

# Introdução à Programação Orientada a Objetos

Prof. Marcelo Cohen

## 1. Contextualizando o problema

- O que leva um programador a mudar do paradigma procedimental para um novo ?
- A resposta esta na complexidade crescente dos sistemas e nas limitações da capacidade humana de compreensão de um sistema como um todo.
- Sistema complexo = conjunto grande e diverso de comportamentos tendo um longo ciclo de vida e muitos usuários dependendo dele.
- A complexidade está na quantidade e diversidade.

## 2. Orientação a Objetos

- Paradigma

“Paradigma é um conjunto de regras que estabelecem fronteiras e descrevem como resolver os problemas dentro destas fronteiras.

Os paradigmas influenciam nossa percepção; ajudam-nos a organizar e a coordenar a maneira como olhamos para o mundo...”

*Reengenharia - Reestruturando a Empresa  
Daniel Morris e Joel Brandon*

## Como assim ???

- Em Ciência da Computação:

– Paradigmas explicam como os elementos que compõem um programa são organizados e como interagem entre si.

## Orientação a Objetos

- Conceito
  - No mundo real, tudo é objeto!
  - Os objetos se relacionam entre si de diversas maneiras

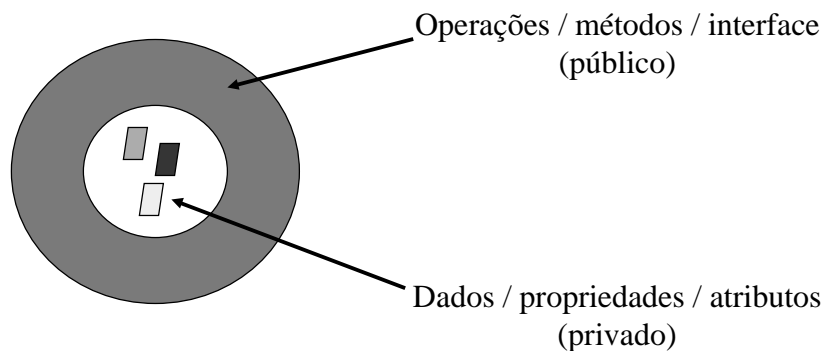
## Orientação a Objetos

- Conceito (2)
  - Um programa orientado a objetos é estruturado como uma comunidade de agentes que interagem entre si, denominados **objetos**.
  - Cada objeto tem um papel a cumprir
  - Cada objeto oferece um serviço ou realiza uma ação que é usada por outros membros da comunidade

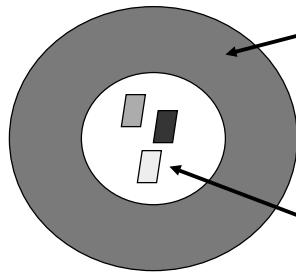
## Exemplo real: montagem de um computador

- Composto por vários componentes:
  - placa-mãe
  - CPU
  - placa de vídeo
  - disco rígido
  - teclado, etc.
- Cada componente é bastante sofisticado, mas o usuário não precisa saber como funciona internamente.
- Cada componente é independente dos demais.
- Para quem está montando, interessa apenas como os componentes interagem entre si:
  - a placa de vídeo encaixa no slot ?
  - O monitor funciona com essa placa ?
  - A CPU é compatível com a placa-mãe ?

## Objeto - representação



## Exemplo: Lâmpada



Operações:

- ligar
- desligar



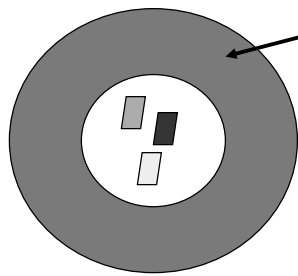
Dados:

- ligada (s/n)
- potência
- voltagem

## Encapsulamento (data hiding)

- É definido como uma técnica para minimizar as interdependências entre módulos, através da definição de interfaces externas.
- “Caixa preta” - não é necessário saber como funciona internamente, mas sim como utilizar.

# Encapsulamento



A interface (pública) de um objeto declara todas as operações permitidas (**métodos**)

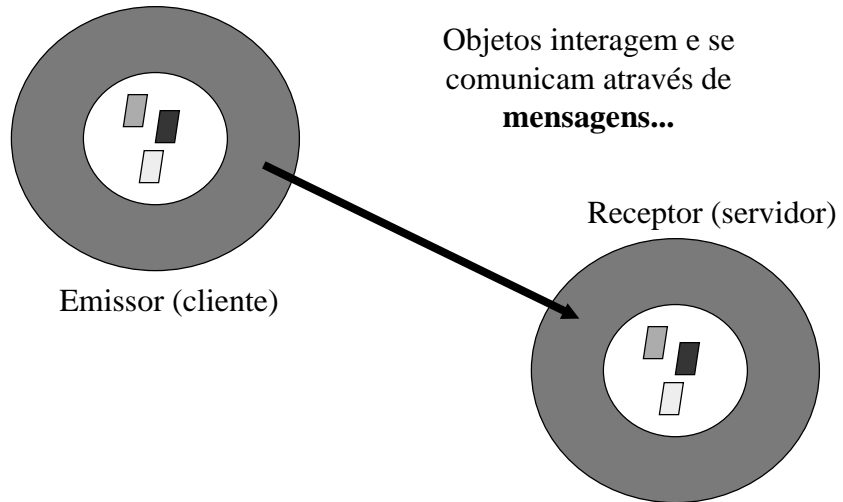
Todo o acesso aos dados é feito através da chamada a um método definido pelo objeto

Mudanças na implementação interna do objeto (que preservem a sua interface externa) não afetam o resto do sistema

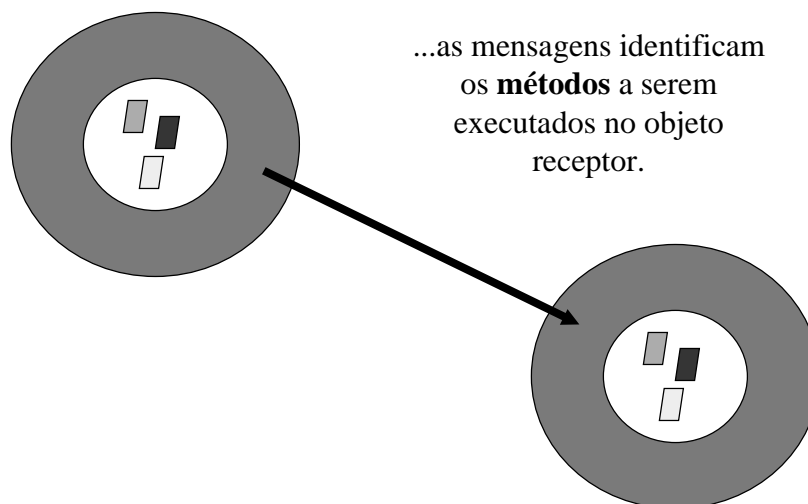
## Encapsulamento - benefícios

- **Segurança:** protege os objetos de terem seus atributos corrompidos por outros objetos.
- **Independência:** “escondendo” seus atributos, um objeto protege outros de complicações de dependência da sua estrutura interna.

## Mensagens



## Métodos

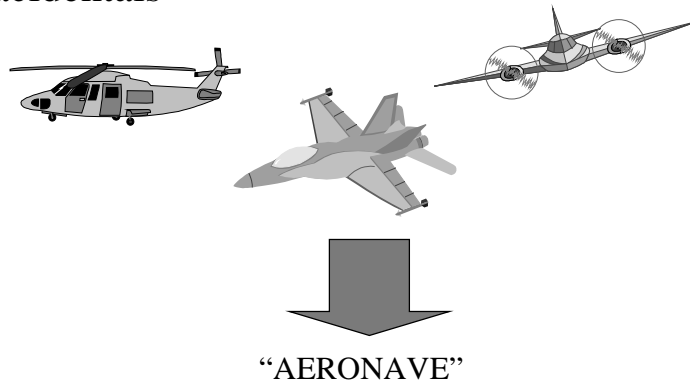


## Mensagens e Métodos

- Para invocar um método, deve-se enviar uma mensagem para o objeto desejado
- Para enviar uma mensagem, deve-se
  - identificar o **objeto** que receberá a mensagem
  - identificar o **método** que o objeto deverá executar
  - passar os **argumentos** requeridos pelo método

## Abstração

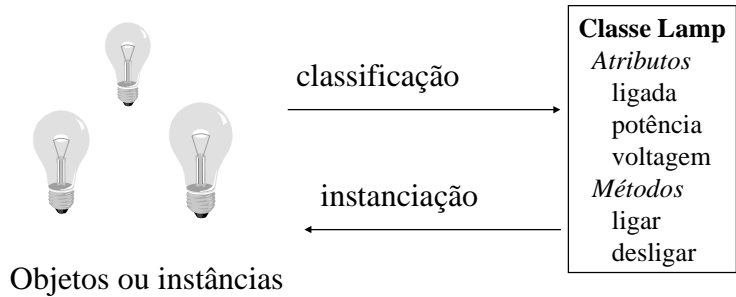
- Focalizar o essencial, ignorar propriedades acidentais



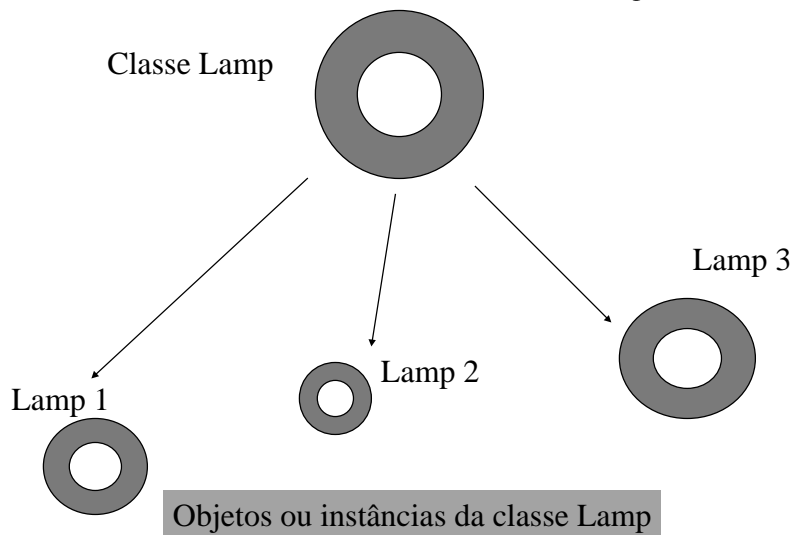


# Classes

- Uma classe determina um conjunto de objetos com:
  - propriedades semelhantes
  - comportamentos semelhantes
  - relacionamentos comuns com outros objetos



## Classe: uma fábrica de objetos



## Código da Classe Lamp

```
import java.lang.*;

class Lamp {
    private int potencia;
    private int voltagem;
    private boolean ligada;

    public Lamp(int pot,int volt){
        potencia = pot;
        voltagem = volt;
        ligada = false;
    }

    public void ligar {
        ligada = true;
    }

    public void desligar {
        ligada = false;
    }
}
```

## Código da Classe Lamp

```
import java.lang.*;

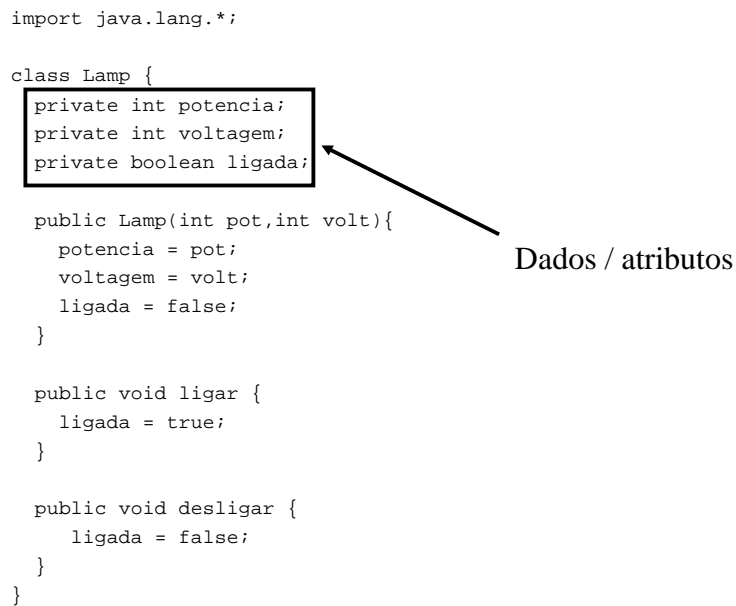
class Lamp {
    private int potencia;
    private int voltagem;
    private boolean ligada;

    public Lamp(int pot,int volt){
        potencia = pot;
        voltagem = volt;
        ligada = false;
    }

    public void ligar {
        ligada = true;
    }

    public void desligar {
        ligada = false;
    }
}
```

Dados / atributos

A diagram illustrating the code for the Lamp class. The code is identical to the one above. A rectangular box highlights the three private attributes: 'private int potencia;', 'private int voltagem;', and 'private boolean ligada;'. An arrow points from the text 'Dados / atributos' to the right side of this box.

## Código da Classe Lamp

```
import java.lang.*;

class Lamp {
    private int potencia;
    private int voltagem;
    private boolean ligada;

    public Lamp(int pot,int volt){
        potencia = pot;
        voltagem = volt;
        ligada = false;
    }
}
```

```
public void ligar {
    ligada = true;
}

public void desligar {
    ligada = false;
}
}
```

Operações / métodos

