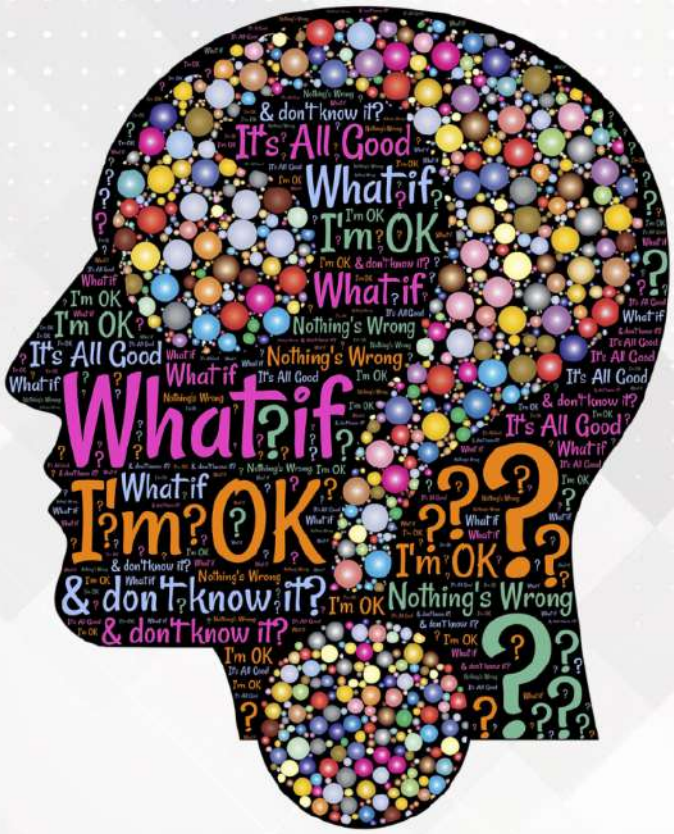
Programação Orientada a Objetos I

DigitalHouse > Coding School



Paradigma

DigitalHouse >
Coding School

Evolução dos Paradigmas

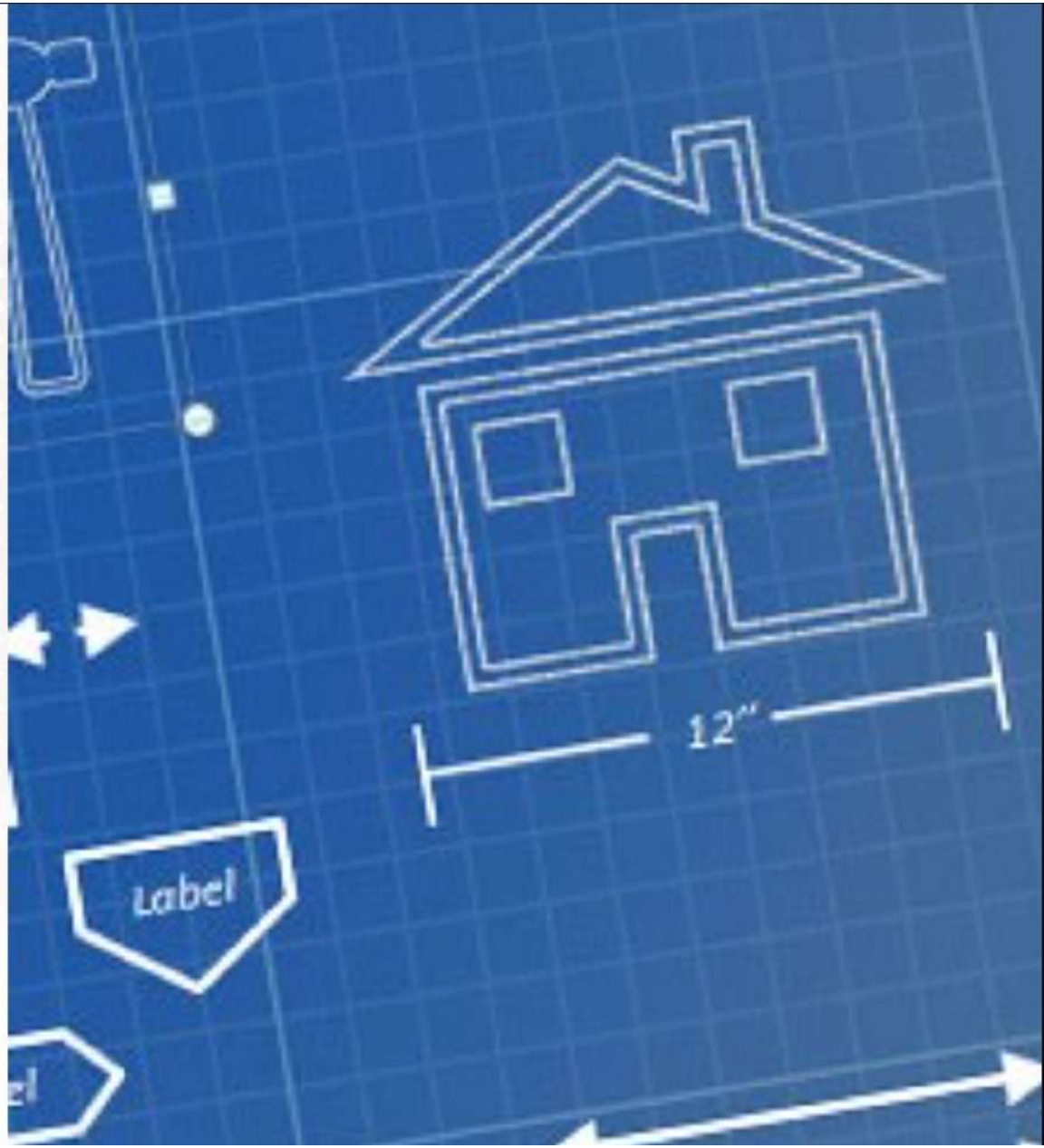
DigitalHouse > Coding School

Programação Estruturada

- Surgiu no início da década de 60 mediante a *Crise do Software*
- Características: Uso de subrotinas, Laços de repetição, condicionais e estruturas em bloco.
- Foi a base para a Orientação a Objetos

Programação
Orientada a Objetos

- Surgiu nos anos 60 atráves da linguagem Simula
- Alan Kay considerado um dos criadores do termo "Programação Orientada a Objetos"
- Objetivos: Facilitar o desenvolvimento de software e representar o mundo real.



DigitalHouse > Coding School

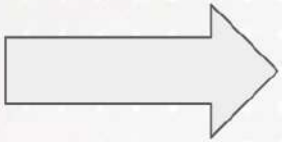
DigitalHouse > Coding School

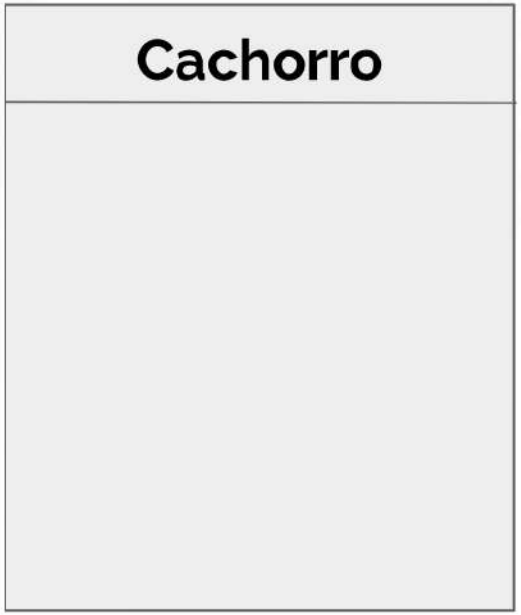
Classes

"As classes são modelos de um objeto, possuindo características e comportamentos."

Cachorro

Classes - Exemplo





Classes - Exemplo

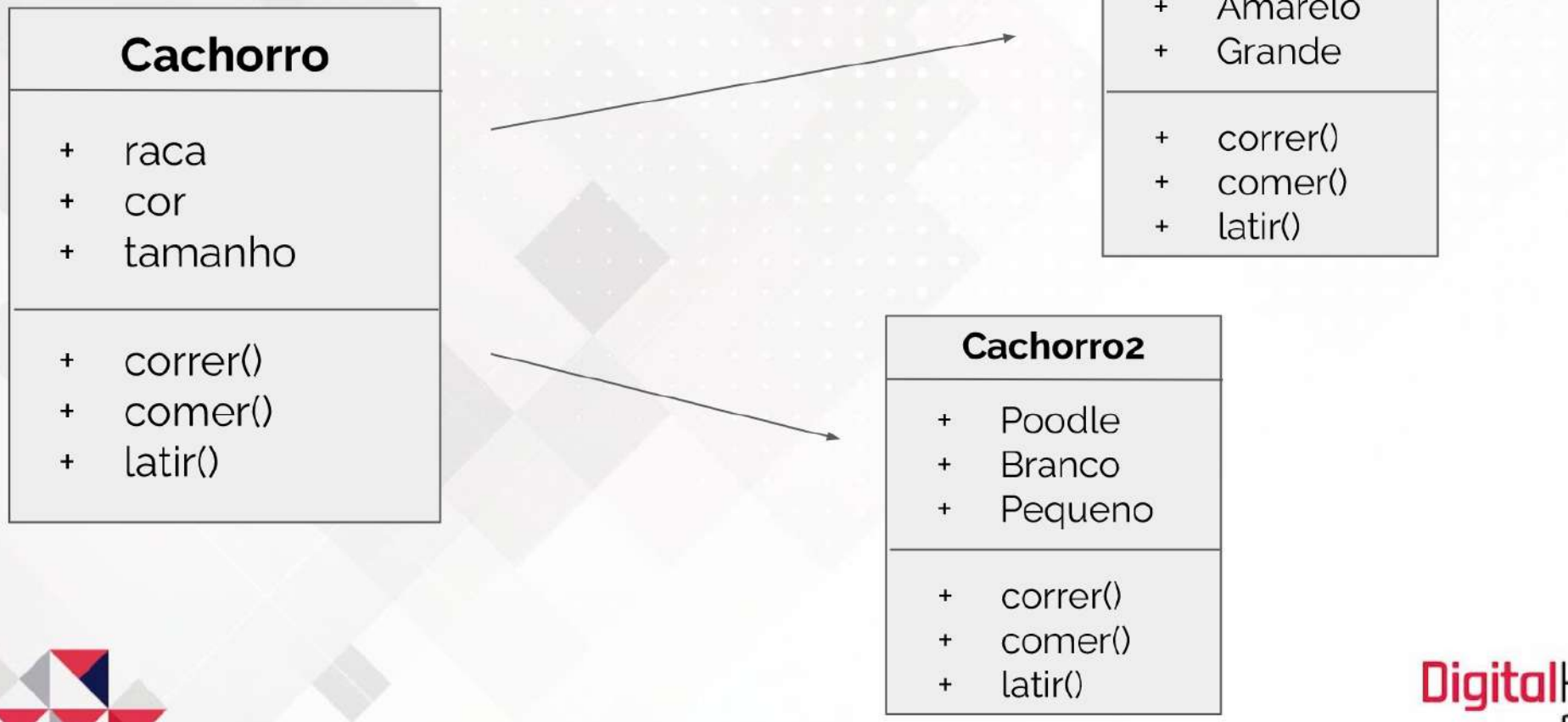
Características = Atributos

Comportamentos = Métodos

Objetos

DigitalHouse >
Coding School

Objetos - Exemplo



DigitalHouse > Coding School

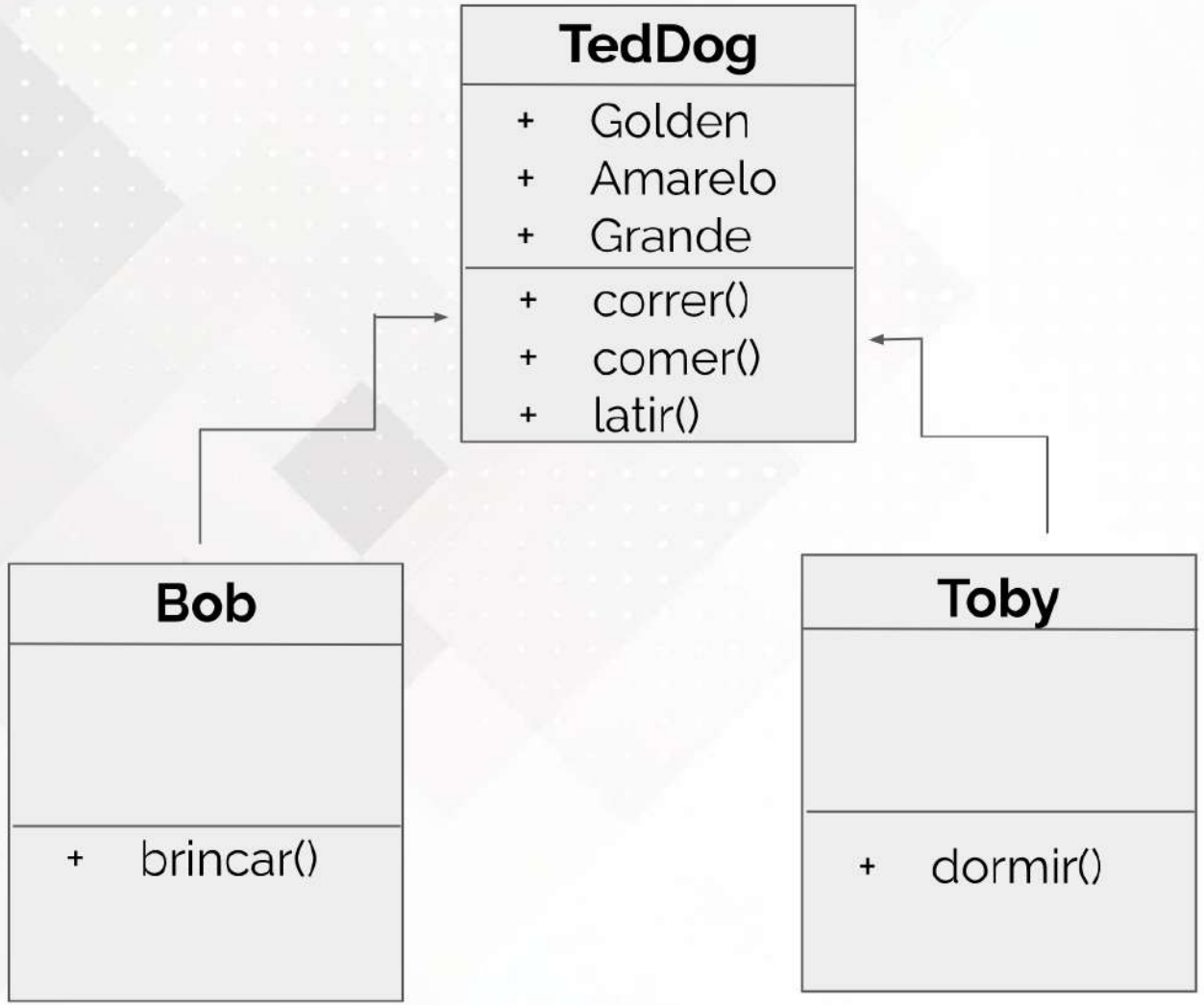
Herança

DigitalHouse >
Coding School

Herança

“Seria uma classe ter o poder de estender de outra classe atributos e métodos. Assim podendo reutilizar código.”

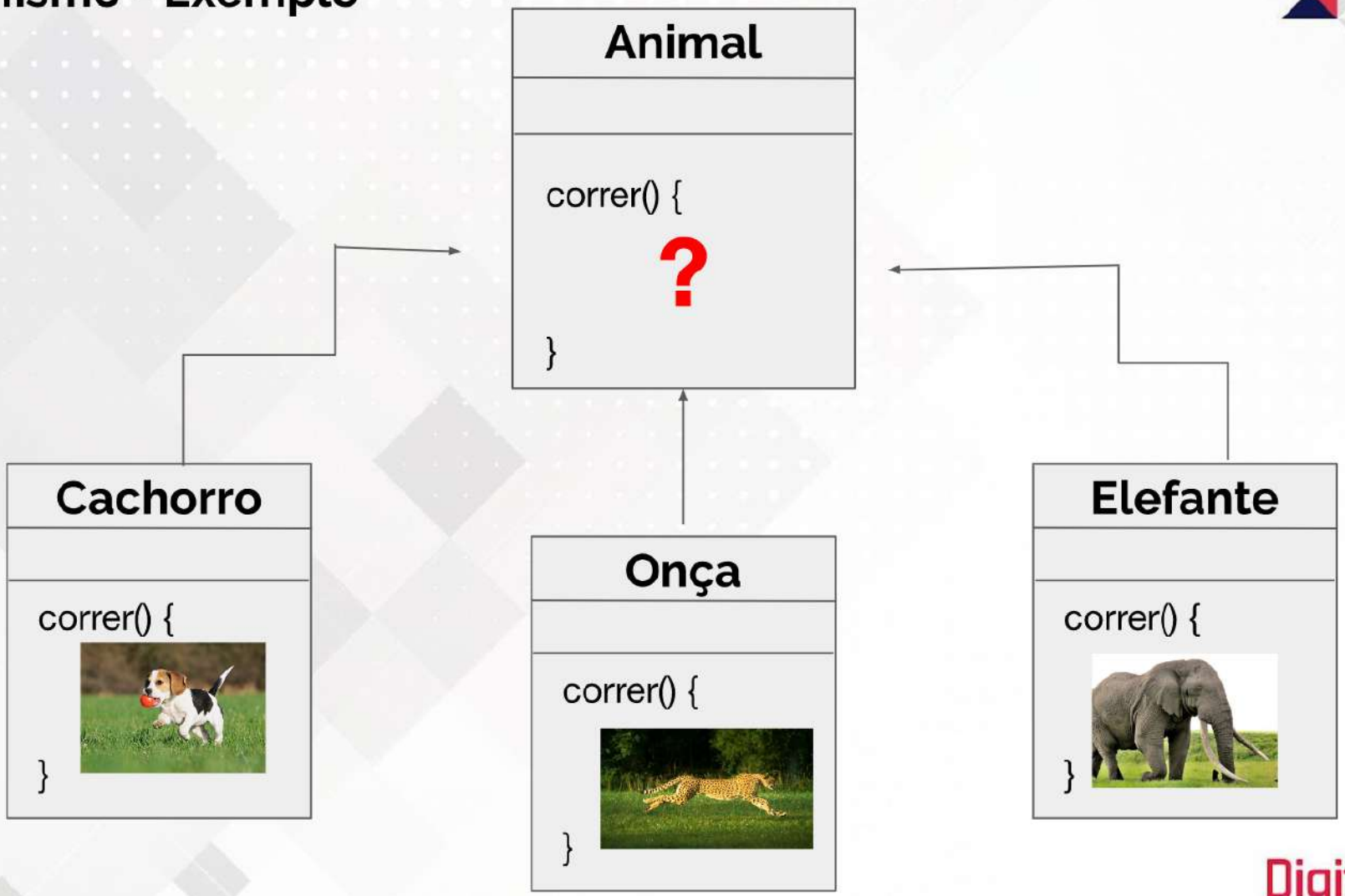
Herança - Exemplo



Polimorfismo

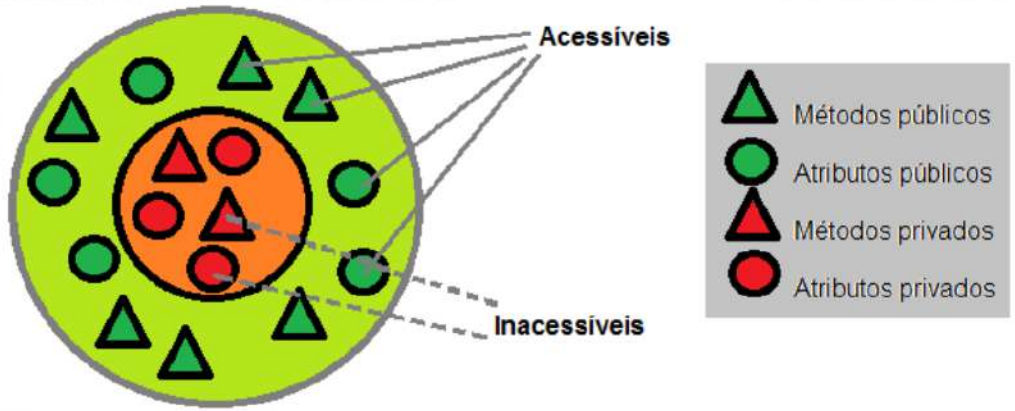
"Os mesmos atributos e métodos podem ser utilizados em objetos distintos, porém, com implementações lógicas diferentes."

Polimorfismo - Exemplo



Encapsulamento

“É uma forma de proteger parte dos dados independente do restante do sistema.”



DigitalHouse > Coding School

DigitalHouse >
Coding School

UML

Unified Modeling Language

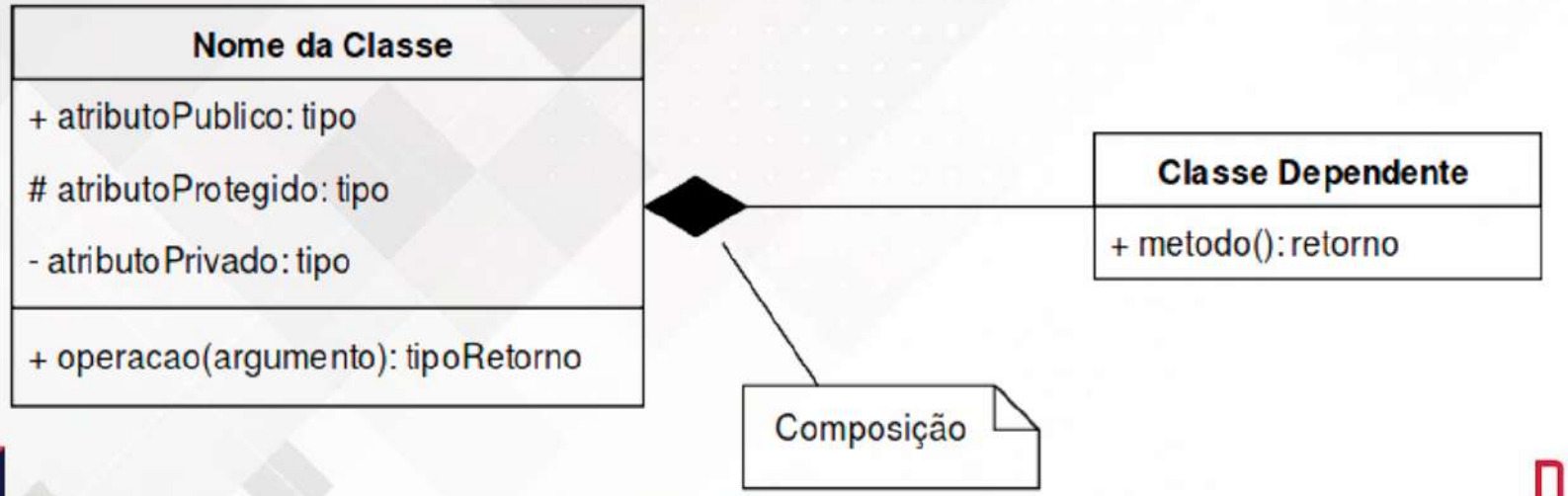
(Linguagem de Modelagem Unificada)

DigitalHouse > Coding School

Diagrama de Classes

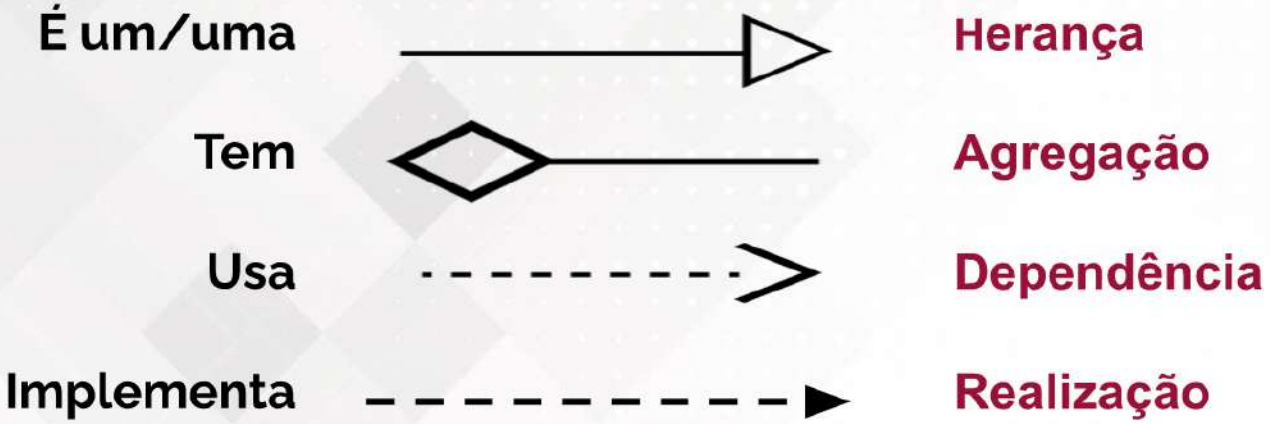
Tipo: int, String, boolean, **OBJETO**

Retorno: o resultado que devolve a função.



DigitalHouse > Coding School

Relacionamentos



Relacionamento – é um/uma

Pessoa
nome sexo cor_cabelo cor_roupa cor_pele cor_sapato altura humor
falar correr andar pensar

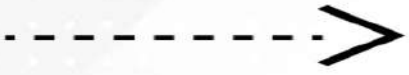
É uma

Relacionamento – Tem

Tem

Relacionamento – usa

usa

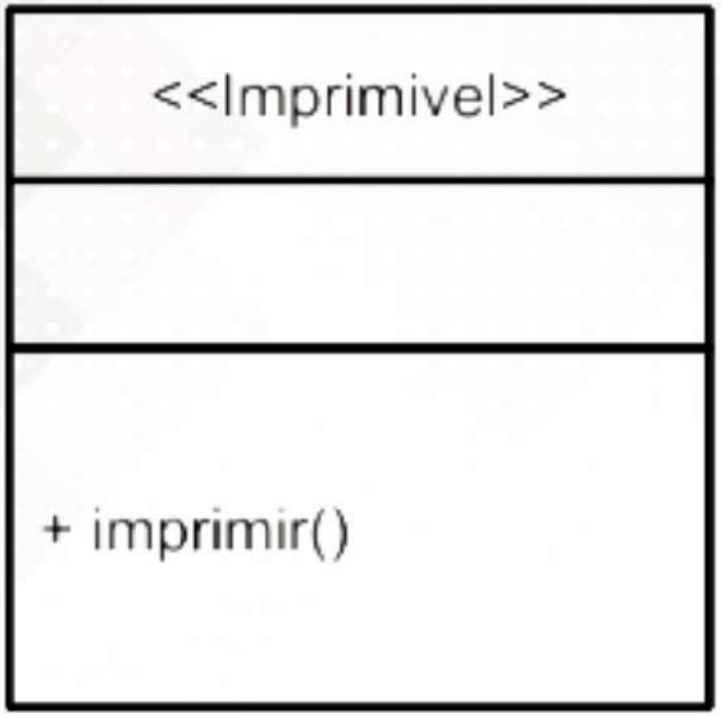


Veículo

DigitalHouse >
Coding School

Interfaces

As interfaces são padrões definidos através de contratos ou especificações.



DigitalHouse > Coding School

Relacionamento – implementação

Implementa

Implementa

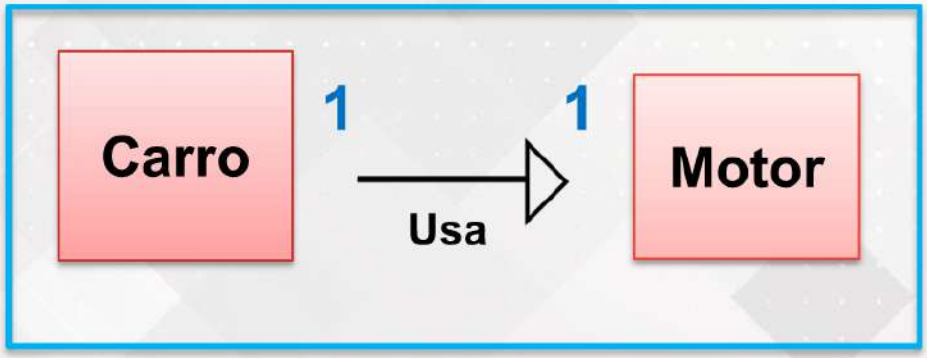
DigitalHouse > Coding School

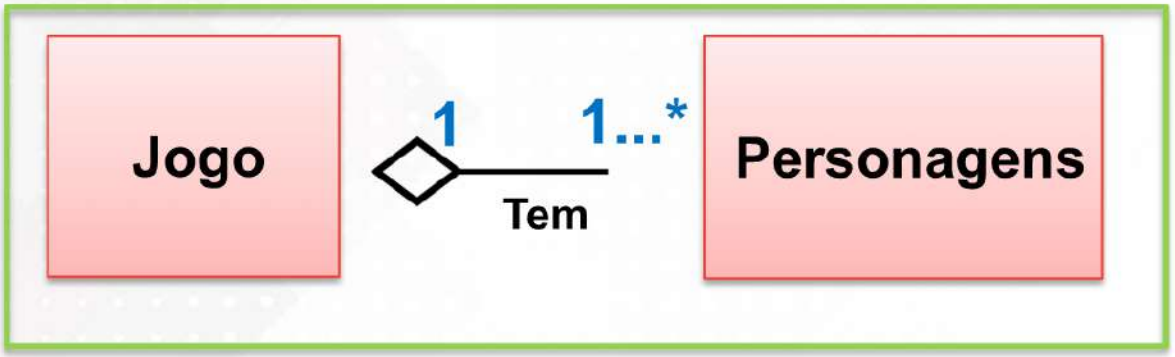
Multiplicidade

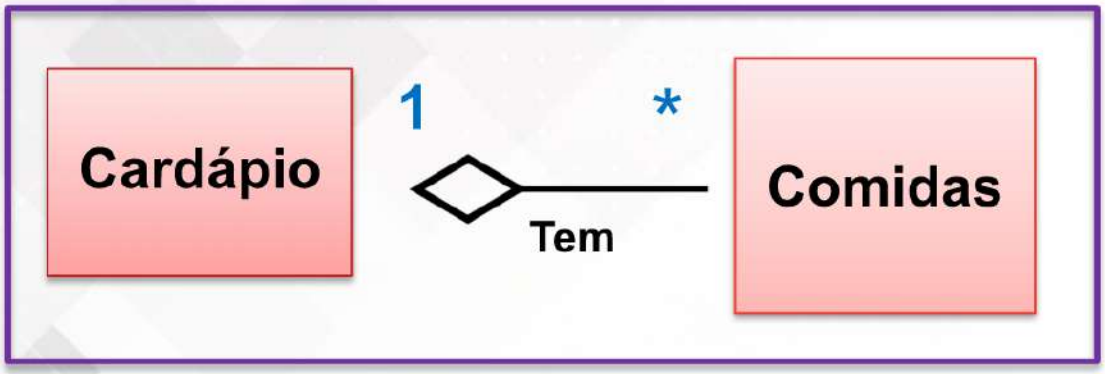
A quantidade de objetos que poderia ter dentro de uma Classe

Um para um ---------- 1..1
Um para vários ---------- 1...*
Um para um ou vários ---------- 1...1*

Multiplicidade - Exemplos





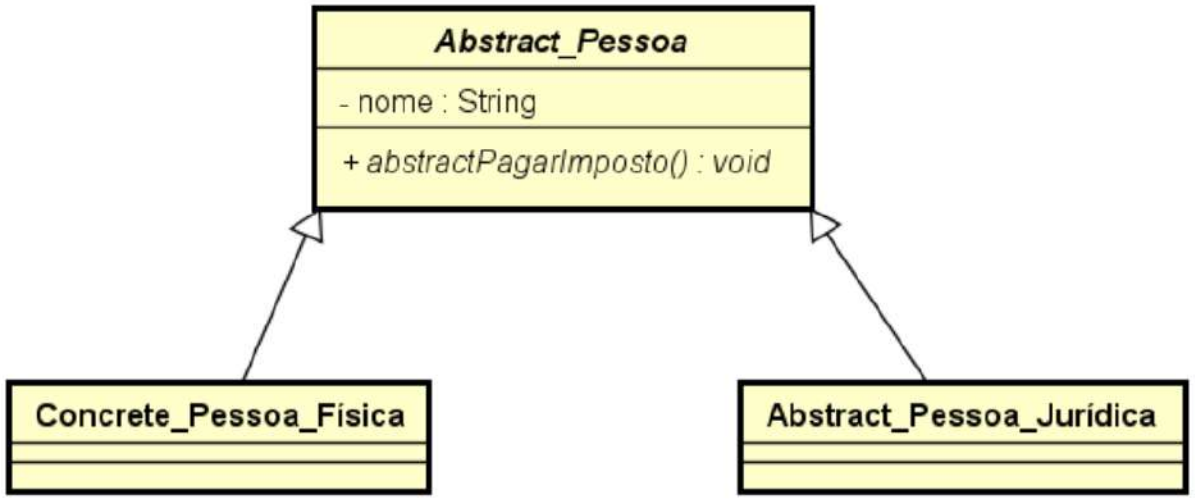


Exercícios

DigitalHouse >
Coding School

Classes abstratas

As classes abstratas servem como "modelo" para outras classes que herdam dela. Uma classe abstrata não pode gerar instâncias.



Coesão

Uma classe faz apenas o que for de sua responsabilidade e nada mais.

Coesão

Baixa Coesão

- Várias funcionalidades em um mesmo objeto;
- Difícil reaproveitamento;
- Difícil manutenção;
- Alta Complexidade.

Alta Coesão

- Objetos devem fazer apenas uma(e bem feita) tarefa;
- Maior capacidade de reaproveitamento;
- Facilidade de manutenção.

Exercício: Passando seu UML para o colega

DigitalHouse > Coding School