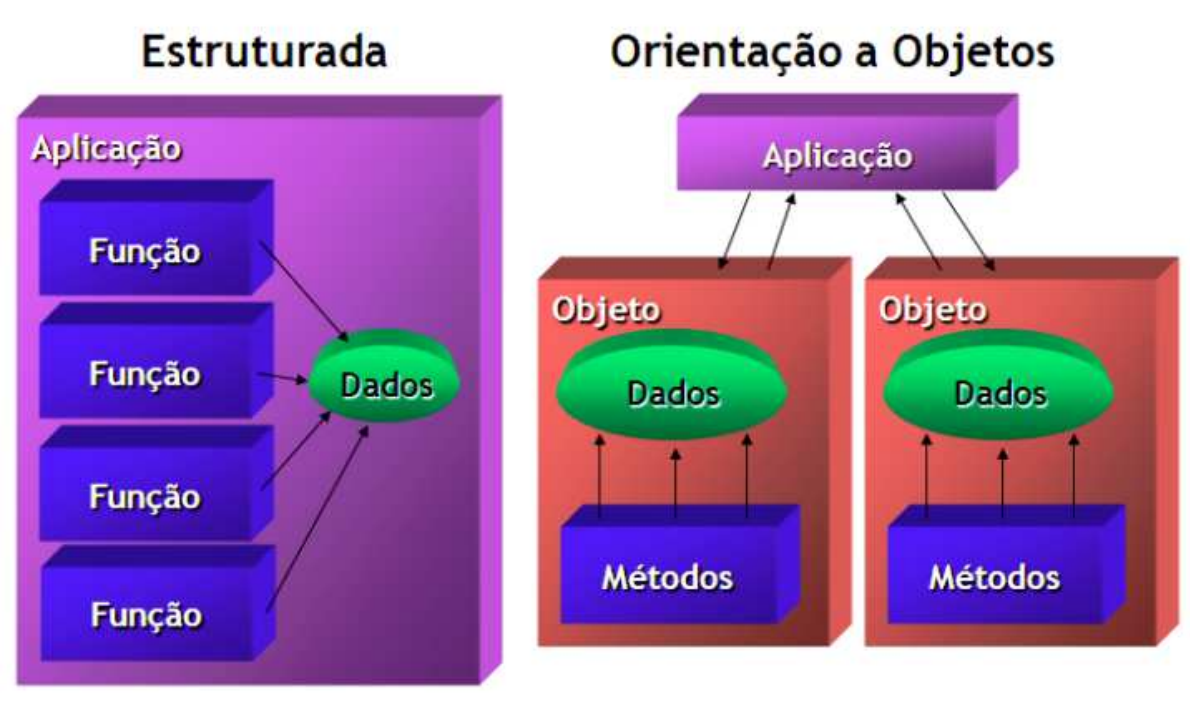
- **Base**
  - Classes e Objetos
  - Métodos e Atributos
- **Vantagens**
  - Melhor organização do código
  - Bom reaproveitamento de código
- **Desvantagens**
  - Desempenho mais baixo que o paradigma estruturado
  - Mais difícil compreensão

Programação Orientada a Objetos

- Um programa orientado a objetos é feito de objetos, com certas propriedades e operações que eles podem realizar
- A programação estruturada tradicional consiste na elaboração de um conjunto de procedimentos (ou algoritmos) para resolver um problema
- Depois disso se acha as formas apropriadas para armazenar e representar os dados (as estruturas de dados)

Programação Orientada a Objetos

- POO x Estruturada



Objetivos da POO

- Aumentar a produtividade do desenvolvimento de software
- Diminuir o custo de construção de software (principalmente em sistemas grandes)
- Facilitar a Manutenção (legibilidade do código, na facilidade de manutenção e expansão e na programação simplificada)
- Aumentar a compreensão do software
- Aproximar o desenvolvimento de software do mundo real (através da modelagem de objetos)

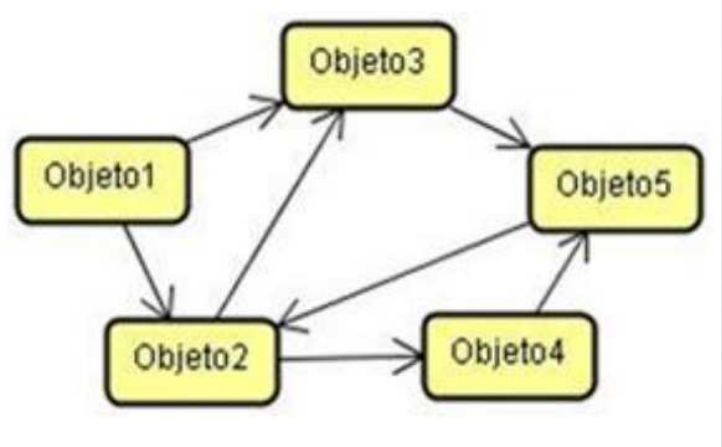
Origem e Histórico

- A ideia foi proposta por Kristen Nygaard e Ole-Johan Dahl (na Noruega) no início da década de 60 (1962 – Linguagem SIMULA)(Ambos ganharam o Turing Award de 2001)
- Na década de 1970, Alan Kay, um pesquisador do Xerox PARC, na Califórnia, criou a primeira linguagem totalmente orientada a Objetos (Smalltalk)
- Nos anos 70 surge Smalltalk, a primeira linguagem totalmente em Orientação a Objeto (O.O)
- C++, evolução de C, já possuía conceitos O.O
- Na década de 80 praticamente todas as linguagens já usavam conceitos O.O
  - Delphi
  - PASCAL
  - Java

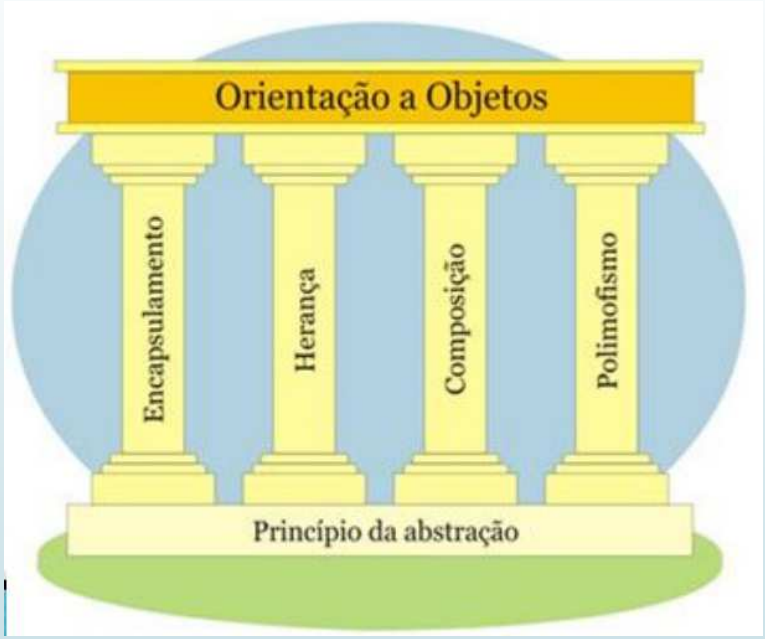
Conceito

- "Uma nova maneira de pensar os problemas utilizando conceitos do Mundo Real. O componente fundamental é o objeto que combina estrutura e comportamento em uma única entidade" - Raumbaugh
- "Sistema orientado a objetos é uma coleção de objetos que interagem entre si"- Bertrand Meyer

Conceito

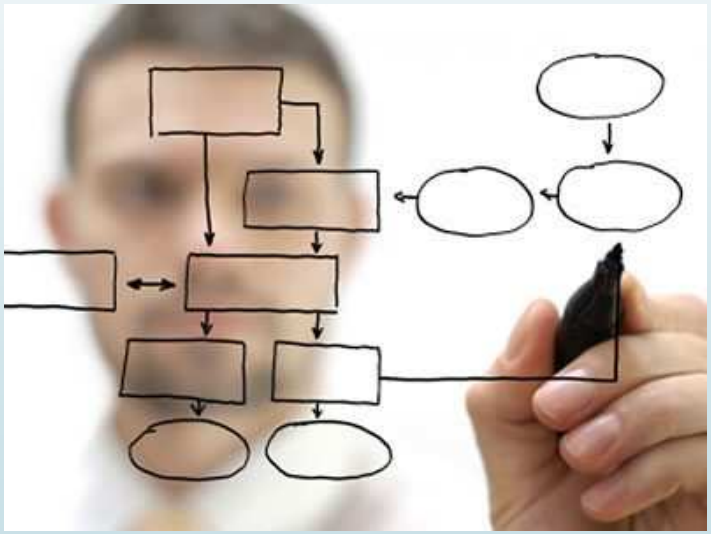


Conceito


Orientação a Objetos
Encapsulamento
Herança
Composição
Polimofismo
Princípio da abstração

Conceito

- "A Orientação a é um paradigma de análise, projeto e programação de sistemas de software baseado na composição e interação entre diversas unidades de software chamadas de *objetos*. Ou seja, é um modelo utilizado no desenvolvimento de software onde trabalhamos com unidades chamadas *objetos*



Conceito/Definição

- Não é apenas uma nova forma de programação
- Se preocupa com a modelagem (análise e projeto) dos processos/tarefas que devem ser realizados
- a Orientação a Objetos é uma nova forma de se pensar e representar de forma mais realista as necessidades dos softwares.

Abstração

- Habilidade de se concentrar nos aspectos essenciais do sistema, ou um contexto qualquer, ignorando o que é supérfluo dentro de um domínio
- Por natureza, as abstrações devem ser incompletas e imprecisas, mas isto não significa que ela perderá sua utilidade.
- nos permite, a partir de um contexto inicial, modelar necessidades específicas. Isso possibilita flexibilidade no processo de programação, pois é possível não trabalharmos com o conceito alvo diretamente, mas sim com suas abstrações

Exemplo

- Empresa de cadeiras
- Poderia pensar inicialmente em uma cadeira de forma mais básica (abstrata) possível
- Com isto, seu processo de produção seria facilitado, pois ela não saberia inicialmente quais os tipos de cadeiras que ela poderia fabricar, mas saberia que a cadeira teria, pelo menos, pernas, assento e encosto.
- A partir disto, ela poderia fabricar diversos tipos: cadeira de praia, cadeira de aula, cadeira digamos "moderna", entre vários outros tipos, a medida que novas demandas viessem a surgir.
- Neste caso, ele adaptaria sua linha de produção a partir de um modelo inicial

Exemplo

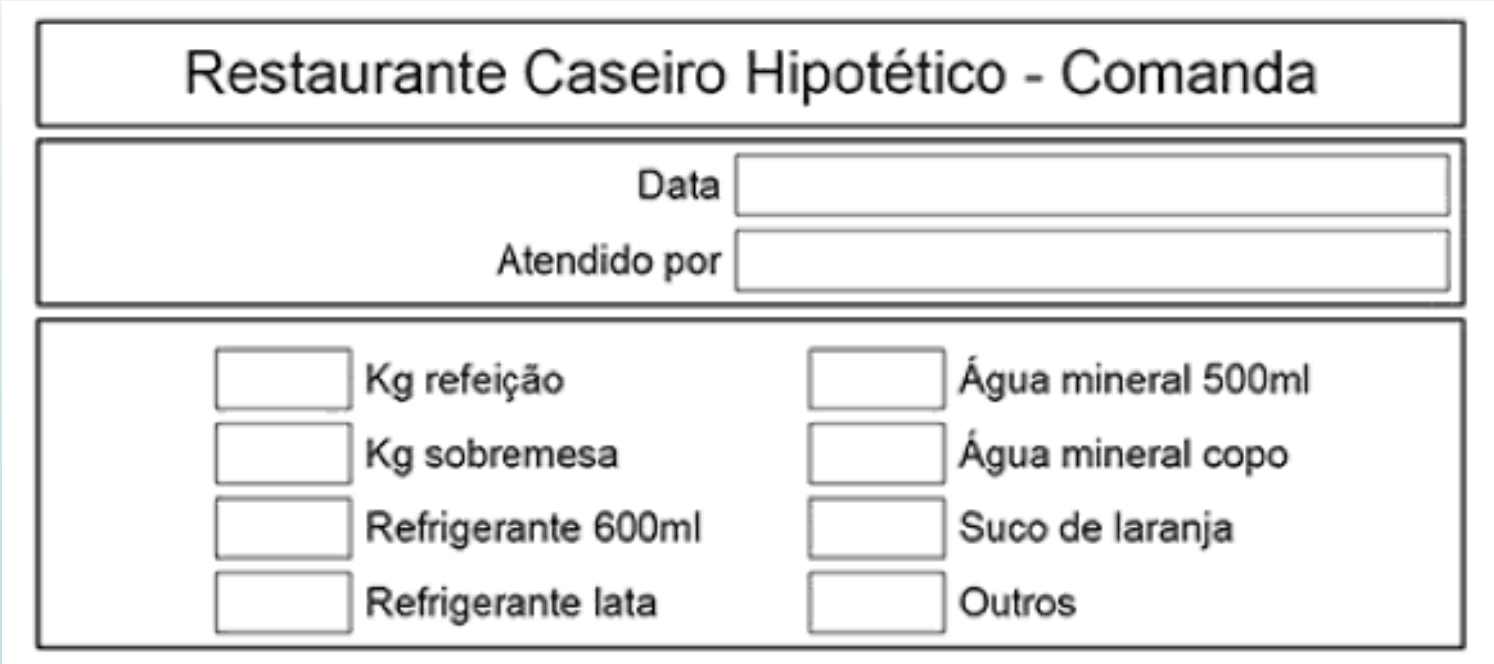
- Em cada tipo, algo poderia ser acrescentado ou modificado de acordo com sua especificidade.
- Assim, na cadeira de aula, teria um braço, já a de praia seria reclinável.
- Por fim, a "moderna" teria o assento acoplado ao encosto.
- Com isso, nota-se que, a partir de um modelo inicial adaptações podem ser realizadas para suprir as necessidades mais específicas

Generalização/Especialização

- Os processos de inicialmente se pensar no mais abstrato e posteriormente acrescentar ou se adaptar são também conhecidos como generalização e especialização (específico), respectivamente
- Exemplos:
  - carro → carro de corrida → sedã → pick-up
  - Pessoa → paciente → cliente → aluno
  - Cadeira → cadeira de balanço → cadeira do aluno (carteira)

Abstração a partir de modelos

- Modelos são representações simplificadas de objetos, pessoas, itens, tarefas, processos, conceitos e ideias
- São usados no dia a dia, independente do uso de computadores
- Vamos ver um exemplo, o Restaurante Caseiro Hipotético, que serve refeições por quilo



Abstração a partir de modelos

- O modelo do restaurante representa certos dados ou informações, que no caso são os itens e modelos quantidade dos pedidos por mesa.
- Como o modelo é uma simplificação do mundo real, os dados contidos no modelo são somente os relevantes à abstração do mundo real sendo feita.
- Por exemplo, para a anotação dos pedidos e cálculo da conta dos clientes do restaurante, alguns dados sobre o restaurante como endereço e data da inauguração são irrelevantes, e não devem ser representadas pelo modelo em questão.

Abstração a partir de modelos

- Um modelo comumente possui operações ou procedimentos associados a ele.
- Estas operações modelos são listas de comandos que processarão os dados contidos no próprio modelo (e em alguns casos, dados adicionais).
- Algumas operações que podemos fazer no modelo do Restaurante Caseiro Hipotético seriam a inclusão de um pedido para uma mesa, a modificação do status de um pedido de uma mesa (isto é, se o pedido foi servido ou não), o encerramento dos pedidos dos clientes de uma mesa e apresentação da conta para os clientes
- Também é possível a criação de modelos que contenham somente dados ou somente operações
- Modelos que contenham somente dados são pouco usados: normalmente os valores destes dados variam de uso para uso do modelo (ex. cada mesa do modelo do Restaurante Caseiro Hipotético pode ter quantidades de itens pedidos diferentes), portanto operações que modificam os valores dos dados serão necessárias

Abstração a partir de modelos

- Modelos que contenham somente operações podem ser considerados bibliotecas de operações
- Exemplos: grupos de funções matemáticas e de processamentos de dados
- As linguagens orientadas a objetos (como Java e C# contém muitas bibliotecas)

Abstração a partir de modelos

- Modelos podem conter sub-modelos e ser parte de outros modelos: O modelo RestauranteCaseiro poderia conter vários exemplares do modelo MesaDoRestaurante que representariam diferentes mesas do restaurante
- Se um modelo Data for criado para representar em conjunto dados sobre um dia, mês e ano, podemos usar este modelo dentro de outros modelos que usem uma data para representar, por exemplo, um nascimento ou evento
- A simplificação inerente aos modelos é em muitos casos, necessária
- Dependendo do contexto, algumas informações devem ser ocultas ou ignoradas.
- Por exemplo, a representação das informações sobre uma pessoa pode ser feita de maneira diferente dependendo do contexto, como nos três exemplos:
- Pessoa como empregado de uma empresa
- Pessoa como paciente de uma clínica médica
- Pessoa como contato comercial

Abstração a partir de modelos

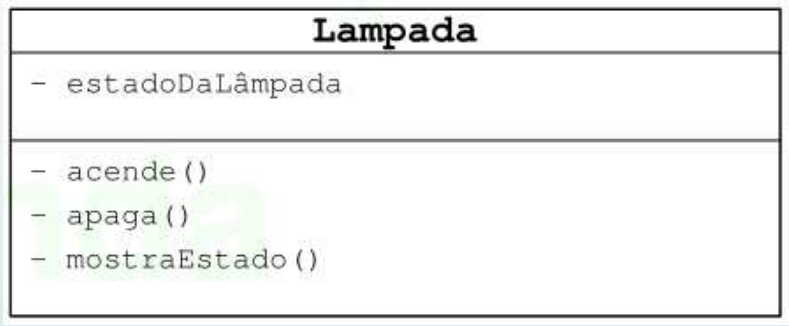
- As três maneiras de representarmos os dados de uma pessoa e operações nestes dados (ou os três modelos de representação de pessoas) são dependentes de contexto
- Alguns dados (atributos) e operações podem ser úteis para um modelo e irrelevantes em outro modelo
- Por exemplo, não faria sentido representar o salário de uma pessoa para fins de registros no banco de dados de pacientes de uma clínica ou a operação verificaObesidade de uma pessoa que não seja paciente de uma clínica médica.
- Por esta razão, e apesar dos três exemplos acima representarem pessoas de forma genérica, é difícil, se não impossível, elaborar um "super modelo" capaz de representar todos os dados e operações relativos a uma pessoa, independente do contexto

Abstração a partir de modelos

- Modelos podem ser reutilizados para representar diferentes objetos, pessoas ou itens: ou itens: o mesmo modelo PacienteDeClinica poderia ser utilizado para representar cada um dos pacientes de uma clínica
- Os pacientes podem ser representados pelo mesmo modelo, mas os dados individuais de cada um podem ser diferentes, criando a necessidade de diversos exemplos de cada modelo - assim João, Pedro e Maria seriam exemplos do modelo Paciente, cada um contendo dados de um paciente, provavelmente diferentes entre si
- Não é necessária a criação de um modelo para cada item, pessoa, ou objeto do mundo real
- A criação de modelos é uma tarefa natural e a extensão desta bordagem à programação deu origem ao paradigma Programação Orientada a Objetos.

Trabalhando com modelos

- Fazer alguns modelos para testar a ideia dos modelos
- Lâmpada incandescente

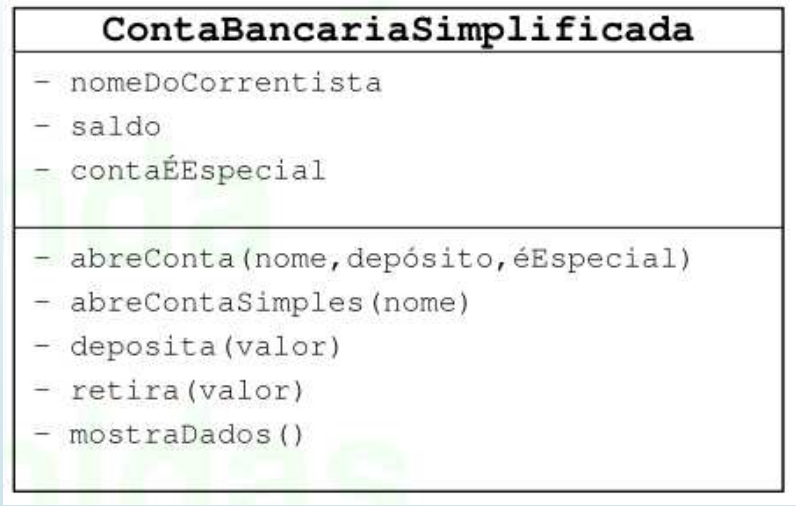


- A decisão de que atributos e métodos devem pertencer a um modelo depende da abrangência e escopo deste modelo
- Se as lâmpadas a serem representadas por este modelo fossem itens à venda em um supermercado, certamente os atributos seriam outros.
- O atributo estadoDaLâmpada e operações como acende(), apaga() e mostraEstado() não seriam usados

Modelo lâmpada em pseudocódigo

```
1  modelo Lampada // representa uma lâmpada em uso
2  início do modelo
3    dado estadoDaLâmpada; // indica se está ligada ou não
4
5    operação acende() // acende a lâmpada
6      início
7        estadoDaLâmpada = aceso;
8      fim
9
10   operação apaga() // apaga a lâmpada
11     início
12       estadoDaLâmpada = apagado;
13     fim
14
15   operação mostraEstado() // mostra o estado da lâmpada
16     início
17       se (estadoDaLâmpada == aceso)
18         imprime "A lâmpada está acesa";
19       senão
20         imprime "A lâmpada está apagada";
21     fim
22
23 fim do modelo
```

Uma conta bancária simplificada



Exercício

- Fazer o modelo ContaBancariaSimplificada em Pseudocódigo...

```
modelo ContaBancariaSimplificada
início do modelo
  dado nomeDoCorrentista,saldo,contaÉEspecial; // 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;
      contaÉEspecial = 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;
      contaÉEspecial = 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 (contaÉEspecial == 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

  operação mostraDados() // mostra os dados da conta, imprimindo os seus valores
    início
      imprime "O nome do correntista é ";
      imprime nomeDoCorrentista;
      imprime "O saldo é ";
      imprime saldo;
      se (contaÉEspecial) imprime "A conta é especial.";
      senão imprime "A conta é comum.";
    fim

fim do modelo
```

Um registro acadêmico

```
modelo RegistroAcademico
início do modelo
  // Dados do registro acadêmico
  dado nomeDoAluno,númeroDeMatrícula;
  dado dataDeNascimento,éBolsista,anoDeMatrícula;

  // Inicializa simultaneamente todos os dados do modelo, passando argumentos
  operação inicializaRegistro(oNome,aMatrícula,aData,temBolsa,qualAno)
    início
      // Usa os argumentos para inicializar os valores no modelo
      nomeDoAluno = oNome;
      númeroDeMatrícula = aMatrícula;
      dataDeNascimento = aData;
      éBolsista = temBolsa;
      anoDeMatrícula = 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 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
```

Perguntas e Discussão