Pilares da Programação Orientada a Objetos

BCC 221 - Programação Orientada a Objectos(POO)

Guillermo Cámara-Chávez

Departamento de Computação - UFOP

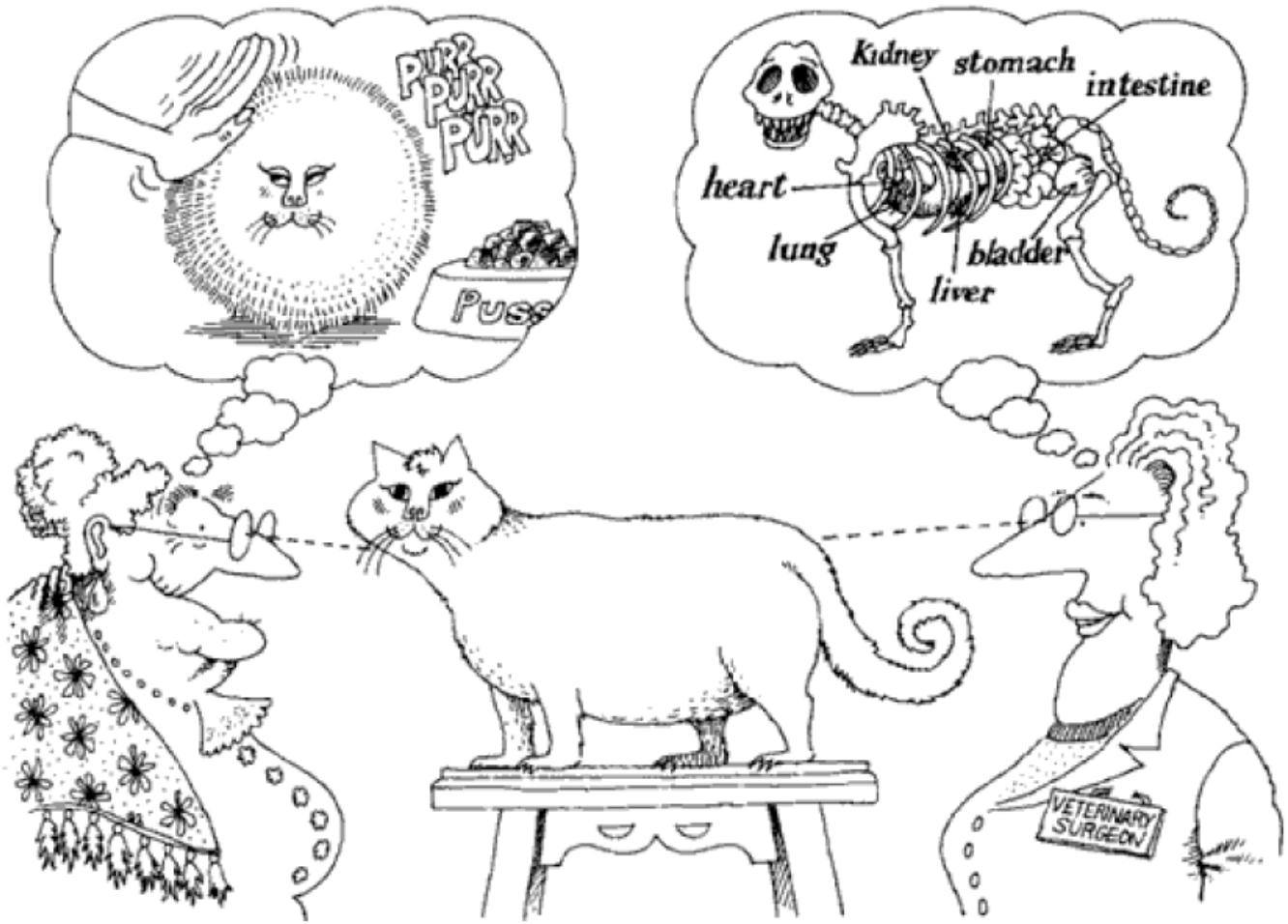
Fundamentos do Modelo de Objetos I

- Abstração de dados
- Encapsulamento
- Modularidade
- 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



UFOP

decom
departamento de computação

Abstração de conceitos do Domínio III

- 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 IV

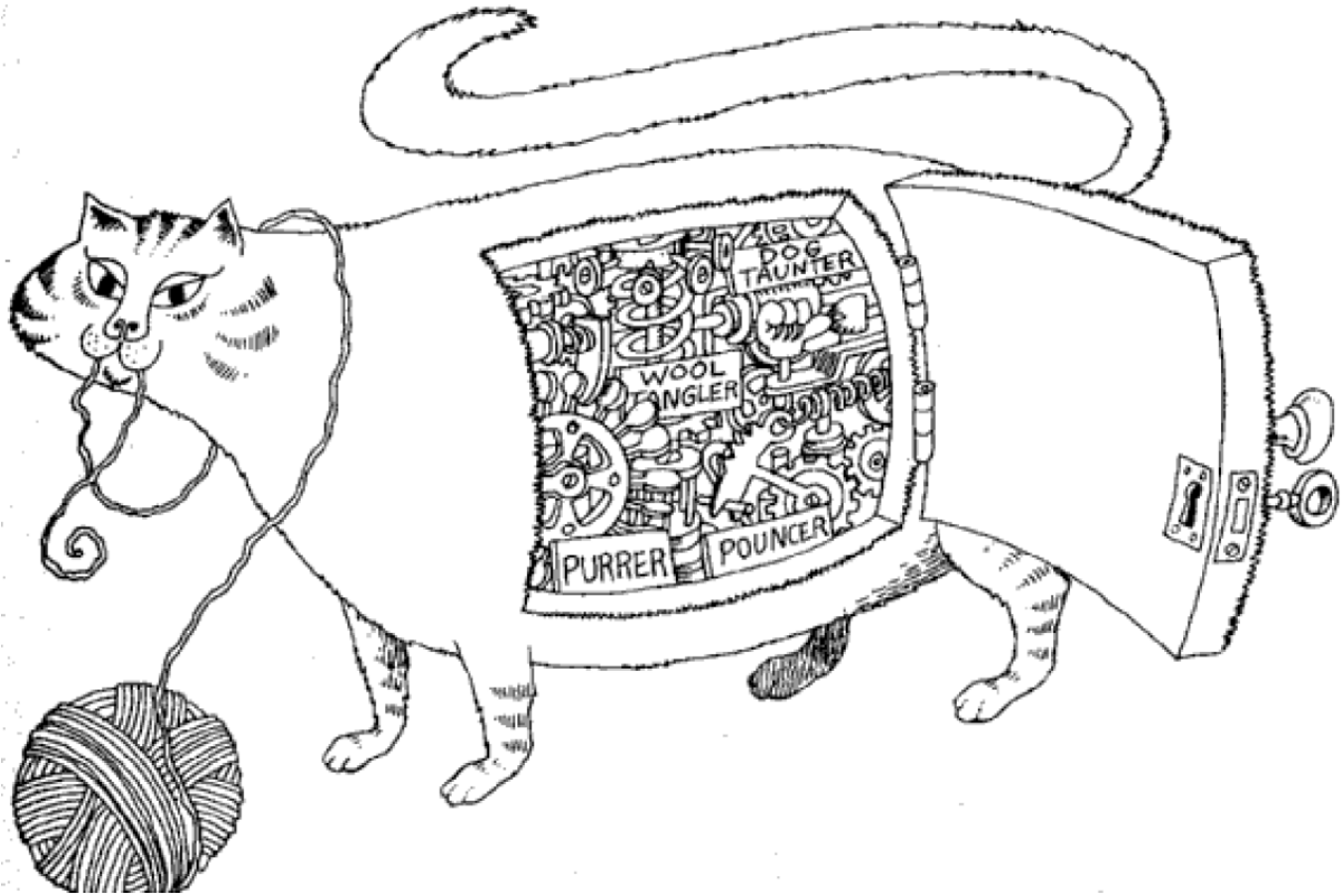
- **Engenheiro**: Qual a Marca? Qual a capacidade? Qual a Velocidade? Indicado para que tipo de edifício? Qual a largura? Qual a Altura?

- **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 ideia 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



UFOP

decom
departamento
de computação

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

- 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

UFOP

decom
departamento de computação

Modularidade III

- 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 IV

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.

Modularidade V

- 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 VI

- 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

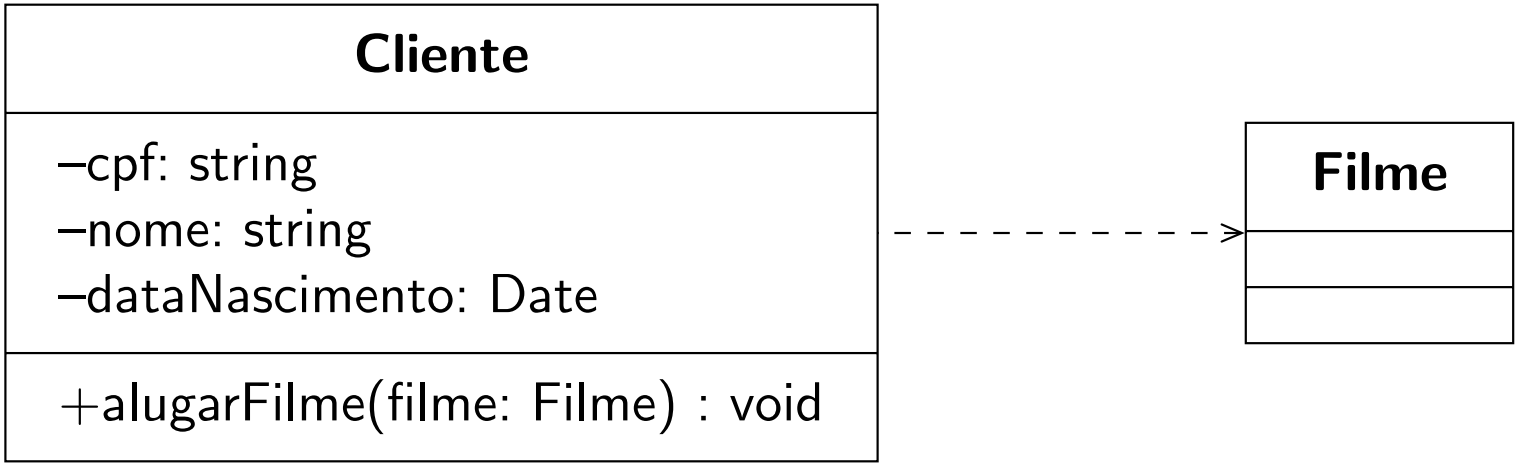
Ideal: Alta coesão e Baixo acoplamento

Modularidade VII

ContaCorrente
–numero: string –saldo: double
+sacar(valor: double) : void +depositar(valor: double) : void

Alta coesão

Modularidade VIII



Baixa coesão

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

UFOP

decom
departamento
de computação

FIM