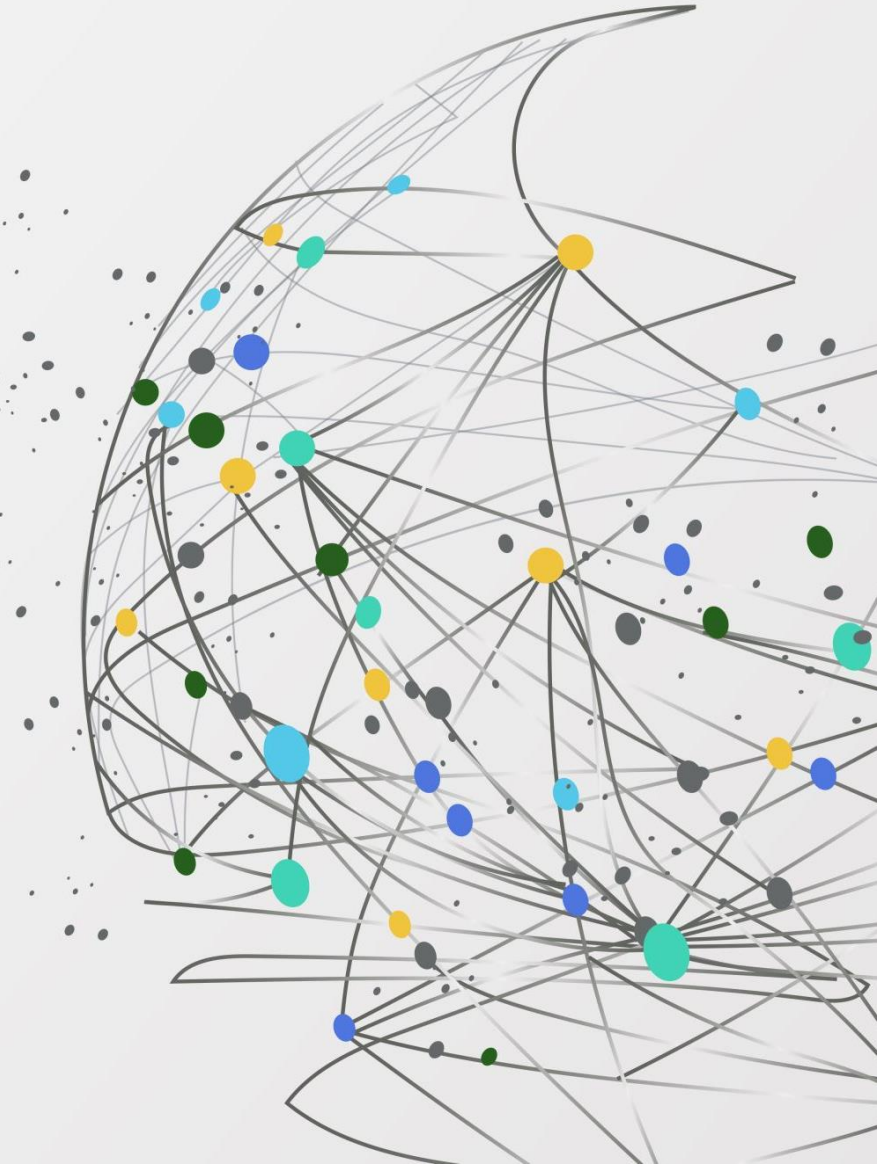


SW-I
Sistemas Web I

Prof. Anderson Vanin

Competências, Habilidades, Bases Tecnológicas e Critérios de Avaliação.

- Base Tecnológica.

Contrato Pedagógico

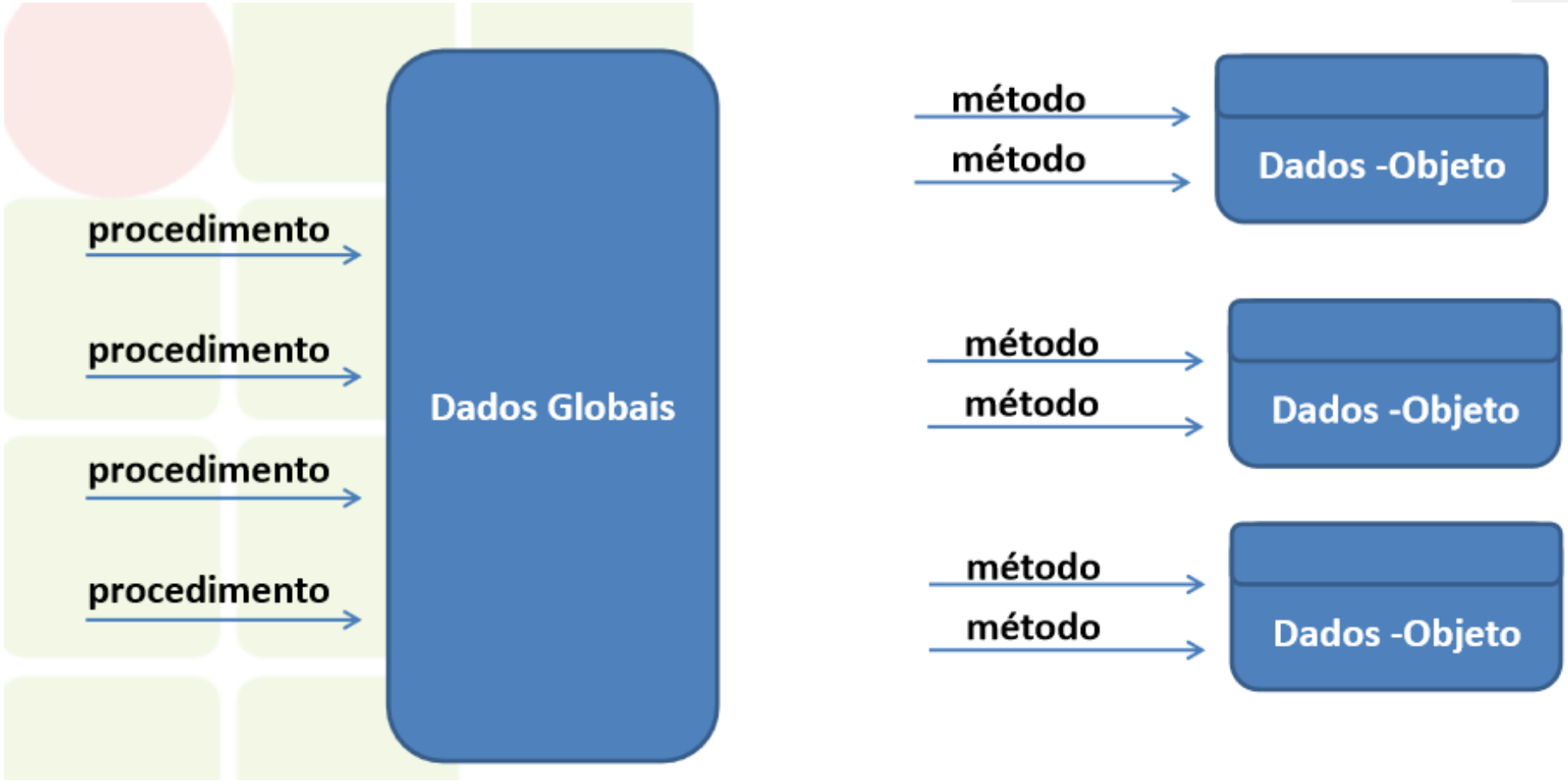
- No mínimo 3 atividades solicitadas durante o Bimestre.
- Entrega das atividades em dia.
- Assiduidade e Participação nas aulas.
- Competências socioemocionais.
- Atividades práticas Individuais.

Fundamentos Orientação a Objetos

Orientação a objeto é um conceito que esta relacionado com a ideia de classificar, organizar e abstrair coisas. Veja a definição formal: "O termo orientação a objetos significa organizar o mundo real como uma coleção de objetos que incorporam estrutura de dados e um conjunto de operações que manipulam estes dados.

A orientação a objetos surgiu com o objetivo de tornar o desenvolvimento de software menos complexo e mais produtivo.

Paradigma Estrutural x Orientação a Objeto



Programação Orientada a Objetos - Definição de Objetos

- Um objeto é algo do mundo real :
  - **– Concreto ou Abstrato**
- As percepção dos seres humanos é dada através dos objetos
- Um objeto é uma entidade que exibe algum comportamento bem definido.

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

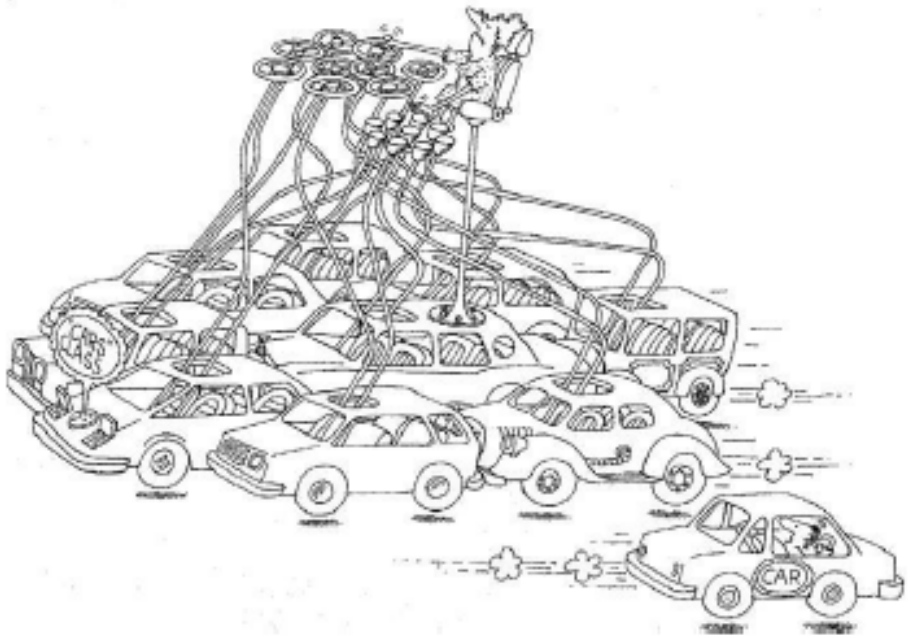
Acender

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

É a capacidade de focar nos pontos mais importantes do domínio de aplicação do sistema e abstrair os detalhes menos relevantes. Na modelagem de um sistema orientado a objetos, uma classe tende a ser a abstração de entidades existentes no mundo real (domínio da aplicação). Ex.: Cliente, Funcionário, Conta Bancária.

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.

Encapsulamento

Prevê o isolamento a determinados elementos do objeto (métodos /atributos) de acordo com a necessidade de acesso a eles. Este conceito parte da premissa de que nem todo método e atributo precisam estar visíveis e acessíveis publicamente. Existem elementos que são pertinentes apenas ao próprio objeto, outros pertinentes aos objetos filhos e outros que são pertinentes todos os objetos associados. O encapsulamento se dá através dos modificadores de acesso, que veremos mais a frente.

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

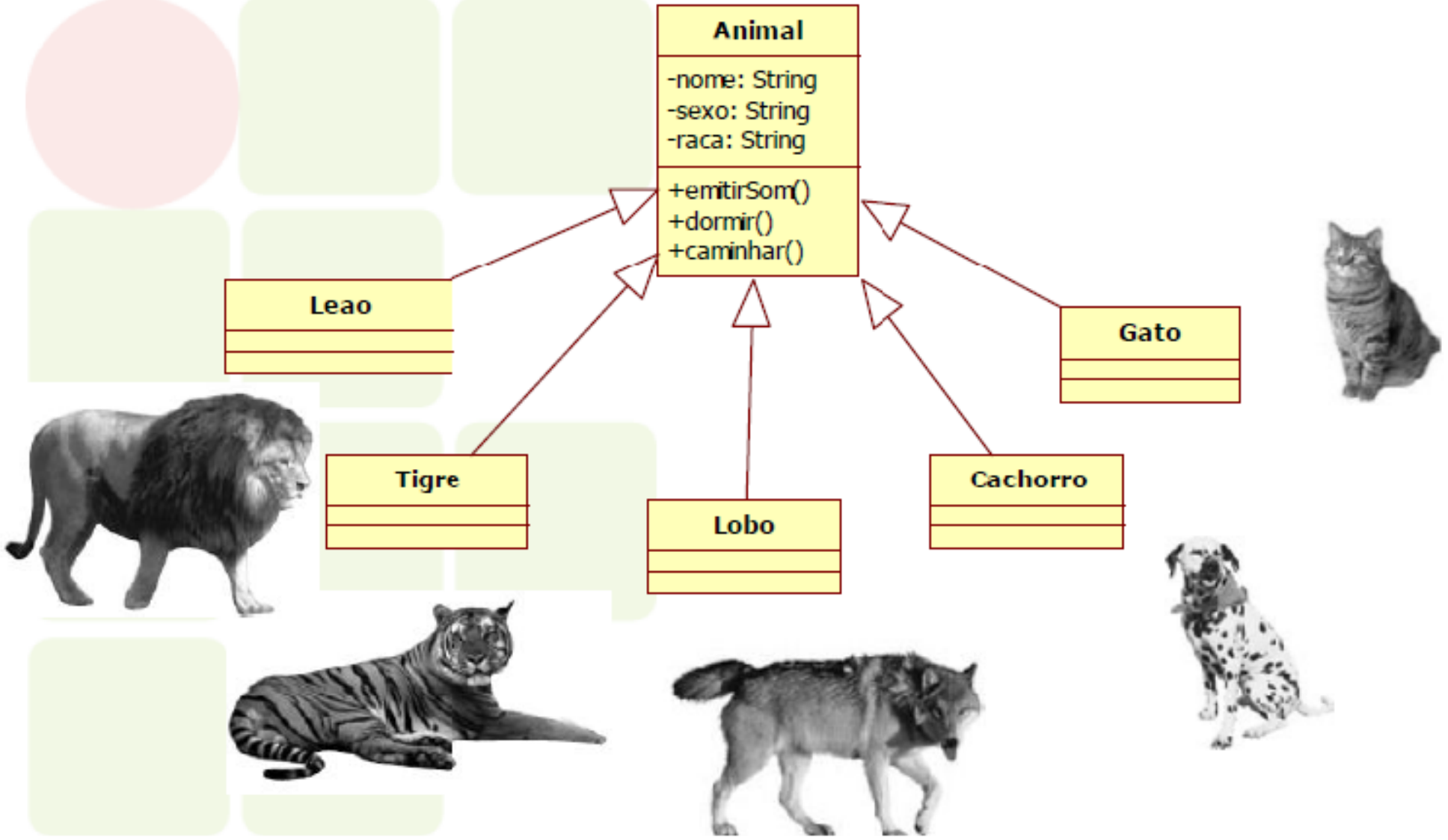
- A abstração ajuda a diminuir a complexidade.
- Encapsulamento ajuda a gerenciar essa complexidade, ocultando a visão dentro de nossa abstrações.
- A modularidade também ajuda, dando-nos uma maneira de agrupar logicamente abstrações relacionadas.
- Um conjunto de abstrações, muitas vezes forma uma hierarquia, 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

Herança

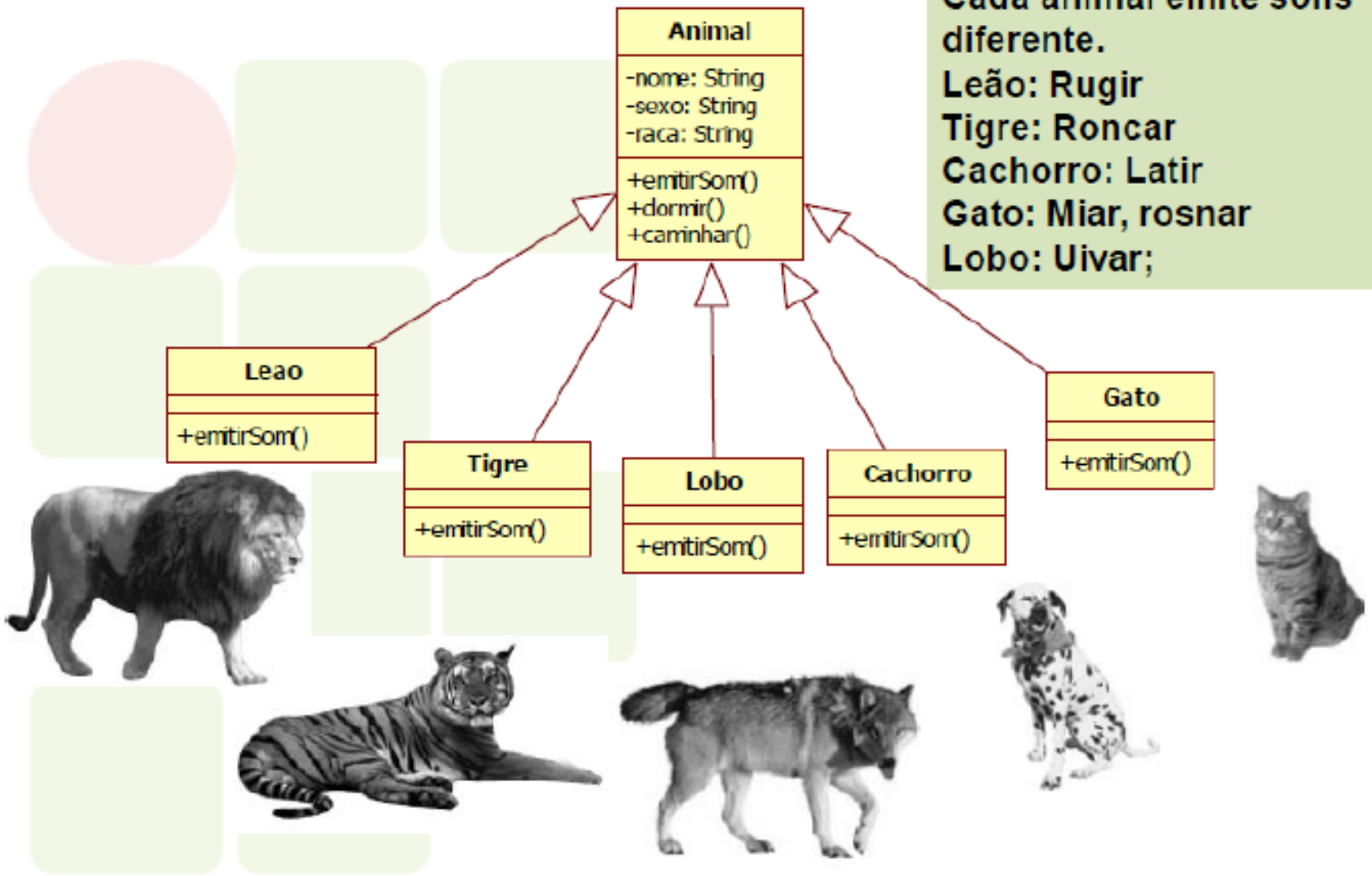

Animal
-nome: String
-sexo: String
-raca: String
+emitirSom()
+dormir()
+caminhar()
Leao
Tigre
Lobo
Cachorro
Gato

Polimorfismo

- **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



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.

Visibilidade

+ ➔ Público

- ➔ Privado

➔ Protegido

UML - Unified Modeling Language

A UML (Linguagem de Modelagem Unificada) é uma linguagem-padrão para a elaboração da estrutura de projetos de software. Ela poderá ser empregada para a visualização, a especificação, a construção e a documentação de artefatos que façam uso de sistemas complexos de software.

Diagramas da UML

Diagramas da UML 2.0

Diagramas estruturais

- Diagrama de classes
- Diagrama de objetos
- Diagrama de componentes
- Diagrama de instalação ou de implantação
- Diagrama de pacotes
- Diagrama de estrutura composta
- Diagrama de perfil

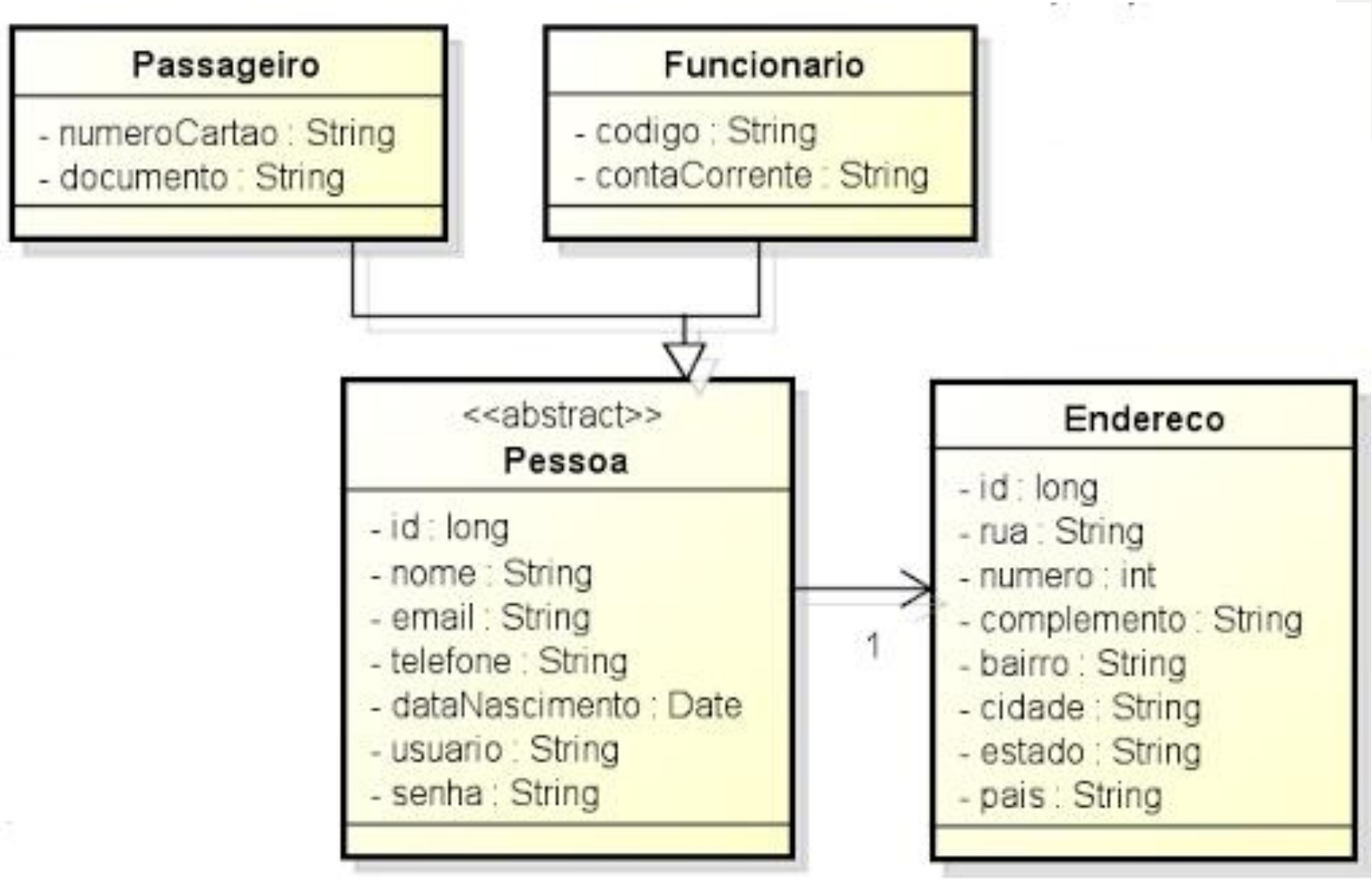
Diagramas comportamentais ou dinâmicos

- Diagrama de caso de uso
- Diagrama de transição de estados ou Máquina de estados
- Diagrama de atividade
- **Diagramas de interação**
  - Diagrama de sequência
  - Diagrama Visão Geral de Interação ou de interação
  - Diagrama de colaboração ou comunicação
  - Diagrama de tempo ou temporal

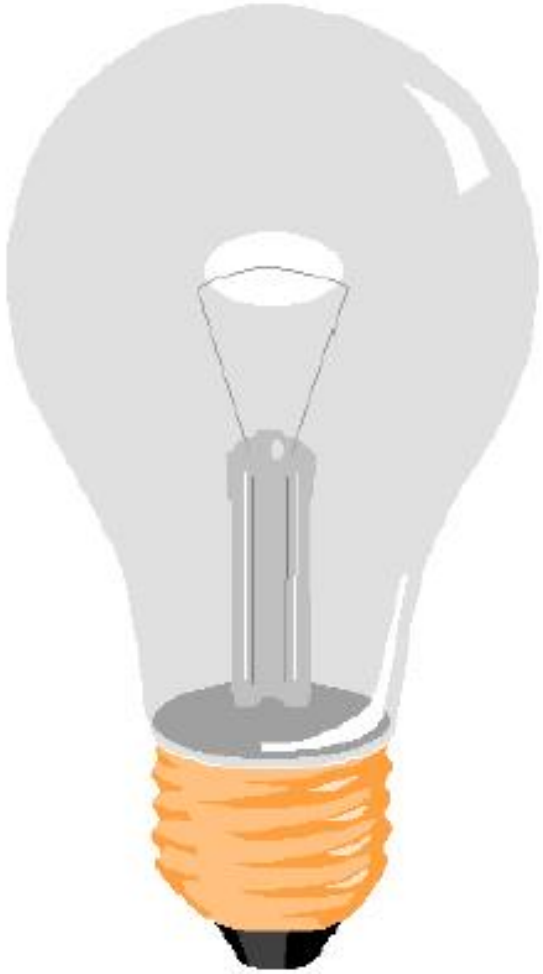
UML - Diagrama de Classes

O Diagrama de Classes é utilizado para fazer a representação de estruturas de classes de negócio, interfaces e outros sistemas e classes de controle. Além disso, o diagrama de classes é considerado o mais importante para a UML, pois serve de apoio para a maioria dos demais diagramas. Se dá pela formação de conjunto de informações sobre determinadas classes, que unidas entre si formam um sentido geral do projeto.

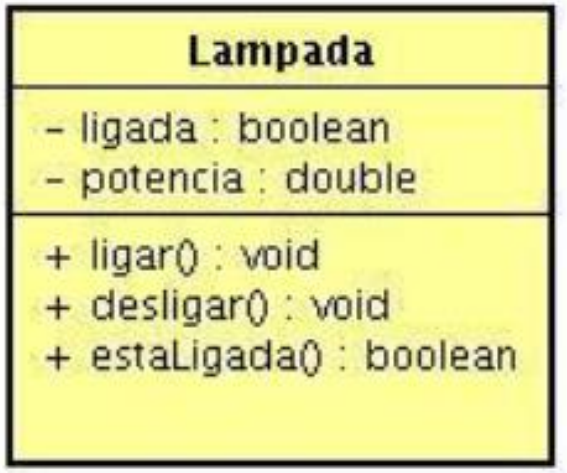
UML - Diagrama de Classes



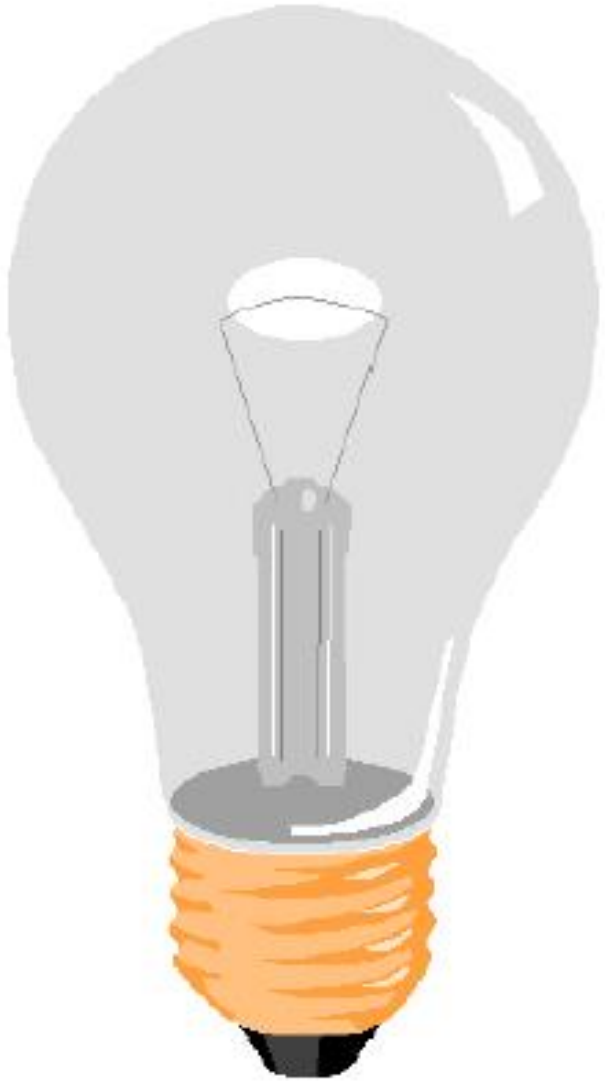
Classes



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



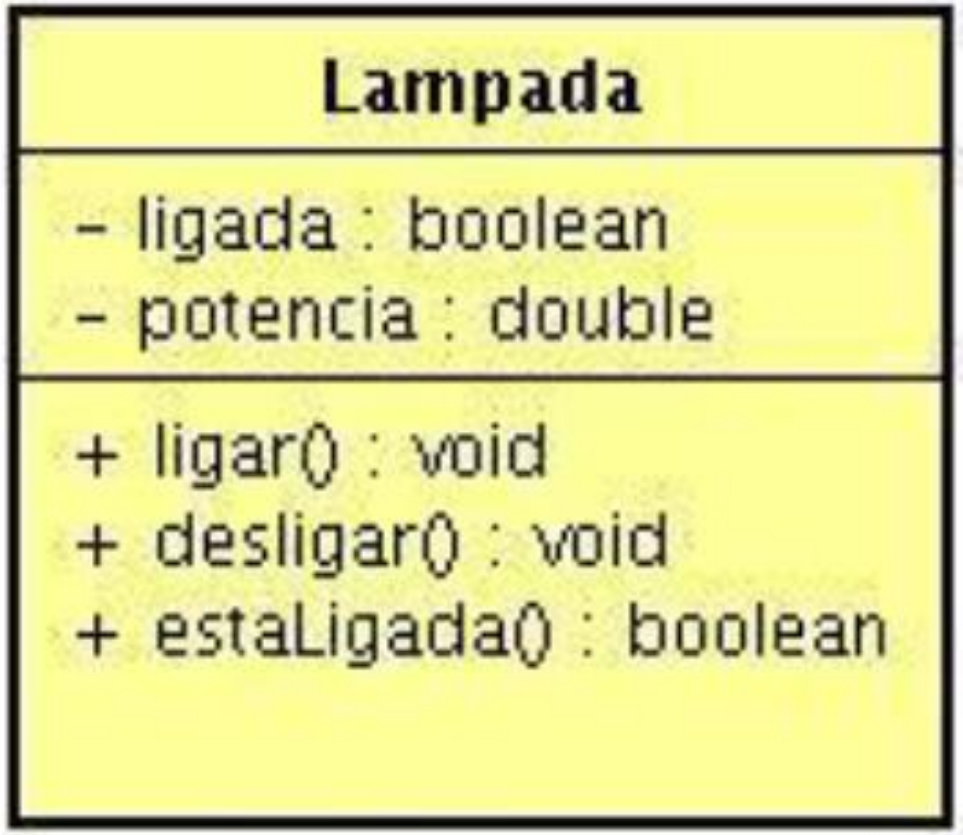
Classes



Nome da classe

Atributos

métodos



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#