



**INSTITUTO FEDERAL
DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA**
Rio de Janeiro

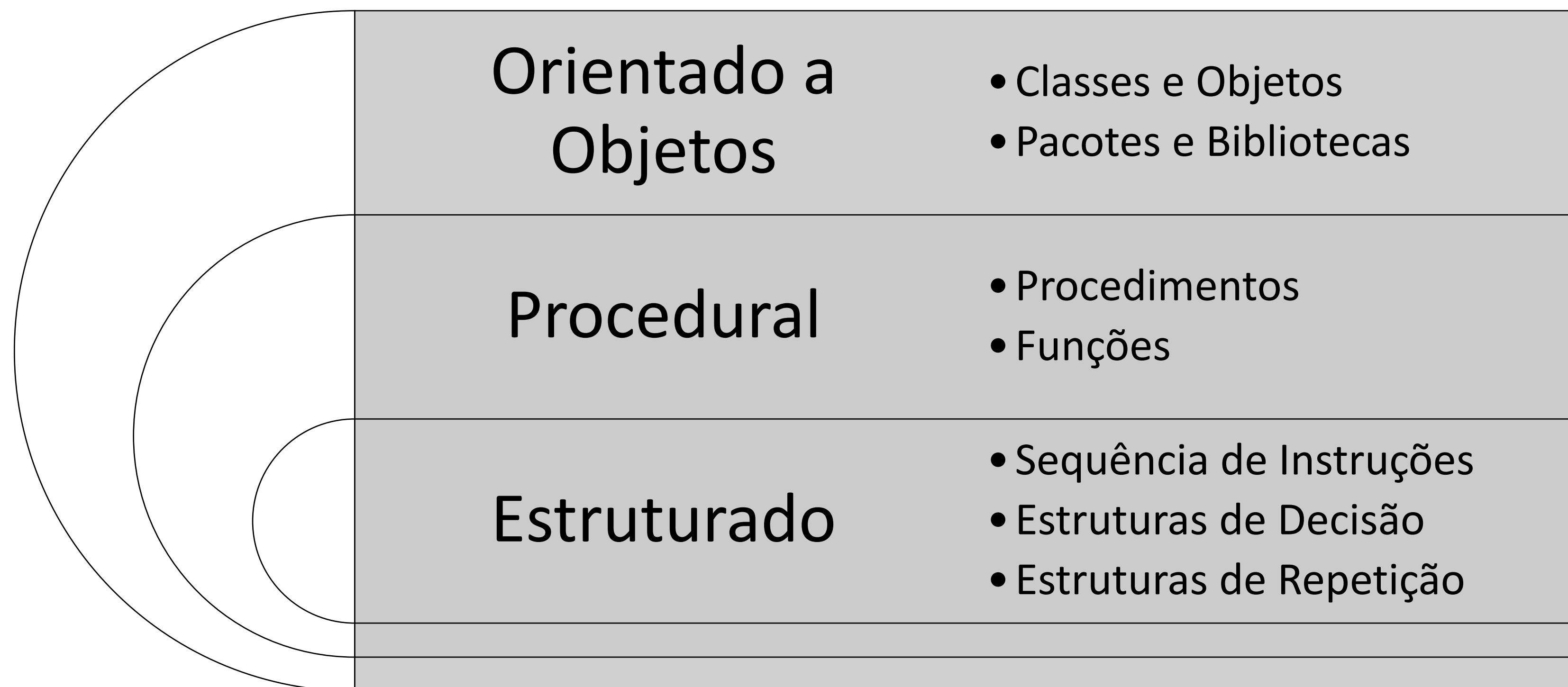
Disciplina

Modelagem de Domínio e
Conceitos de Orientação a Objetos

PROF. FERNANDO TAMBERLINI ALVES

Paradigmas de Programação

- ✓ A orientação a objetos pode ser vista como um passo natural na evolução dos paradigmas



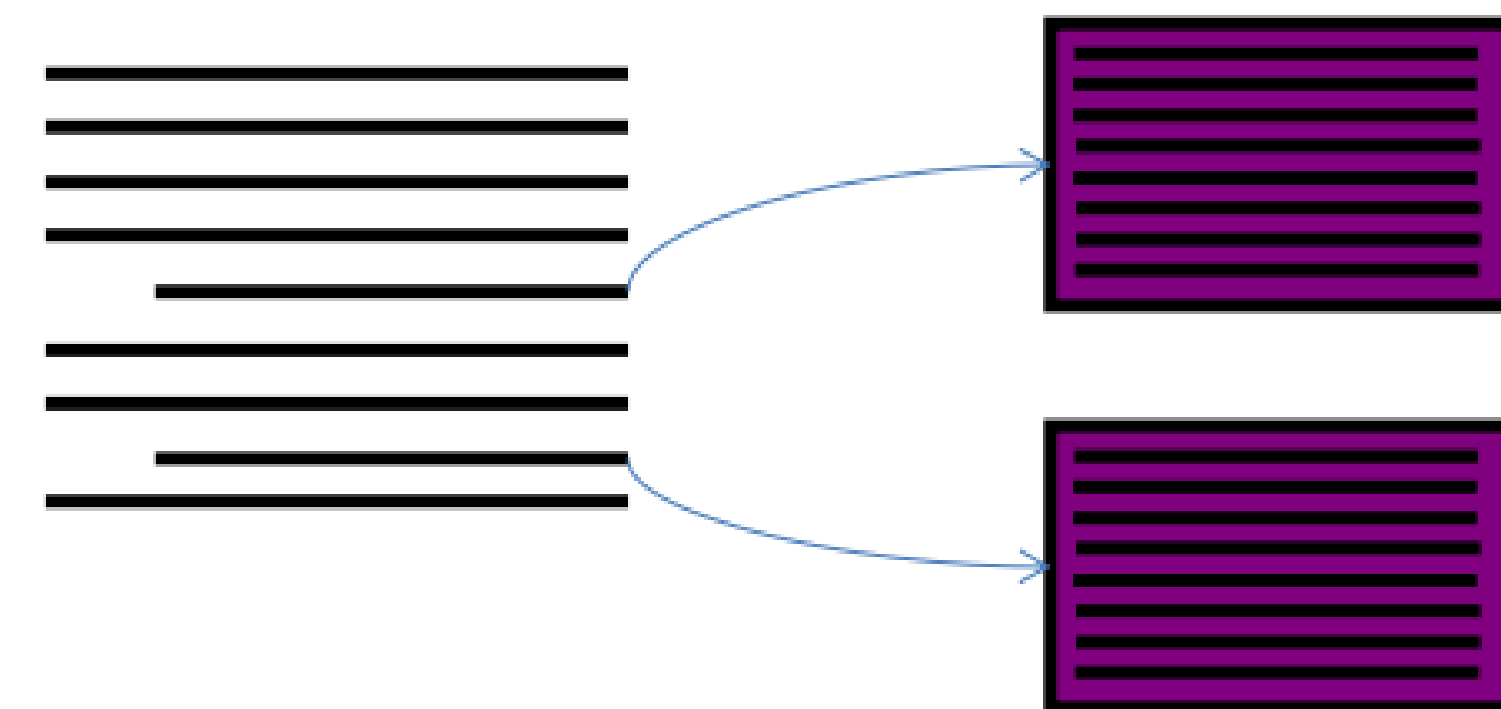
Paradigma Estruturado

- ✓ Só usa sequência, decisão e repetição
- ✓ Código mais fácil de ler, mas ainda difícil para sistemas grandes devido a repetição de código
- ✓ O que fazer se for necessário repetir uma sequência de linhas de código em diferentes locais?



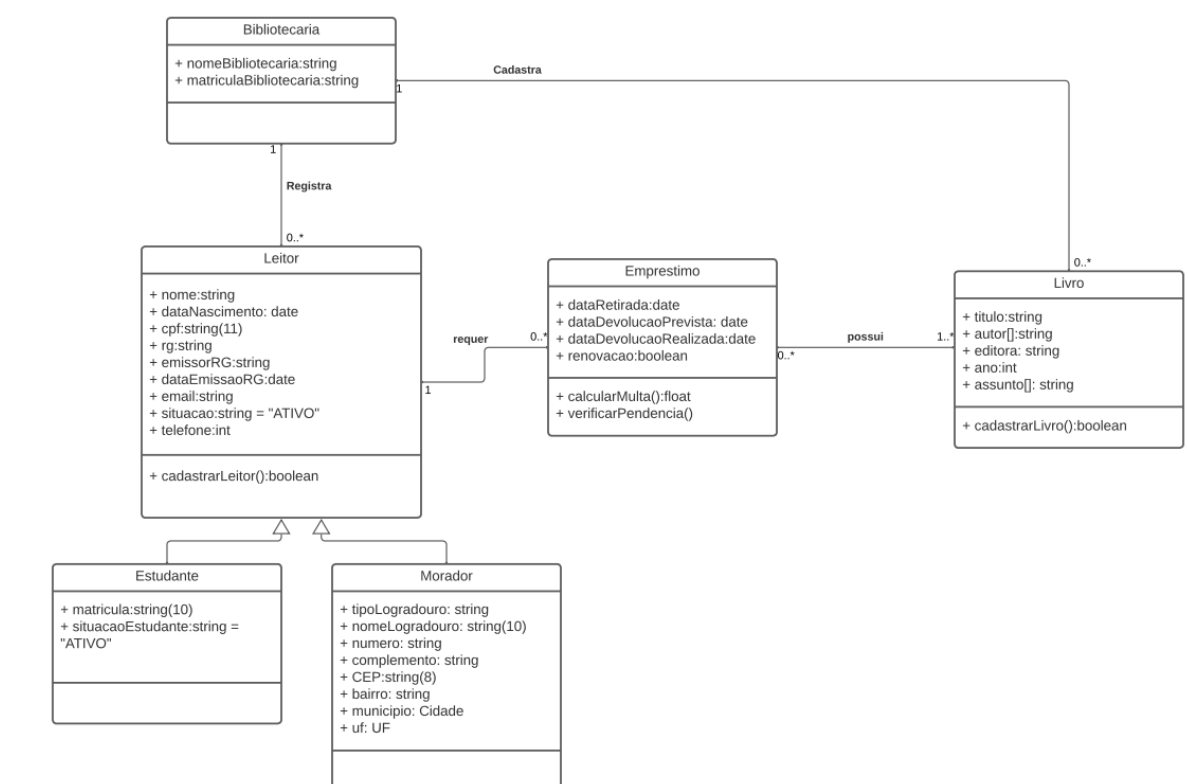
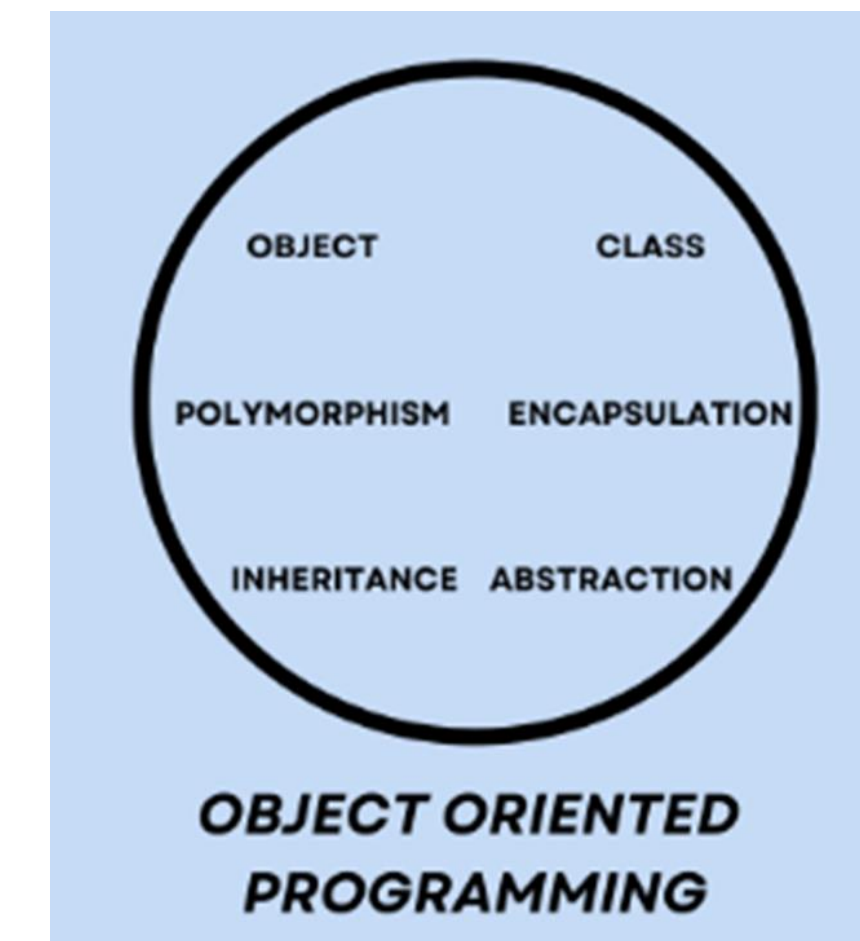
Paradigma Procedural

- ✓ Sinônimo: paradigma procedimental
- ✓ Uso de subprogramação
 - Agrupamento de código permitindo a criação de ações complexas
 - Atribuição de um nome para essas ações complexas
 - Chamada a essas ações complexas de qualquer ponto do programa
- ✓ Essas ações complexas são denominadas procedimentos, sub-rotinas ou funções



Paradigma Orientação a Objetos

- ✓ Classes e Objetos
 - Agrupamento de atributos e métodos
- ✓ Pacotes de Classes
 - Agrupamento de classes afins
 - Representam bibliotecas de apoio
- ✓ Facilitam o reuso e reaproveitamento de código
- ✓ Facilitam alta coesão e baixo acoplamento



P. Procedural x POO

Paradigma Procedural

Organiza o programa em termos de algoritmos

Paradigma Orientado a Objetos

Organiza o programa em termos de objetos



- ✓ Podemos criar programa pensando em termos de objetos ao invés de algoritmos?
- ✓ O mundo é composto de objetos
 - Uma loja tem produtos, pedidos, estoque, etc.
 - Um restaurante tem mesas, garçons, comidas, bebidas, etc.
 - Uma universidade tem professores, alunos, disciplinas, etc.
 - Uma rodoviária tem ônibus, passageiros, bagagens, etc.
- ✓ E se criarmos programas basicamente criando objetos equivalentes ao mundo real, e fazendo com que esses objetos se comuniquem?

Exemplo: Agenda Eletrônica

Paradigma Procedimental

- Variáveis
 - Vetor de nomes
 - Vetor de endereços
 - Vetor de telefones
- Procedimentos
 - Listagem de todos os nomes
 - Listagem do endereço dado um nome
 - Listagem do telefone dado um nome
 - Adição de nome, endereço e telefone
 - Remoção de nome, endereço e telefone

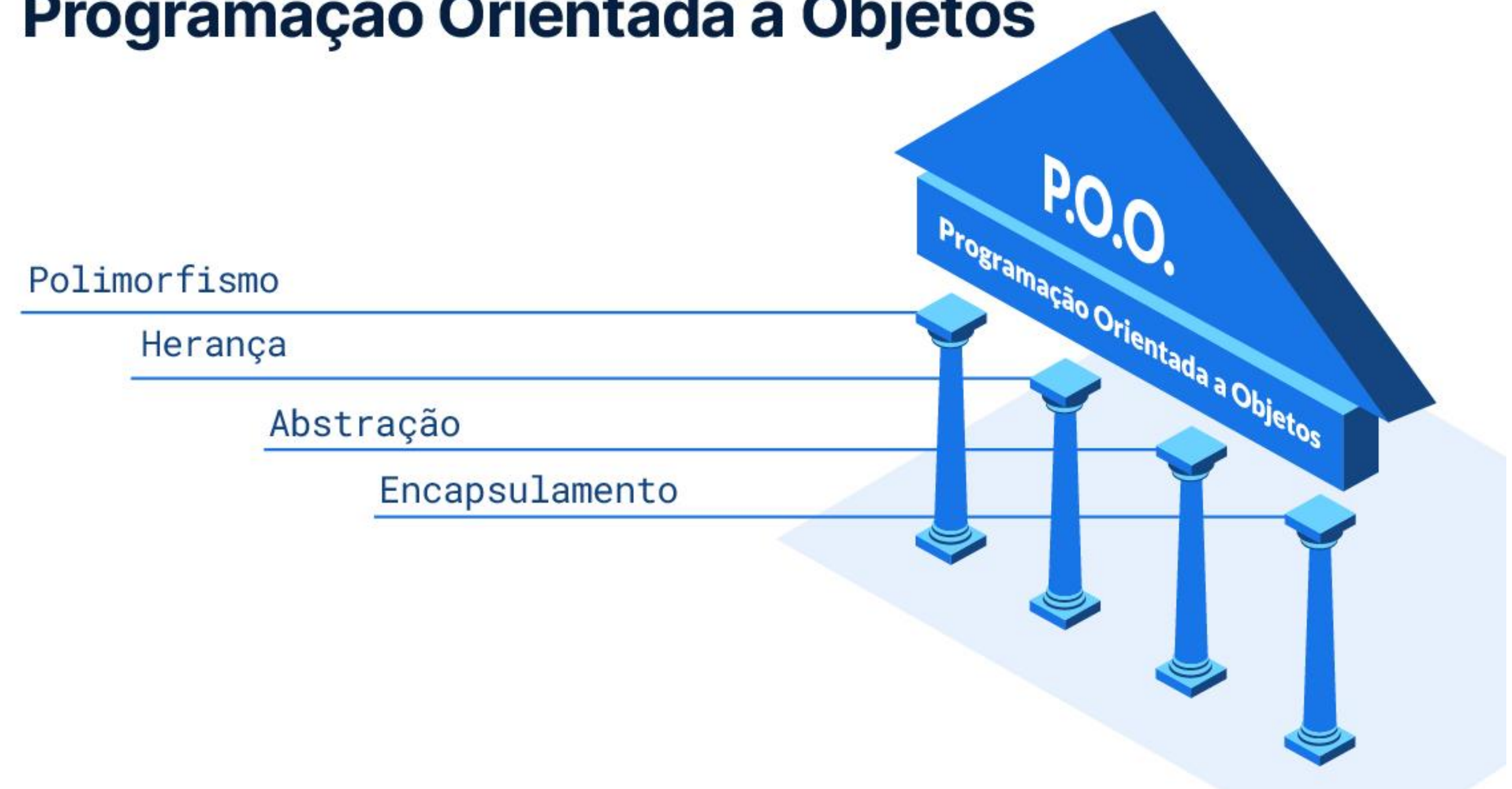
Paradigma OO

- Objeto Agenda
 - Atributo : Vetor de Contatos
 - Métodos
 - Listagem de Contatos
 - Adição de um Contato
 - Remoção de um Contato
- Objeto Contato
 - Atributos: Nome, Endereço Telefone
 - Métodos
 - Exibição de nome, endereço e telefone
 - Edição de nome, endereço e telefone

Paradigma Orientação a Objetos

- ✓ Abstração
- ✓ Encapsulamento
- ✓ Herança
- ✓ Polimorfismo

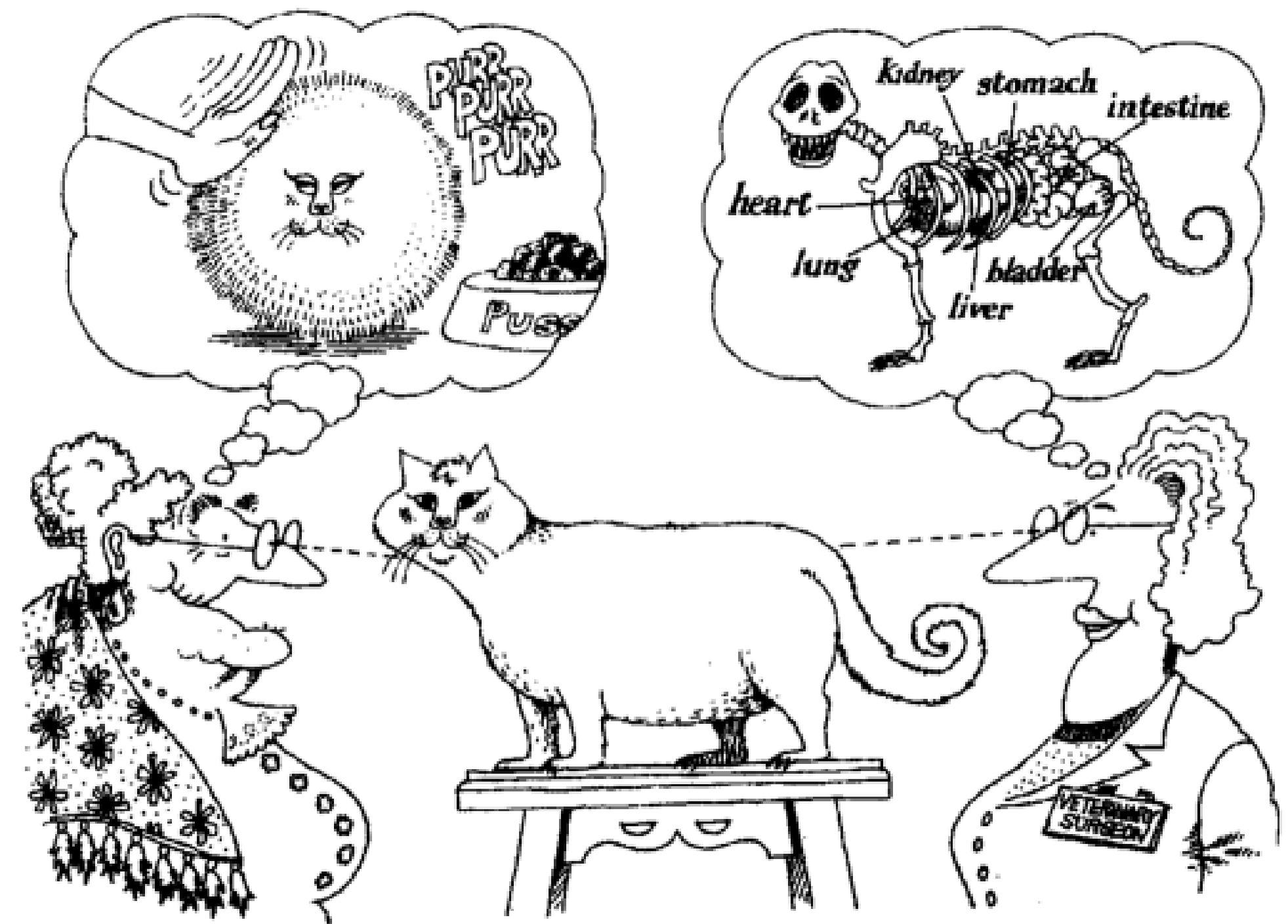
Programação Orientada a Objetos



Paradigma Orientação a Objetos

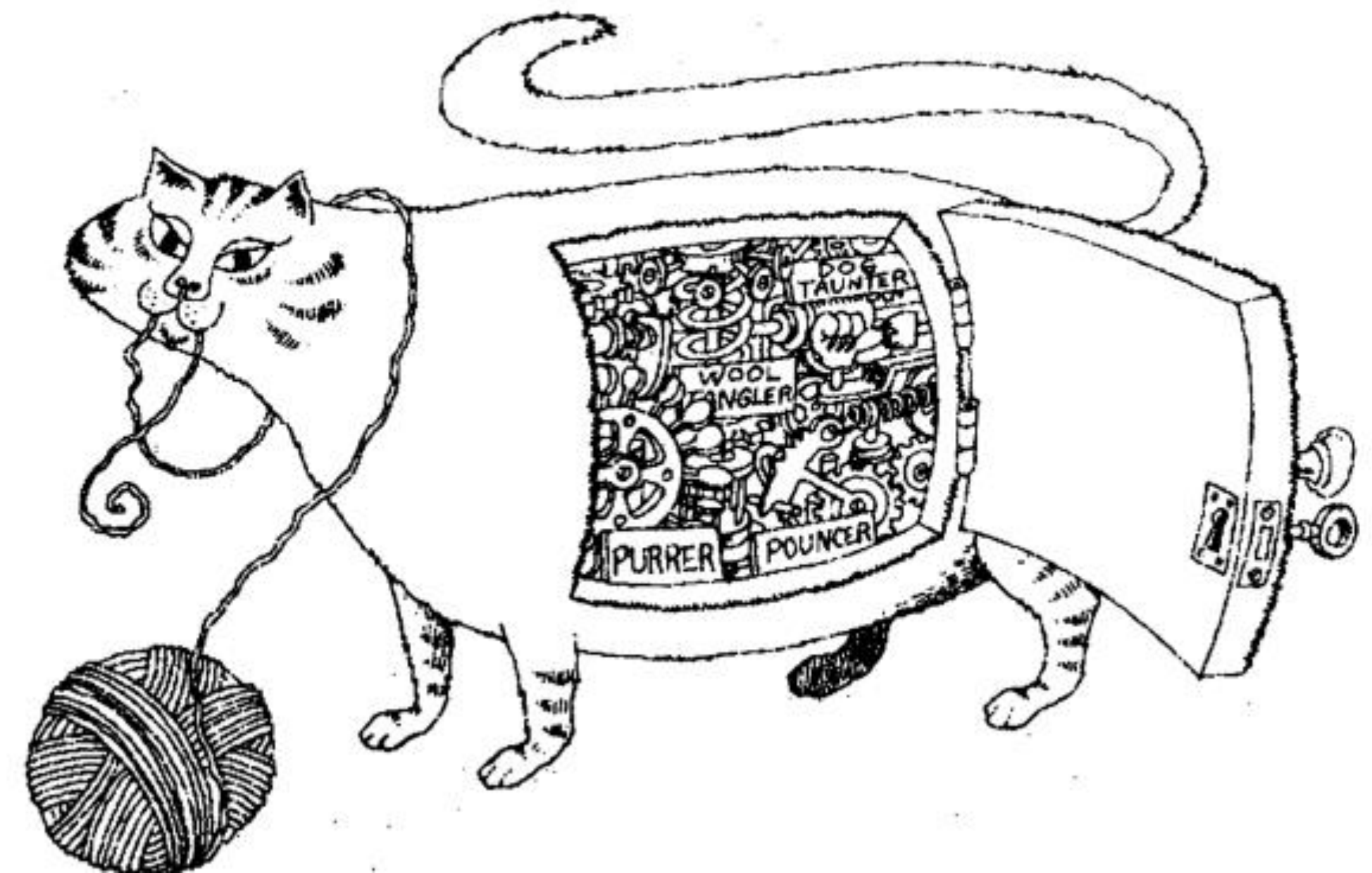
✓ Abstração

- É o princípio de ocultar detalhes complexos de implementação, expondo apenas uma interface simplificada e relevante para o usuário, permitindo que ele interaja com objetos sem se preocupar com suas complexidades internas



✓ Encapsulamento

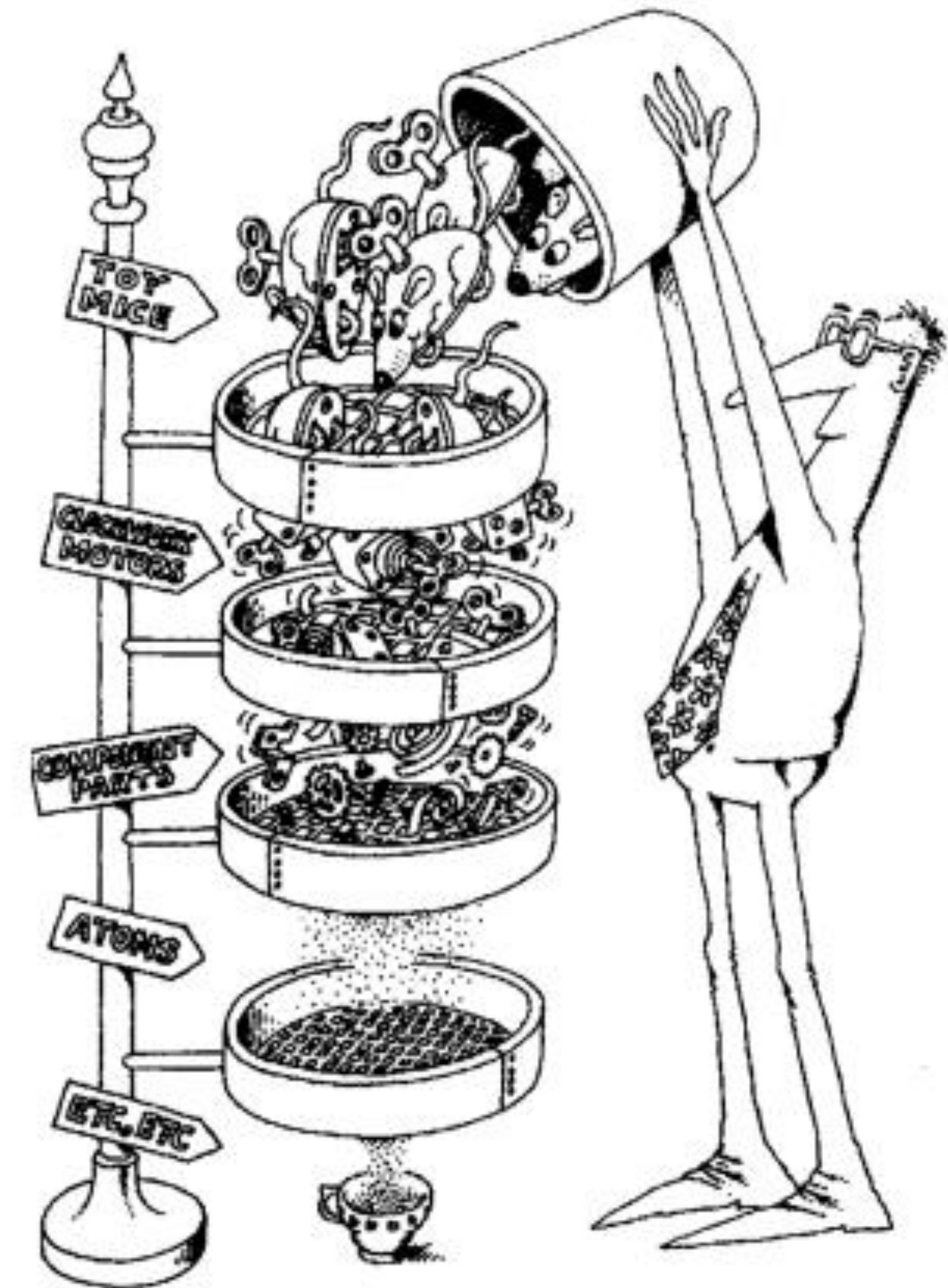
- É o princípio de restringir o acesso direto aos dados de um objeto, permitindo que eles sejam manipulados apenas por meio de métodos definidos, protegendo a integridade do estado interno do objeto.



Paradigma Orientação a Objetos

✓ Herança

- É o mecanismo pelo qual uma classe (subclasse) pode herdar atributos e métodos de outra classe (superclasse), permitindo reutilização de código e estabelecimento de hierarquias entre classes.



Paradigma Orientação a Objetos

✓ Polimorfismo

O polimorfismo é a capacidade de objetos de diferentes classes responderem ao mesmo método de formas distintas. Isso permite tratar objetos de classes diferentes de maneira uniforme, promovendo flexibilidade e extensibilidade no código.



Trabalhando com objetos

- Como criar um objeto?
- Como acessar um atributo?
- Como executar um método?

```
1  ✓ const nomeObjeto = {  
2      atributo1 : 80,  
3      atributo2 : "blue",  
4  ✓      metodo1 : function() {  
5          console.log("Olá Mundo !");  
6          console.log("Minha cor é " + this.atributo2);  
7      }  
8  }  
9  nomeObjeto.atributo1;  
10 nomeObjeto.metodo1()
```

Trabalhando com Classes

- Como definir uma classe?
- Como instanciar um objeto a partir de uma classe?

```
class NomeClasse{  
    constructor(p1,p2){  
        this.atributo1 = p1;  
        this.atributo2 = p2;  
    }  
    metodo1(){  
        return this.atributo1*this.atributo2/2;  
    }  
}  
  
const nomeObjeto = new NomeClasse(5,8);
```

- Retornado a codificação do jogo
 - Criados os arquivos .html e .js
 - Codificado os arquivos index.html e canvas.js
 - Codificado os objetos tela, placar, cobra e jogo
 - A última versão já era possível movimentar a cobra a partir do teclado
- Vamos codificar tudo novamente ou iniciar a partir do que foi salvo no GitHub?

8º Passo – Codificando alimento

cobra > JS alimento.js > ...

```
1  class Alimento {
2      arqImagem='alimento.png';
3      constructor(valor,tamanho){
4          this.tamanho = tamanho;
5          this.valor = valor;
6          this.x = Math.round(this.tamanho+(tela.largura-2*this.tamanho)*Math.random())
7          this.y = Math.round(this.tamanho+placar.altura+(tela.altura-2*this.tamanho)*Math.random())
8          this.imagem = new Image();
9          this.imagem.src = this.arqImagem;
10     }
11     desenhar(){
12         canvasCtx.drawImage(this.imagem, this.x, this.y, this.tamanho, this.tamanho)
13     }
14 }
```

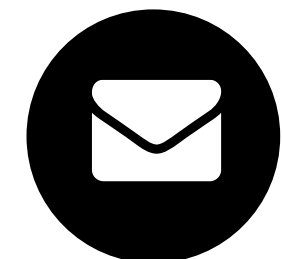
E depois? O que falta?

- Todos os objetos foram criados?
- E agora o que fazer?
 - Implementar os métodos faltantes de forma iterativa e incremental

Dúvidas?



Dúvidas?



fernando.alves@ifrj.edu.br



ftamberlini.dev.br

Referências:

- Notas de Aula do Prof. Leonardo Murta – UFF
- Livro “Object-Oriented Analysis and Design with Applications”