Aula 2 – POO 1
Introdução

Profa. Elaine Faria

UFU - 2020

Sobre o Material

- Agradecimentos
  - Aos professores José Gustavo e Fabiano, por gentilmente terem cedido seus materiais.

- Os slides consistem de adaptações e modificações dos slides dos professores José Gustavo e Fabiano

Paradigma

- Maneira de abordar determinado problema segundo um conjunto de procedimentos, valores ou conceitos que direcionam o pensamento
- Visão sobre a estruturação e execução de um procedimento
- Técnicas, estruturas, conceitos utilizados para o desenvolvimento de determinado software

Introdução

- **Crise do software** → dificuldades e consequentes frustrações que o desenvolvimento e a manutenção de softwares trouxeram para as grandes empresas
  - Expressão inicialmente utilizada no fim da década de 60, e início da década de 70
- Ligação com complexidade do desenvolvimento de software e simultânea imaturidade do processo

Introdução

- Causa: constatações
  - Como nós desenvolvemos o software
  - Como nós mantemos o volume crescente de softwares existentes
  - Como nós mantemos o ritmo do desenvolvimento dos softwares, tendo em vista a crescente demanda
- Exemplo → *bug* do ano 2000
- Crise do software levou empresas a buscarem novas soluções para a obtenção de maiores lucros e menores custos
  - Orientação a objetos

Introdução

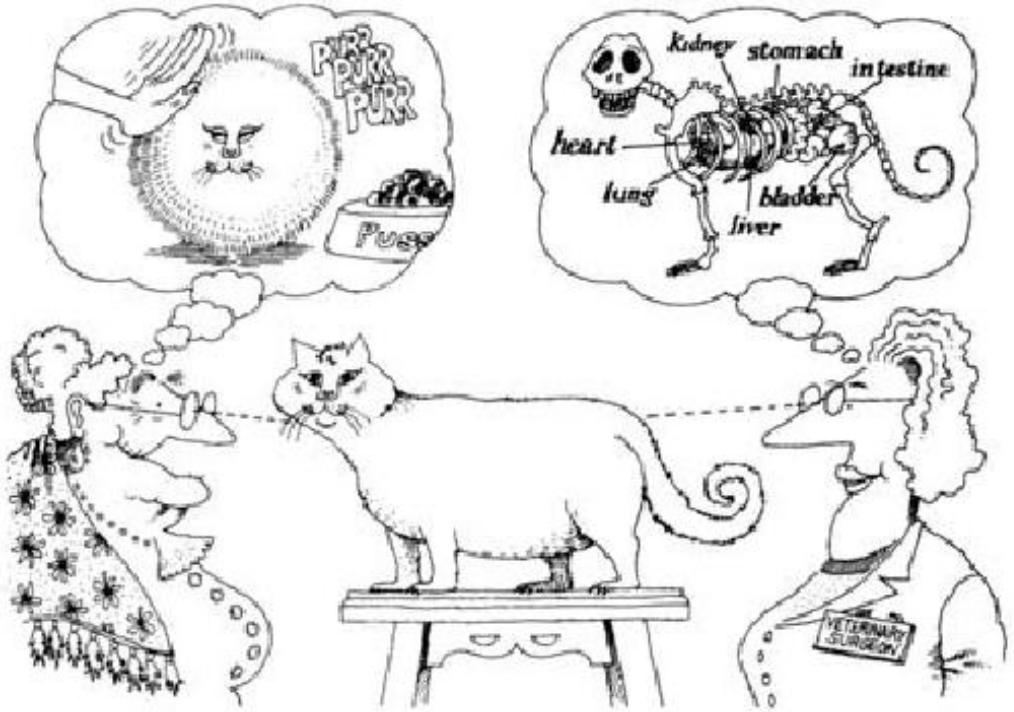
- Organizações começaram a adotar a Orientação a Objetos, mas com fanatismo
  - Alguns casos descartando totalmente as técnicas utilizadas anteriormente
- Orientação a Objetos não é mágica
  - “Um tolo com uma boa ferramenta continua a ser um tolo”
- Projetos com um mau gerenciamento e pouca competência por parte dos desenvolvedores transformam-se em softwares ruins

Evolução da O.O.

- Os modelos Orientados a Objetos evoluíram a partir da própria evolução das linguagens de programação
  - Primeiras linguagens de programação
    - Semelhantes à máquina
    - Linguagens imperativas
  - Programas extremamente dependentes de hardware, e de difícil manutenção
  - Idéia da O.O. → ao invés de programar pensando como a máquina, pode-se programar pensando como humanos

Abstração

- Extrair do domínio do problema os detalhes relevantes e representa-los não mais na linguagem do domínio, e sim na linguagem da solução
- Diferentes níveis de abstração



Evolução dos modelos de abstração

- Evolução dos modelos de abstração
  - Todas as linguagens de programação fornecem abstrações
    - Linguagem *Assembly* → pequena abstração da máquina
    - Linguagens imperativas (FORTRAN, BASIC e C) → abstrações da linguagem *Assembly*
    - Nessas linguagens, é necessário pensar em termos da estrutura do computador, ao invés de se pensar em termos da estrutura do problema

Evolução dos modelos de abstração

- Evolução dos modelos de abstração
  - Programação estruturada: implementação de módulos de programas desenvolvidos utilizando metodologias *top-down*
    - Objetivo: sistema como um conjunto de passos e chamadas a funções e procedimentos que atuam em dados (mas são distantes conceitualmente) para resolver determinado problema

Visão Estruturada

- Paradigma de programação procedimental:
  - Ausência de uma metodologia
  - Atividade de programar muito peculiar a cada programador

Gerando...

Uma insegurança na qualidade dos softwares produzidos.

Visão Estruturada

- Resolução de problemas por meio da definição de funções
  - Resultantes da decomposição do problema inicial em sub-problemas menores
- Este processo
  - É chamado de Decomposição Funcional
  - Identifica as principais funções ou passos de um processo
    - Podem ser decompostos em funções mais simples
  - Degrada o domínio do problema por não levar para a solução os conceitos envolvidos e suas relações.

Visão Estruturada

- Vantagem:
  - Aparantemente mais intuitivo e direto
- Desvantagens:
  - Descreve apenas a solução para um problema, e não o problema
  - Dificulta o entendimento da solução
  - Dificulta a manutenção do software
  - Dificulta a reutilização do software
  - Quanto maior o problema, mais difícil fica

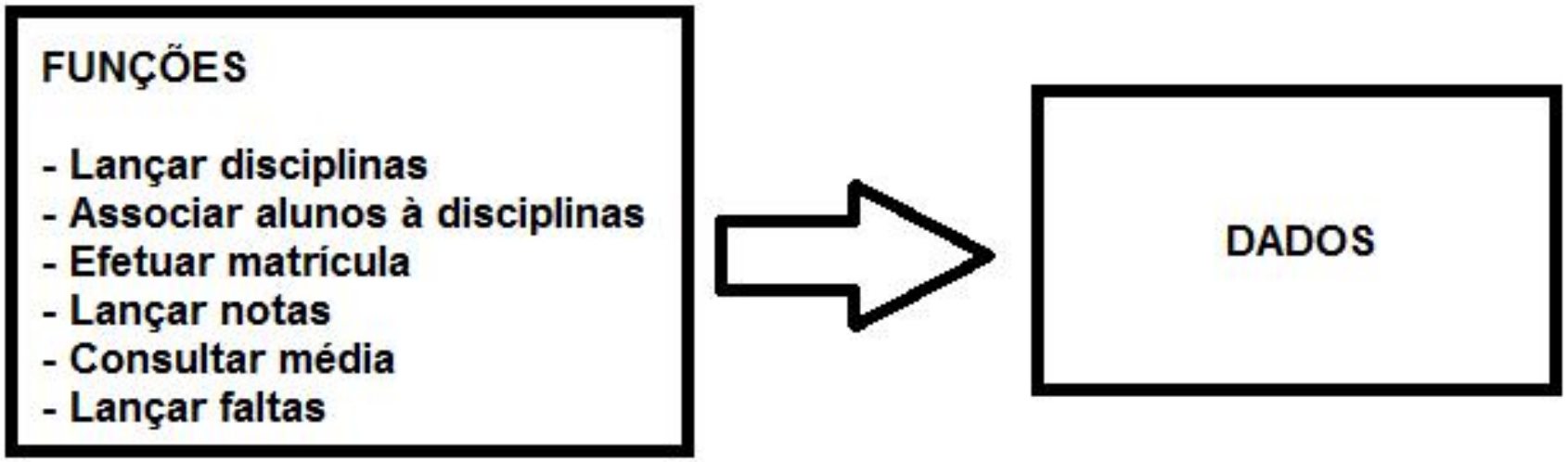
Visão Estruturada

- Dispersão de dados e funções, sem vínculo
- Dados podem ser lidos e modificados por qualquer função, desordenadamente
- Não existe um modelo de dados
- A construção de um modelo de dados posteriormente é mais confusa
- Dificuldade na depuração e correção

Visão Estruturada

- Decomposição do problema - Exemplo: gestão acadêmica
  - Funcionalidades
    - Lançar disciplinas
    - Associar alunos à disciplinas
    - Efetuar matrícula
    - Lançar notas
    - Consultar média
    - Lançar faltas

Visão Estruturada



Visão O.O.

- Tentar aproximar o mundo real e o mundo virtual
- Simular o mundo real dentro do computador
  - Utilizar objetos, afinal, nosso mundo é composto de objetos
- O programador é responsável por moldar o mundo dos objetos, e definir como os objetos devem interagir entre si.
  - Possibilita a criação de códigos com baixo acoplamento e que podem ser facilmente reutilizados → principais motivos para se programar orientado a objetos

Visão O.O.

- Trazer o espaço da solução para o espaço do problema
  - Ambos em uma única linha de pensamento
  - **Objeto**: pequena parte do problema, com informações específicas (dados), estados e operações a serem executadas
    - Semelhante ao comportamento dos objetos no mundo real, que também possuem características e comportamentos

Visão O.O.

- Exemplo de problema: gestão acadêmica
  - Componentes do problema
    - Disciplinas
    - Alunos
    - Matrículas
  - Comunicação entre componentes
    - Mensagens

Visão O.O.

- Vantagens
  - Mais intuitivo: mais próxima da maneira que os seres humanos utilizam para pensar
    - Bom para o cliente
    - Bom para o analista e desenvolvedor
  - Mais fácil de manter e atualizar
  - Mais coesa

Programa em Linguagem O.O.

- Definição
  - Conjunto de objetos dizendo uns para os outros o que fazer através do envio de mensagens
    - Pode-se pensar nas mensagens como sendo chamadas a funções (métodos) que pertencem a um objeto em particular
- Cada objeto
  - Tem a sua própria região de memória, que também pode ser composta por outros objetos
    - Exemplo: carro, aluno, livro, etc.
  - Tem um conjunto de operações que ele realiza, e que afetam seus prórios dados

Programa em Linguagem O.O.

- Cada objeto possui
  - um **tipo**: isto é, pertence a uma **classe**
- Cada objeto possui
  - Identidade
  - Estado
  - Comportamento
- Transferência de responsabilidade do procedimento para o próprio dado (objeto)

Programação Tradicional X O.O.

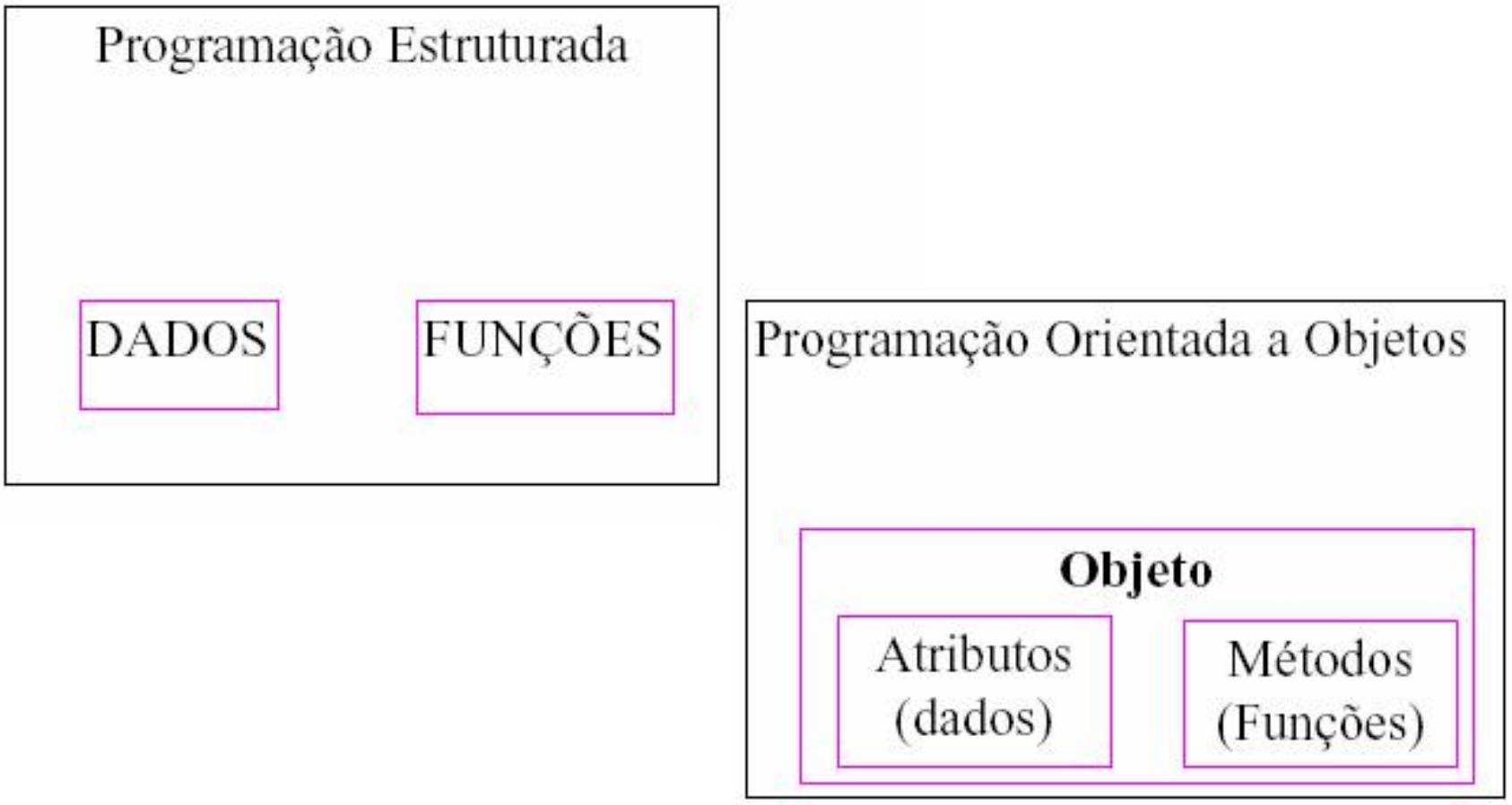
- Programação Tradicional
  - Sistema esperando que uma sequência de operações seja executada e apresente um resultado esperado

- Orientação a objeto
  - Sistema como um conjunto de objetos que possuem determinados comportamentos, e que interagem entre si

Programação Tradicional X O.O.

- Programação Tradicional
  - Dados e programa: entidades diferentes e separadas
  - Duas partes: dados e funções
- Orientação a Objeto
  - Dados e procedimentos: parte de um só elemento básico (objeto)
  - Duas partes (dados e funções) PARA CADA OBJETO

Programação Tradicional X O.O.



Programação Tradicional X O.O.

Vantagens das técnicas O.O.

- Código Enxuto
  - Sistemas O.O. não precisam pensar em termos de *loops* ou desvios, mas em eventos que alteram o estado dos objetos
  - Alterações no estado dos objetos necessitam de pequenos pedaços de código
  - Objetos com pequenos pedaços de código resultam em uma probabilidade muito menor de erros e *bugs*

Vantagens das técnicas O.O.

- Caixa Preta
  - Depois que os objetos são construídos (e testados, eliminando todos os possíveis erros), transformam-se em caixas pretas, e os projetistas não precisam se preocupar mais com “como” os objetos realizam suas operações
- Maneira Natural de Pensar
  - A maneira O.O. de pensar é mais natural para a maioria das pessoas do que as técnicas de análise e projetos estruturados

Vantagens das técnicas O.O.

- Dados e funcionalidade encapsuladas
  - Na programação convencional, os dados podem assumir qualquer estrutura e os processos podem fazer qualquer coisa aos dados
  - No mundo orientado a objetos, as estruturas de dados relacionam-se aos objetos e é possível criar restrições de acesso
  - Com isso, essas estruturas de dados podem ser usadas somente com os métodos projetados para esse tipo de objeto

Vantagens das técnicas O.O.

- Independência
  - Cada objeto executa uma função específica independentemente de outros objetos
  - O objeto responde a mensagens, sem saber por que a mensagem foi enviada ou quais serão as consequências de suas ações
  - Consequentemente, partes do sistema podem ser mudadas de uma forma amplamente independente de outras

Vantagens das técnicas O.O.

- Reusabilidade
  - Sistemas frequentemente podem ser construídos de objetos existentes
  - O repositório deve conter uma biblioteca sempre crescente de tipos de objetos, alguns comprados e outros construídos dentro da própria organização
  - A recriação de sistemas que funcionem de forma correta é mais fácil com as técnicas O.O.

Criação de um Sistema O.O.

- Modelagem Conceitual
  - Análise dos objetos do mundo real envolvidos no problema e suas relações
- Representação da visão que o usuário tem das informacões gerenciadas pelo sistema
  - Descobrir os conceitos que compõem o domínio do problema
  - Definir os elementos básicos
    - Classes
    - Atributos
    - Associações

Referências

- DEITEL, H. M., DEITEL, P. J., **Java: Como Programar,** Bookman, São Paulo, 2002
- DORÇA, F., **Notas de Aula de Programação Orientada a Objetos**, disponível em http://www.facom.ufu.br/~fabiano
- PAIVA, J. G. S., **Notas de Aula de Programação Orientado a Objetos**