

INSTITUTO FEDERAL DE
EDUCAÇÃO, CIÊNCIA E TECNOLOGIA
RIO GRANDE DO NORTE
Campus Natal - Central

Programação Estruturada e Orientada a Objetos

Fundamentos Orientação a Objetos

2013

O que veremos hoje?

- Introdução aos fundamentos de Orientação a Objetos

Transparências baseadas no material do
Prof. Jailton Carlos

Objetivos

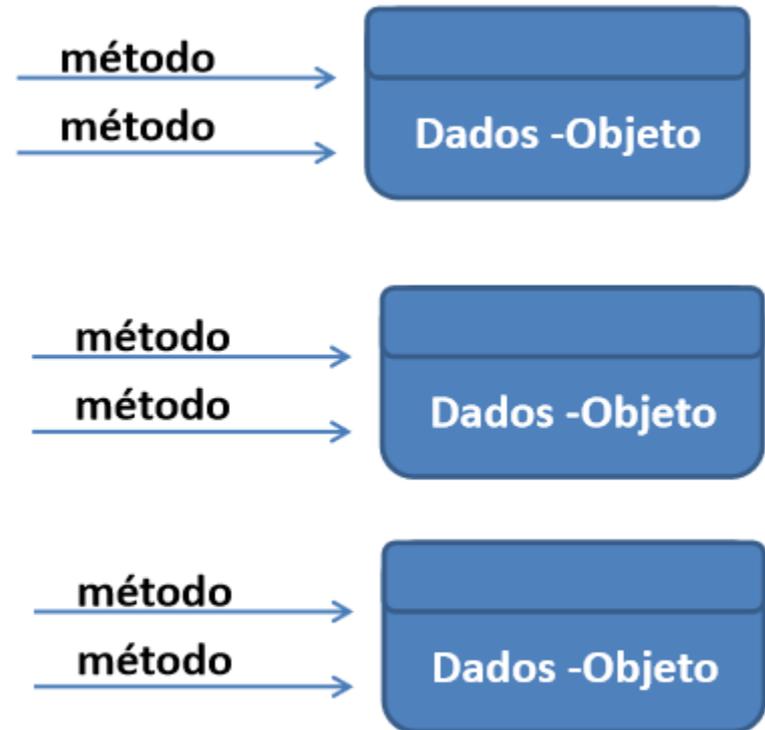
- Comparação Programação Estrutura e POO
- Compreender os conceitos de Programação Orientada a Objetos (POO);
- Entender as vantagens de se utilizar POO;
- Entender os conceitos de POO aplicado a uma linguagem de programação (C# como exemplo);
- Aprender a criar classes e utilizá-las em um contexto prático.

Plano de Aula

- Comparação Programação Estrutura e POO
- Definição de Objetos
 - *Conceito, propriedades*
- Definição de Classe
- Elementos Principais de Programação Orientada a Objeto
 - *Abstração, Encapsulamento, Modularidade, Herança, Polimorfismo*
- Visibilidade
- *Classe em C#*

Paradigma

Estrutural x Orientação a Objeto



Paradigma

Estrutural x Orientação a Objeto

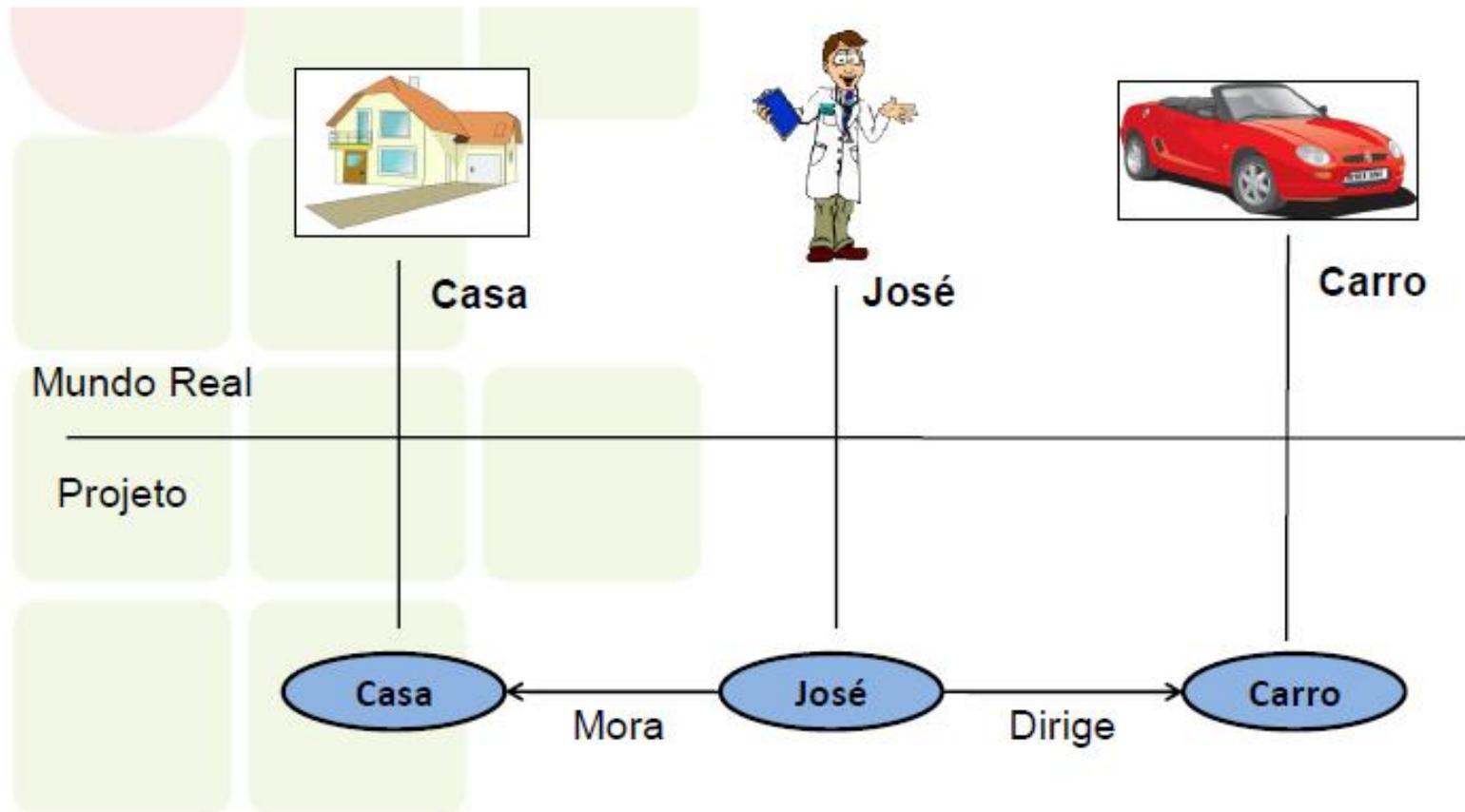
- Consiste no mapeamento do problema no mundo real a ser resolvido num modelo computacional.
- Programação Estrutura
 - Consiste na criação de um conjunto de procedimentos (algoritmos) para resolver o problema
 - Encontrar modos apropriados de armazenar os dados
- Programação Orientada a Objetos
 - Consistem em identificar os objetos e as operações relevantes no mundo real
 - O mapeamento desses em representações abstratas no espaço de soluções

Programação Orientada a Objetos

- Paradigma de Programação
 - Dominante nos dias atuais
- Substituiu as técnicas de programação procedimental (estruturada)
- “Fornece um mapeamento direto entre o mundo real e as unidades de organização utilizadas no projeto”
- Diversas unidades de software, chamadas de objetos, que interagem entre si
- Separa claramente a noção de o que é feito de como é feito

Programação Orientada a Objetos

- Representação:



Objetos

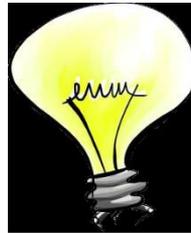
- Características
 - Dados representam características
 - São chamados **atributos**
 - São as variáveis do objeto
- Comportamento
 - Operações definem comportamento
 - São os **métodos** de um objeto
 - São as funções que são executadas por um objeto

Objetos - Propriedades

- Estado
 - Representado pelos valores dos atributos de um objeto
- Comportamento
 - Definido pelo conjunto de métodos do objeto
 - Estado representa o resultado cumulativo de seu comportamento
- Identidade
 - Um objeto é único, mesmo que o seu estado seja idêntico ao de outro;
 - Seu valor de referência
- Os valores dos **DADOS** são modificados a partir das **OPERAÇÕES** sobre estes dados

Objetos - Propriedades

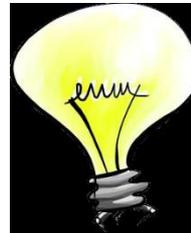
- Estado



Acesa

Apagada

- Comportamento



Acender

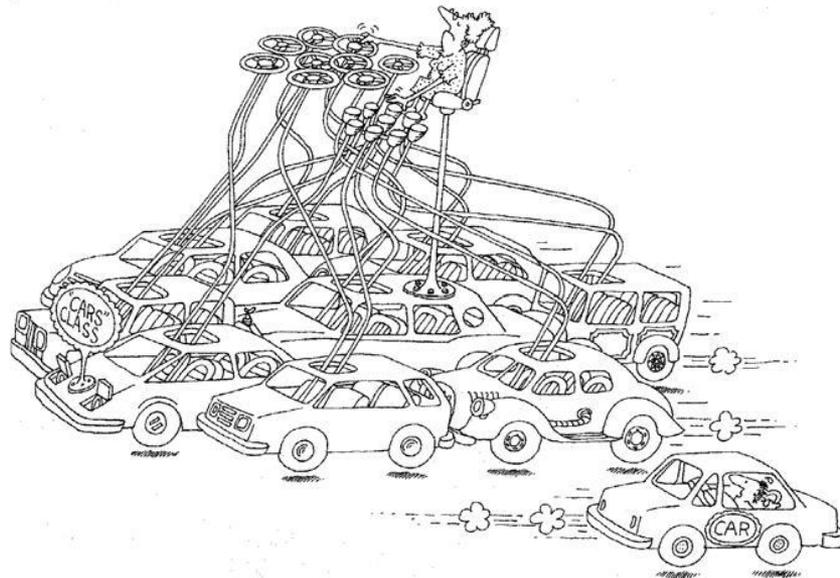
Apagar

- Identidade



Classes

- São especificações para objetos;
- Representam um conjunto de objetos que compartilham características e comportamentos comuns.



Todo carro tem em comum:

Característica

Cor

Pneu

Direção

Comportamento

Dirigir

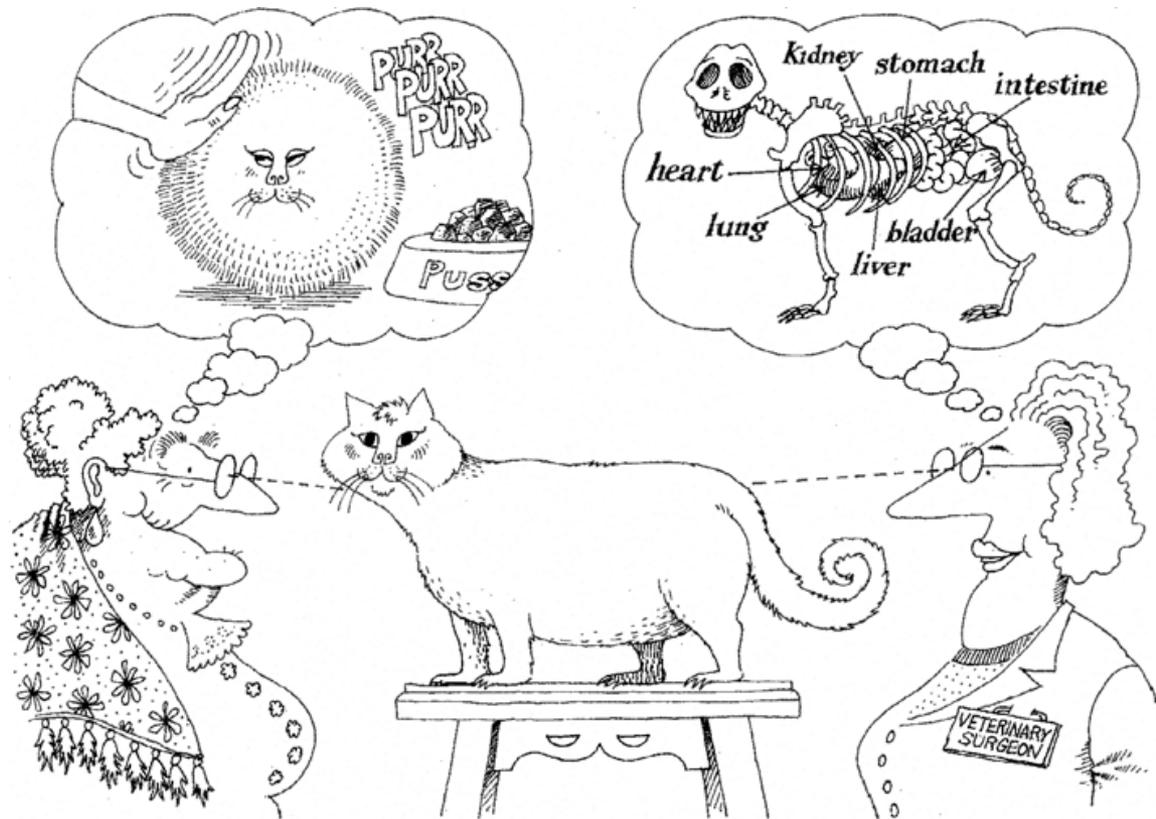
Frear

Abstração

- Abstração é uma das formas fundamentais que nós lidamos com a complexidade.;
- Quando queremos diminuir a complexidade de alguma coisa, ignoramos detalhes sobre as partes para concentrar a atenção no nível mais alto de um problema;
- Não se analisa o “todo”, em POO é importante analisar as partes para entender o todo.

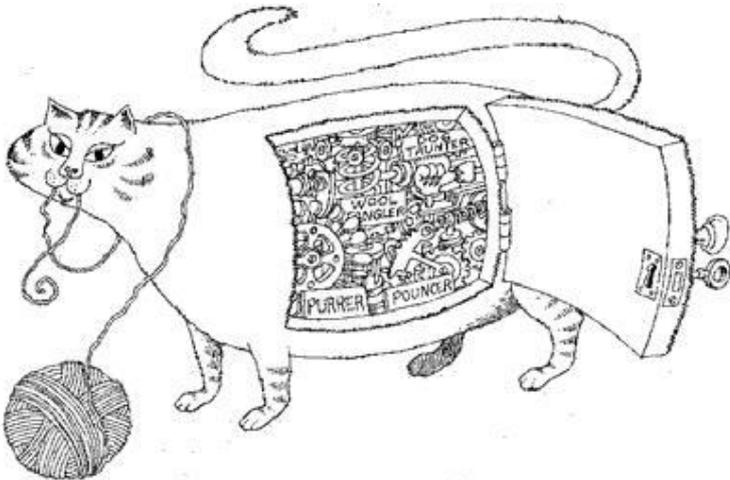
Abstração

- Foca a característica essencial de alguns objetos relativo a perspectiva do visualizador



Encapsulamento

- Encapsulamento é o processo de esconder todos os detalhes de um objeto que não contribuem para as suas características essenciais;
- O encapsulamento é o modo de dar ao objeto seu comportamento “caixa-preta”, que é o segredo da reutilização e confiabilidade.



Se o estado de um objeto foi modificado sem uma chamada de método desse objeto, então o encapsulamento foi quebrado

Encapsulamento e Abstração

- São conceitos complementares
- Abstração foca sobre o comportamento observável de um objeto, enquanto encapsulamento se concentra na execução que dá origem a esse comportamento

Modularidade

- Modularização é o processo de dividir um todo em partes bem definidas, que podem ser construídas e examinadas separadamente.
- Essas partes se interagem entre si, fazendo com que o sistema funcione de forma adequada
- Particionar um programa em componentes individuais, pode reduzir a complexidade.

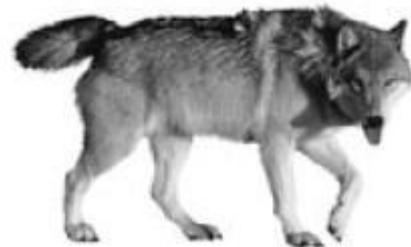
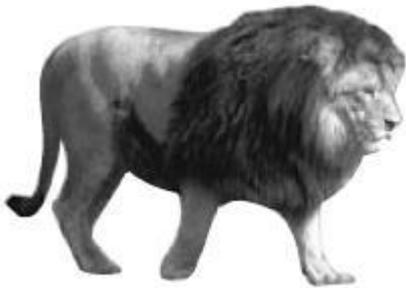


Herança

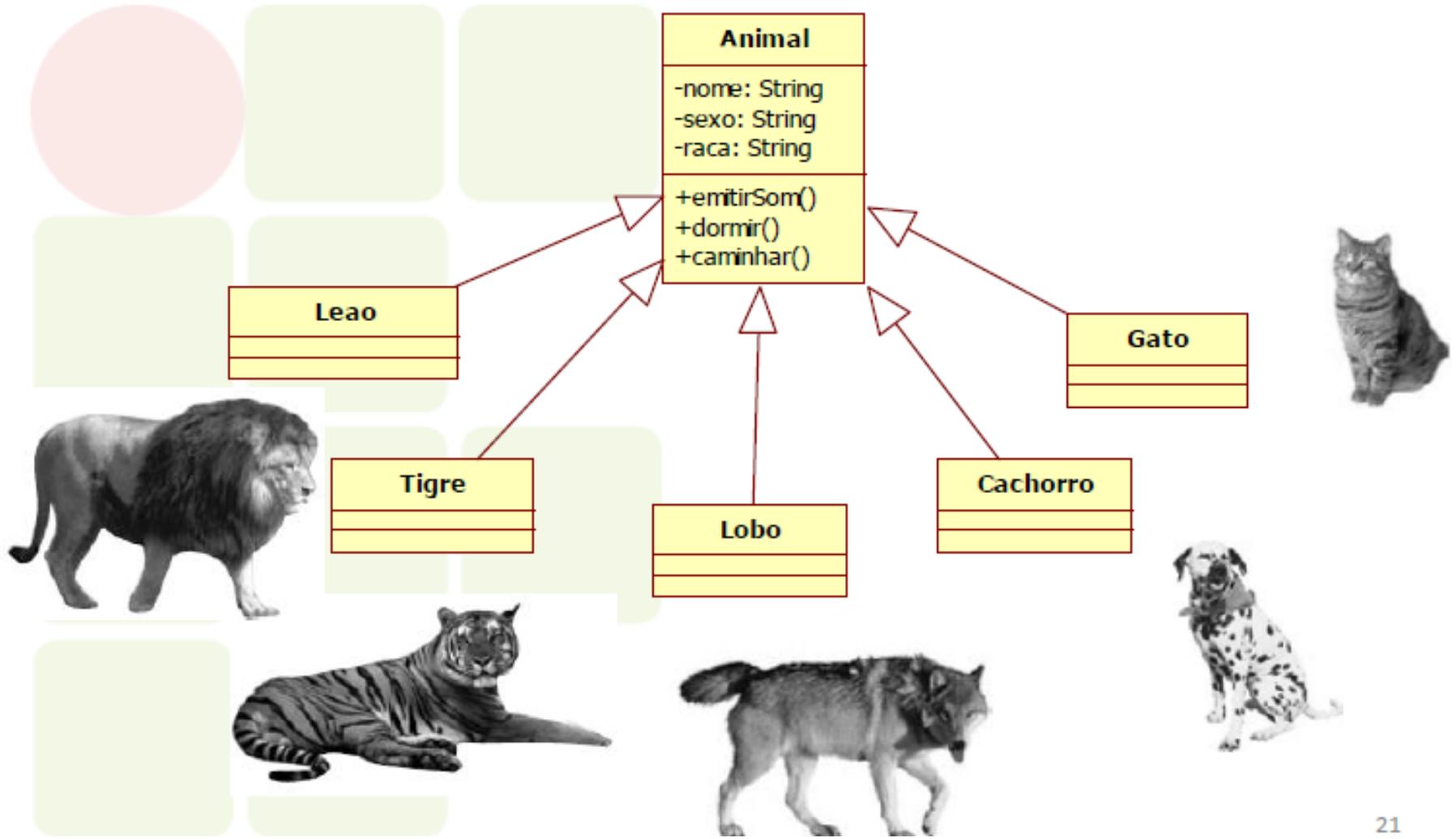
- A abstração ajuda a diminuir a complexidade.
- Encapsulamento ajuda a gerenciar essa complexidade, ocultando a visão dentro de nossa abstrações.
- A modularidade também ajuda, dando-nos uma maneira de agrupar logicamente abstrações relacionadas.
- Um conjunto de abstrações, muitas vezes forma uma hierarquia, e identificando essas hierarquias no nosso projeto, simplifica grandemente o nossa compreensão do problema.

Herança

- Herança é o mecanismo para expressar a similaridade entre Classes, simplificando a definição de classes iguais que já foram definidas.
- O que um leão, um tigre, um gato, um lobo e um dálmatas têm em comum?
- Como eles são relacionados?



Herança



Polimorfismo

- Polimorfismos
 - **Poli** -> varias; **Morfos** -> formas;
- Significa que um objeto pode assumir diferentes formas;
- O conceito de polimorfismo está associado a Herança;
- É caracterizado como o fato de uma operação poder ser implementada de diferentes maneiras pelas classes na hierarquia.

Polimorfismo

Cada animal emite sons diferentes.

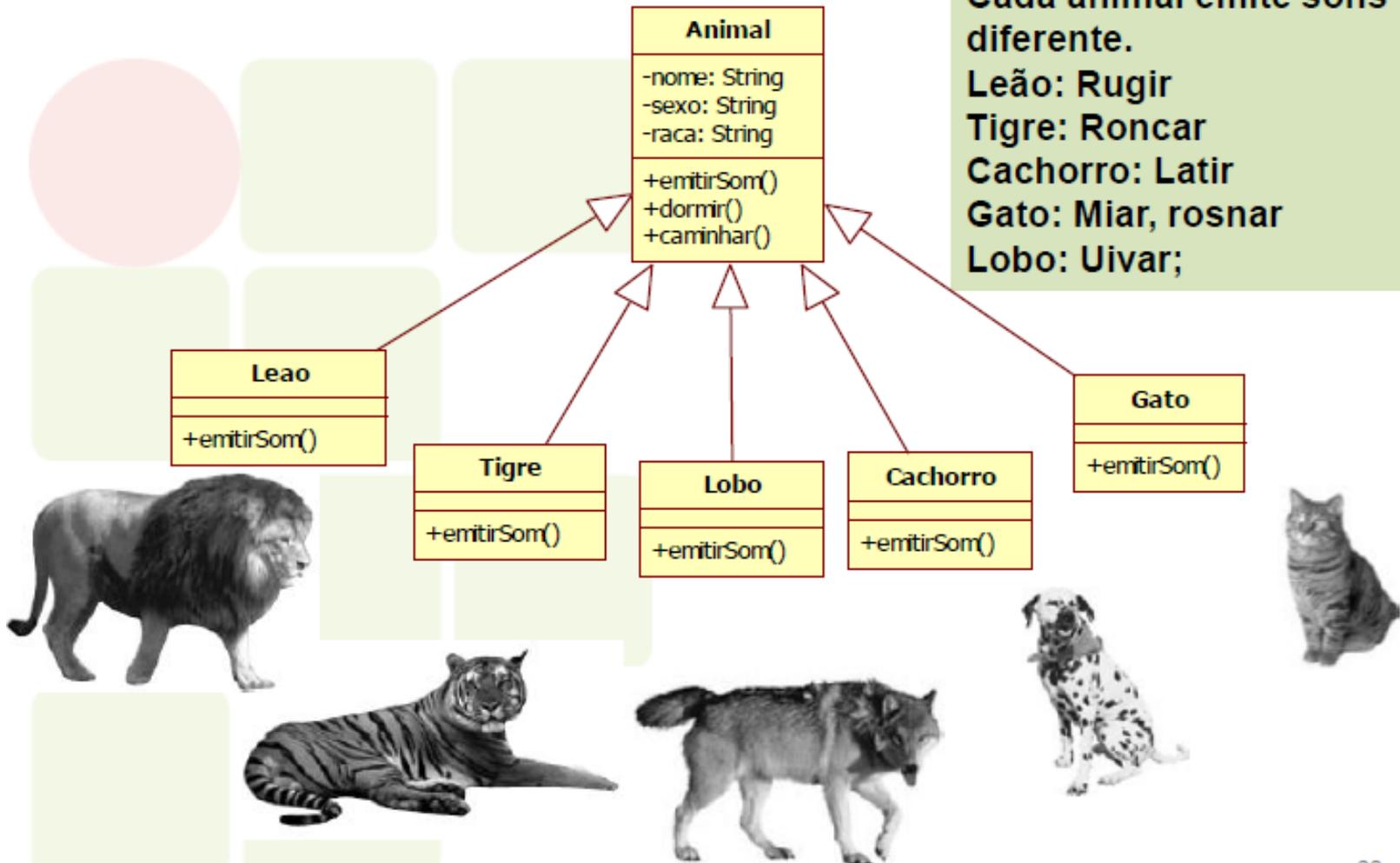
Leão: Rugir

Tigre: Roncar

Cachorro: Latir

Gato: Miar, rosnar

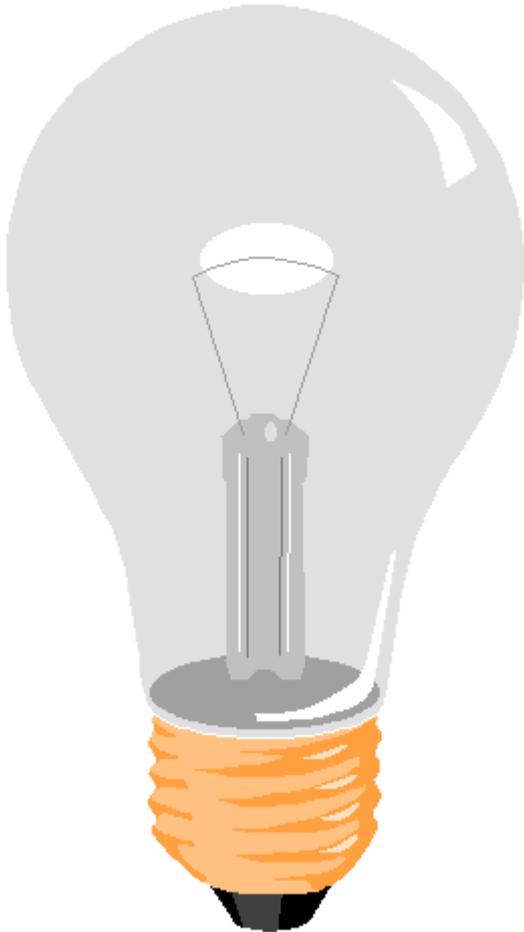
Lobo: Uivar;



Visibilidade

- Private
 - O nível de acesso se restringe apenas a classe;
 - Não é passado por herança;
- Public
 - O nível de acesso é irrestrito;
 - Por padrão, é a visibilidade definida para métodos e atributos em uma classe
- Protected
 - É visível em toda a classe;
 - É passado por herança (mesmo em pacotes diferentes);
- Internal
 - Com este modificador, o acesso é limitado apenas ao assembly atual.
- Protected Internal
 - Com este modificador, o acesso é limitado ao assembly atual e aos tipos derivados da classe que contém o modificador.

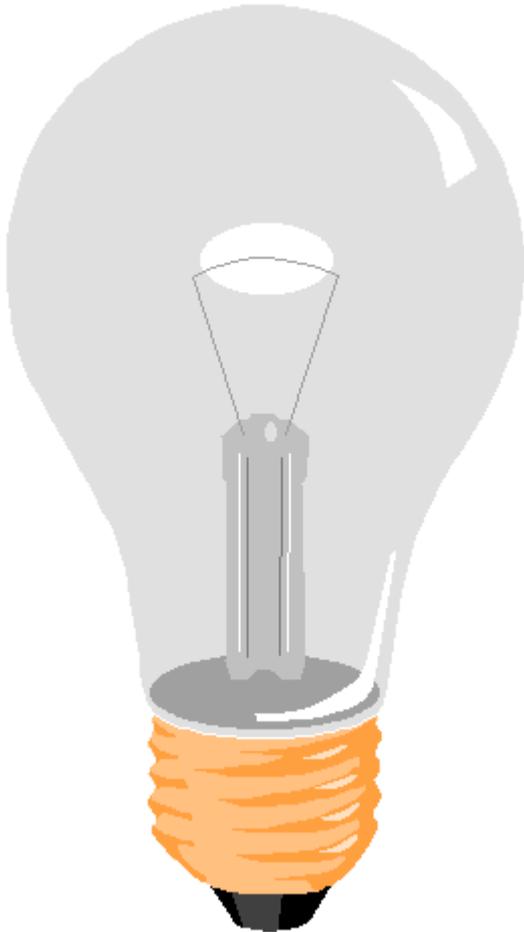
Classes



- Classe Lampada
 - Atributos
 - potencia, ligada
 - métodos
 - ligar, desligar, estaLigada

Lampada
- ligada : boolean - potencia : double
+ ligar() : void + desligar() : void + estaLigada() : boolean

Classes



Nome da classe

Atributos

métodos

Lampada
- ligada : boolean - potencia : double
+ ligar() : void + desligar() : void + estaLigada() : boolean

Classes em C#

- Declaração de uma classe em C#
 - A palavra-chave *class* é utilizada para definir uma nova classe
 - A classe Lâmpada, por exemplo, representa uma entidade lâmpada
 - Cada lâmpada armazena o seu estado (ligado ou desligado) e realiza as operações ligar e desligar

Classes em C#

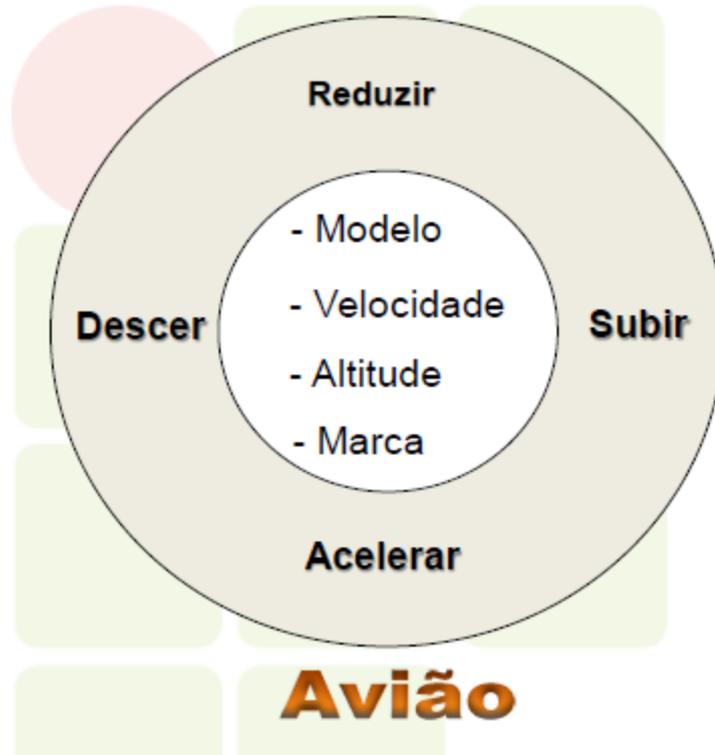
- Declaração de uma classe em C#

```
public class Lampada
{
    private bool ligado;
    public void ligar() { ligado= true; }
    public void desligar() { ligado=false; }
    public bool estaLigada(){return ligado;}
}
```

Resumo

- Objeto
 - Qualquer entidade que possui características e comportamento
- Classe
 - Descreve um tipo de objeto
 - Define atributos e métodos
- Atributo
 - Define características do objeto
- Método
 - Operações que o objeto pode realizar

Exercício – implemente as classes abaixo em C#



Exercícios

- Imagine uma lâmpada que possa ter três estados: apagada, acesa e meia-luz. Usando a classe `Lampada`, vista nas transparências, escreva uma classe em C# para essa nova lâmpada.
- Crie uma classe `Livro` que represente os dados básicos de um livro.
- Escreva uma classe contenha a data do dia das mães, ela deve conter dia, mês e ano.

Dúvidas

