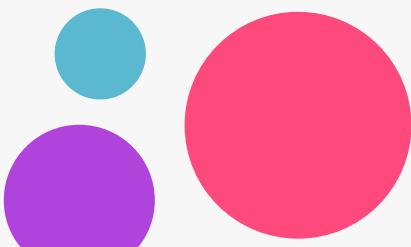


# Paradigma orientado a objetos

Aula 06 - Programação orientada a objetos

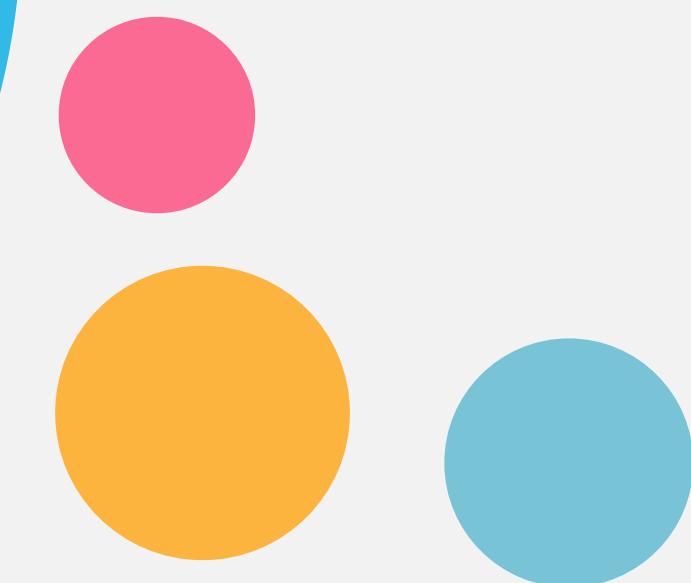
• • •

Professor Daniel Marques



1

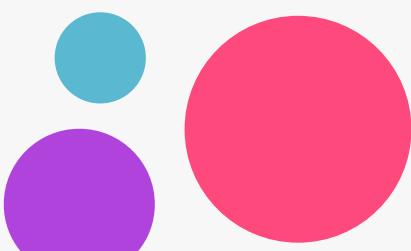
Paradigma orientado  
a objetos



# O que é um paradigma?

*Pa·ra·dig·ma*

- 1) **Algo que serve de exemplo geral ou de modelo;**
- 2) **Conjunto das formas que servem de modelo de derivação ou de flexão;**
- 3) **Conjunto dos termos ou elementos que podem ocorrer na mesma posição ou contexto de uma estrutura.**

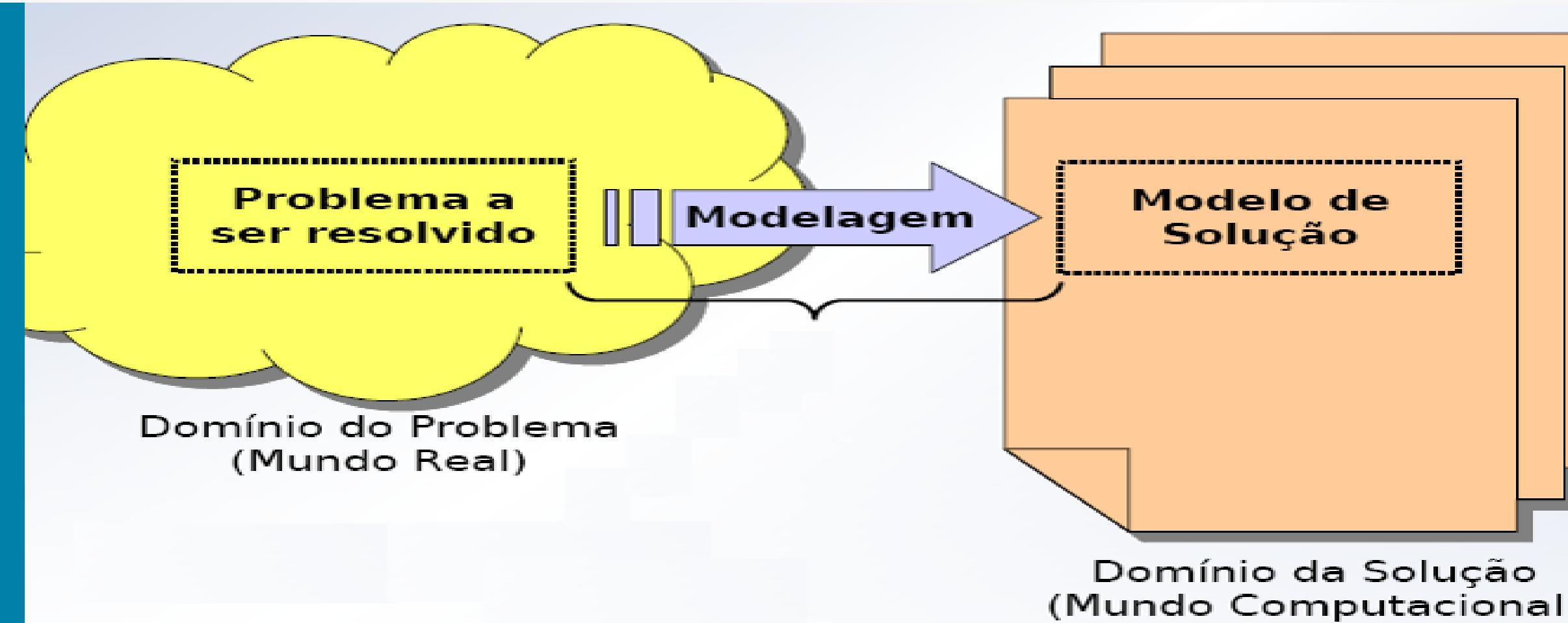


# Paradigmas de desenvolvimento de software

Maneiras diferentes de pensar em resolver os problemas através de sistemas computacionais.

Exemplos:

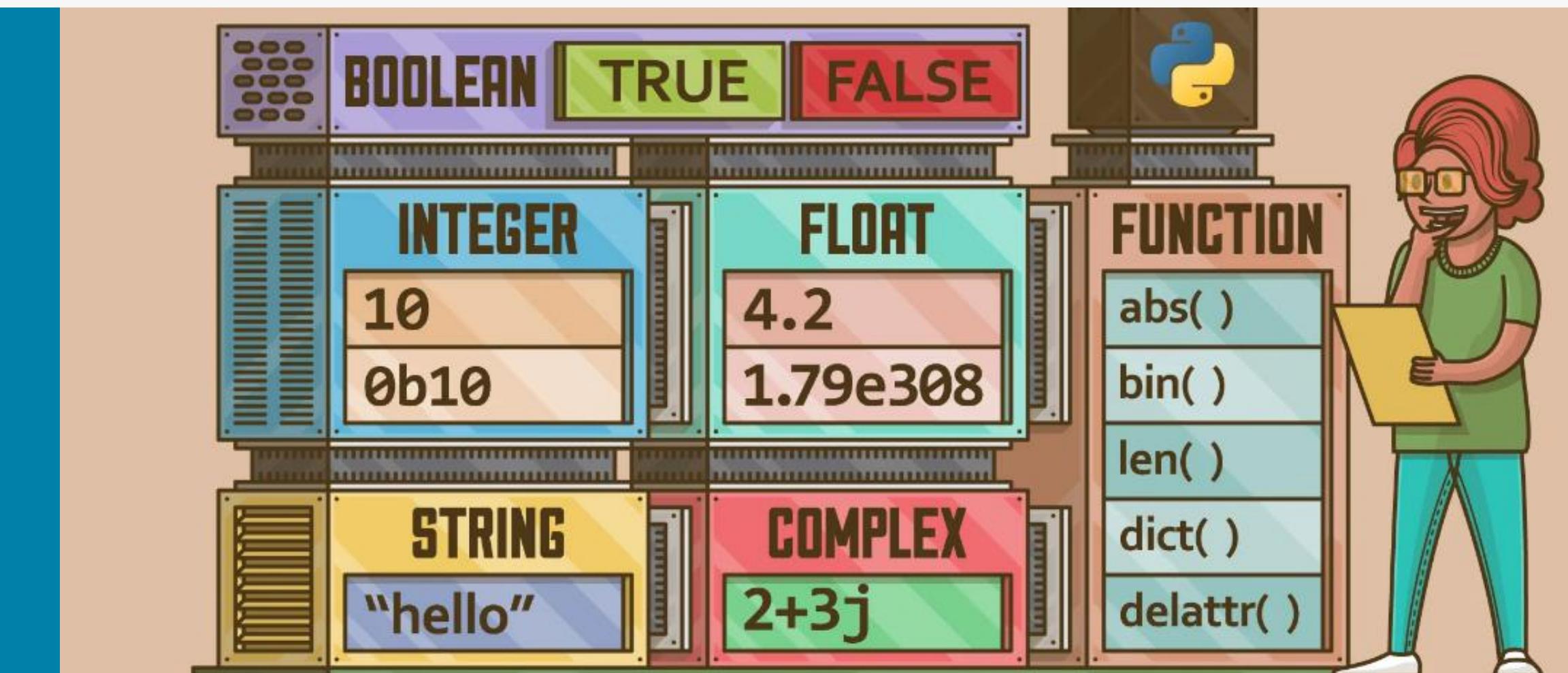
- Estruturado (funções e módulos)
- Orientado a objetos



# Paradigma estruturado

## Linguagens imperativas

- Programa é um conjunto de funções;
- Fazem clara distinção entre funções e dados;
- Funções, a princípio, são ativas e tem comportamento;

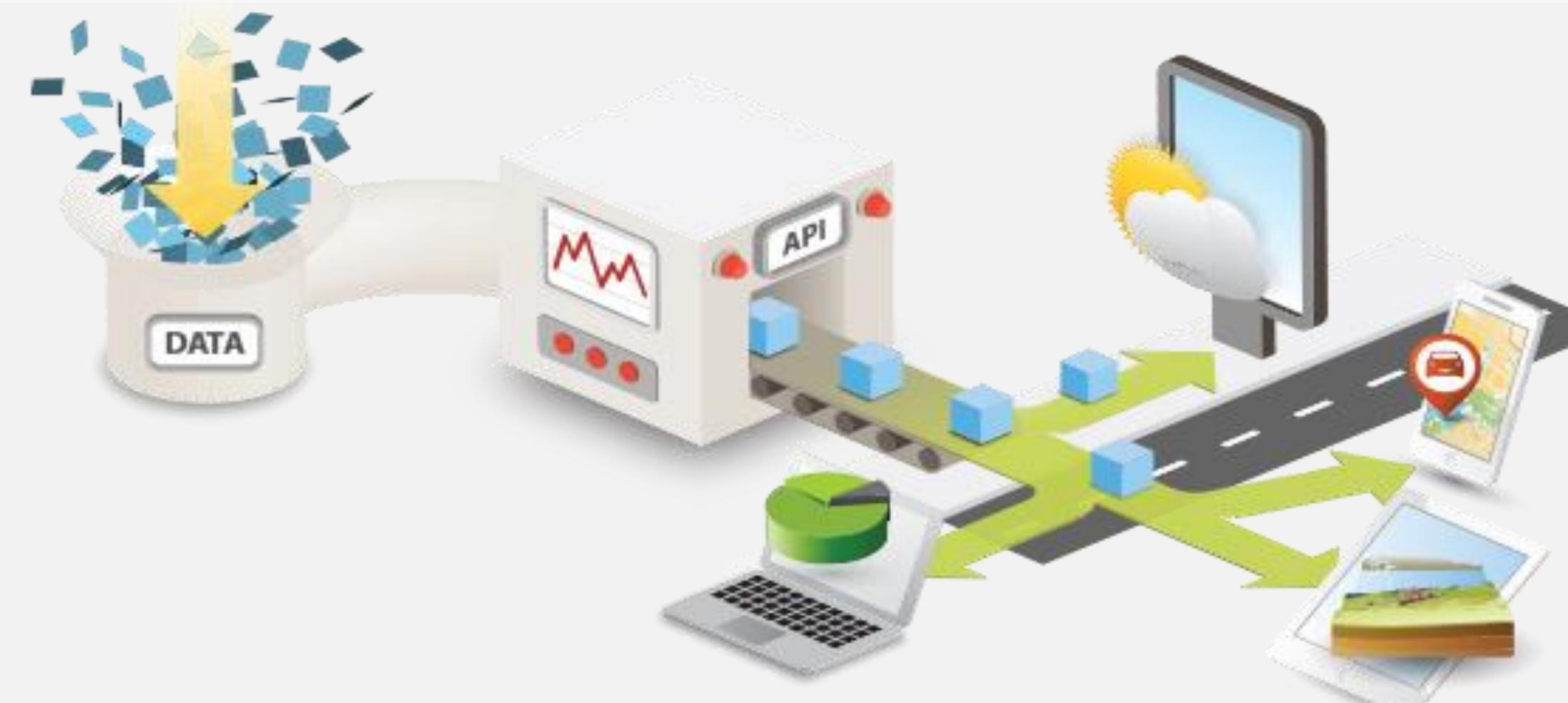


- Dados são repositórios passivos de informação, afetados por funções
- Dados (variáveis) são usados para apoiar as funções;

# Paradigmas de desenvolvimento de software

Programas processam dados

- Entrada
- Processamento
- Saída



# Paradigmas de desenvolvimento de software

O **paradigma de programação orientado a objetos** considera que os dados a serem processados e os mecanismos de processamento destes dados devem ser considerados em conjunto;



A criação de modelos que representam conjuntamente dados e operações nestes dados é a solução

# Estruturado x Orientado a objetos

*Exemplo: Agenda*



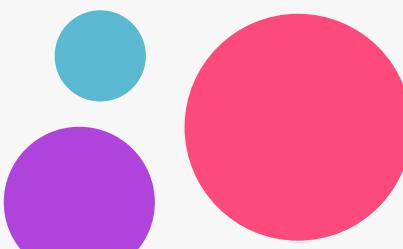
## Paradigma Estruturado

- **Variáveis**
  - **Vetor de nomes**
  - **Vetor de endereços**
  - **Vetor de telefones**
- **Funções**
  - **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**



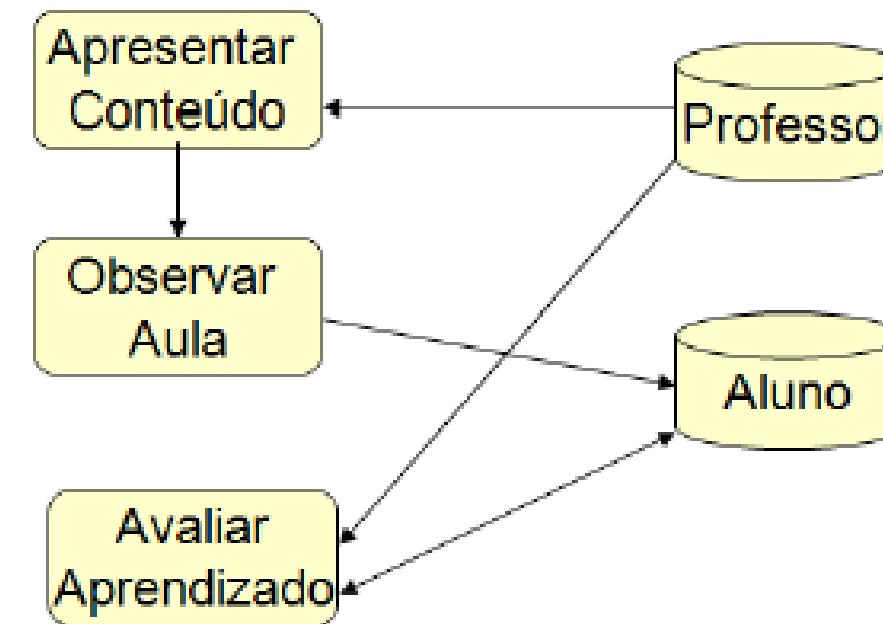
## Paradigma Orientado à Objetos

- |  |   |
|--|---|
| <b>Objeto Agenda</b> <ul style="list-style-type: none"><li>• <b>Atributos</b><ul style="list-style-type: none"><li>▪ <b>Vetor de Contatos</b></li></ul></li><li>• <b>Métodos</b><ul style="list-style-type: none"><li>• <b>Listagem de Contatos</b></li><li>• <b>Adição de um Contato</b></li><li>• <b>Remoção de um Contato</b></li></ul></li></ul> | <b>Objeto Contato</b> <ul style="list-style-type: none"><li>• <b>Atributos</b><ul style="list-style-type: none"><li>▪ <b>Nome</b></li><li>▪ <b>Endereço</b></li><li>▪ <b>Telefone</b></li></ul></li><li>• <b>Métodos</b><ul style="list-style-type: none"><li>• <b>Exibição do nome, endereço e telefone</b></li><li>• <b>Edição de nome, endereço e telefone</b></li></ul></li></ul> |
|--|---|

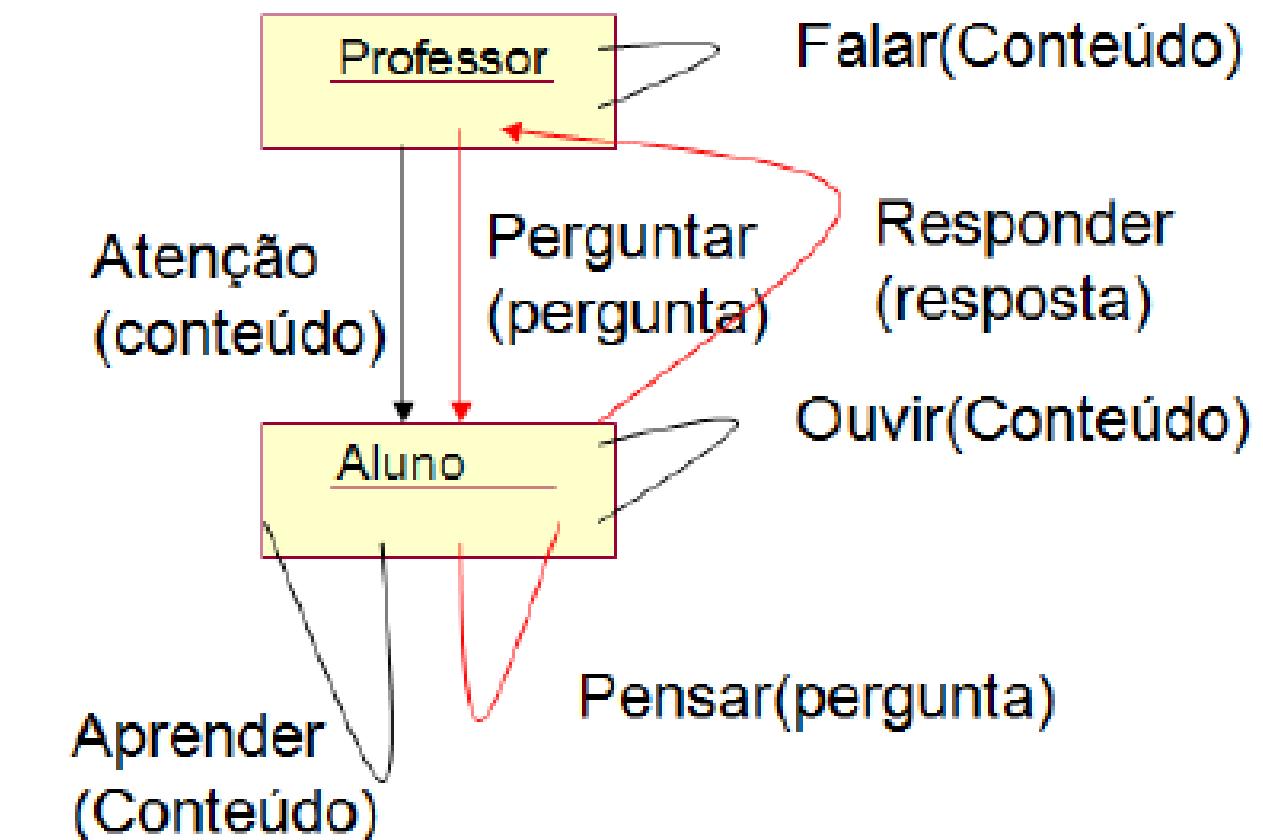


# Estruturado x Orientado a objetos

Processo de ensinar com o  
paradigma estruturado e  
com o paradigma  
orientado a objeto



Paradigma Estruturado

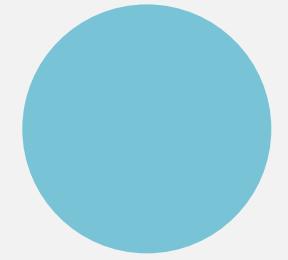
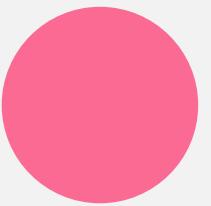


Paradigma orientado a objetos



2

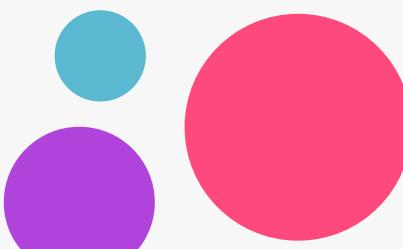
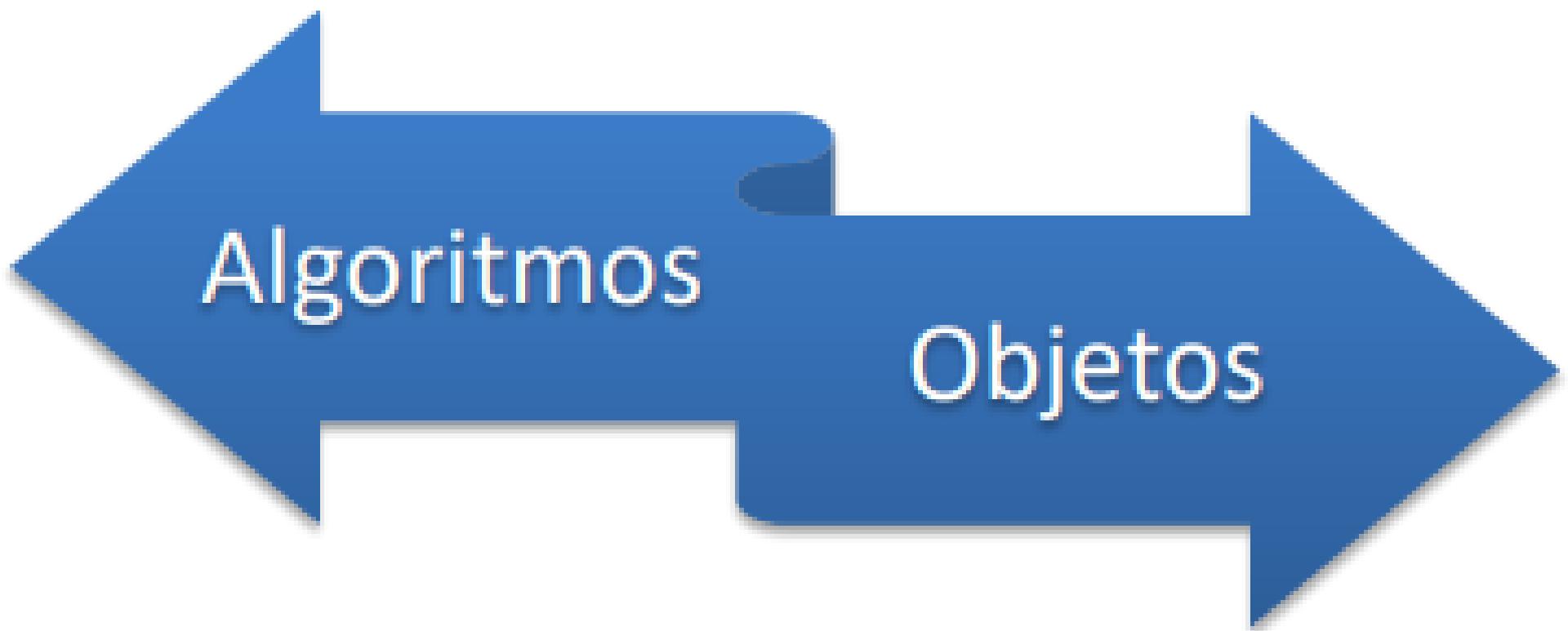
Objetos



# Paradigma Estruturado x Orientado a Objetos

---

- O paradigma estruturado organiza o programa em termos de algoritmos
- O paradigma orientado à objetos organiza o programa em termos de objetos



# Algoritmos X Objetos

*Podemos criar programas pensando em termos de objetos ao invés de algoritmos?*



## O mundo é composto por objetos

- Uma loja tem produtos, pedidos, estoque;
- Um restaurante tem mesas, garçons, comida;
- Uma rodoviária tem ônibus, passageiros, bagagens

**E se criarmos programas  
basicamente criando objetos?**



- Objetos equivalentes ao mundo real;
- Fazer com que esses objetos se comuniquem

```
1 public class Aluno {  
2     private Double nota;  
3     private String nome;  
4     private Integer idade;  
5     public Aluno(String nome, Integer  
6         nota) {  
7         this.nome = nome;  
8         this.idade = idade;  
9     }  
10    //Getters e Setter aqui...  
11}
```

# O que são objetos?

## Definição

- Um objeto é a representação computacional de um elemento ou processo do mundo real
- Cada objeto possui suas características e seu comportamento.



### Características ou Atributos

- Nome
- Peso
- Altura

### Comportamento ou Métodos (funções)

- Falar
- Andar
- Comer

## Exemplos de objetos

- Cadeira
- Carro

- Cliente
- Aluno
- Aula

- Lápis
- Avião
- Cliente

# Características de um objeto



## Definição

- Uma característica descreve uma propriedade de um objeto
- Cada característica é chamada de **atributo** e funciona como uma **variável** pertencente ao objeto

Exemplo de características para o objeto carro

**CLASSE CARRO**



**OBJETO**



**CELTA**

**OBJETO**



**CORSA**

**OBJETO**



**FIAT UNO**

Características de um carro

- Cor
- Marca

- Número de portas
- Ano de fabricação
- Tipo de combustível

# Comportamento de objetos



## Definição

- Um comportamento representa uma **ação ou resposta** de um objeto a uma ação do mundo real;
- Cada comportamento é chamado de **método** e funciona como uma **função** pertencente ao objeto

Exemplos de comportamento para o objeto carro

CLASSE CARRO	OBJETO CARRO A	OBJETO CARRO
Atributos de objeto	Marca	Ford
	Modelo	Fiesta
	Cor	branco
	Combustível	gasolina
Métodos	ligar	
	acelerar	
	frear	

- Exemplos de comportamento:
- Frear
  - Virar para a direita
  - Virar para a esquerda
  - Ligar

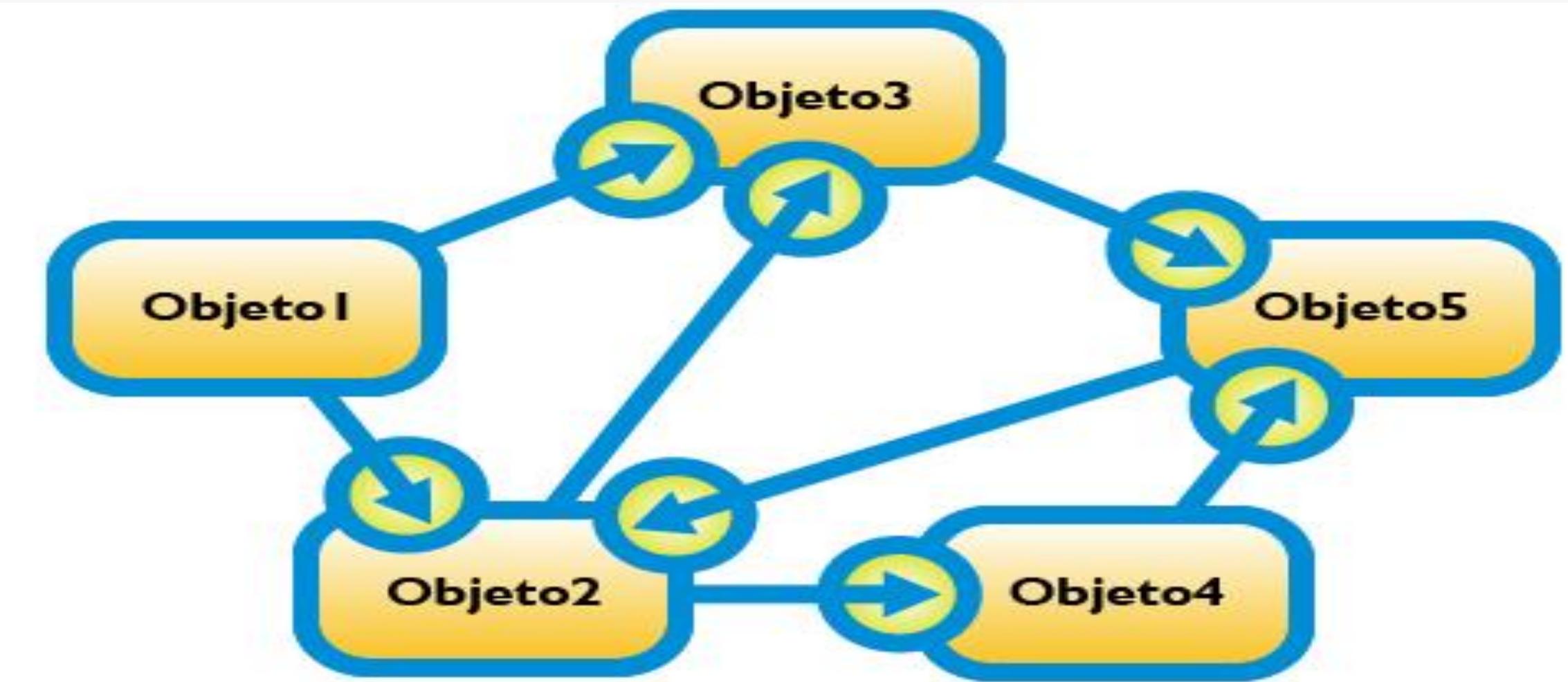


3

Motivação

# Motivação da programação orientada a objetos

A programação orientada a objetos, se inspira nessas ideias para modelar os programas como um conjunto de “objetos” que se relacionam!



- Tem como principal objetivo facilitar a modelagem e desenvolvimento de sistemas, através da interação entre objetos.
- Sua principal vantagem é a proximidade com a forma que os seres humanos visualizam e entendem o mundo ao seu redor.

# Objeto no Mundo Real

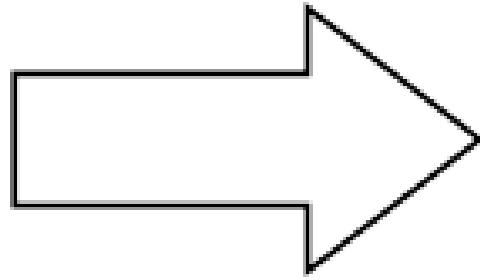
Características

Comportamento

# Objeto Computacional

Atributos

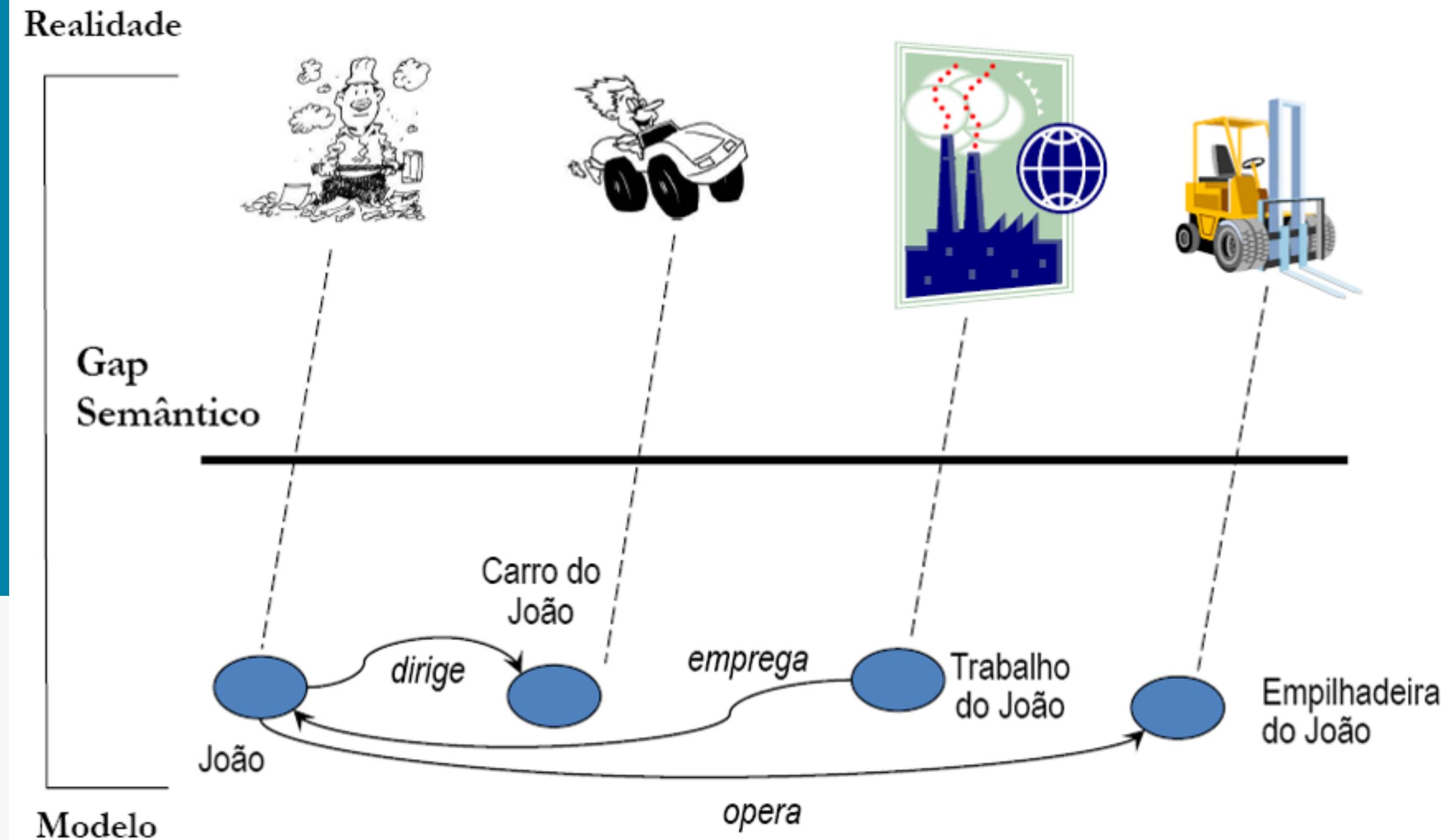
Métodos



Mapeamento de Objetos

# Motivação da programação orientada a objetos

Um dos objetivos da engenharia de software é diminuir essa “diferença semântica”





# Motivação da programação orientada a objetos

---

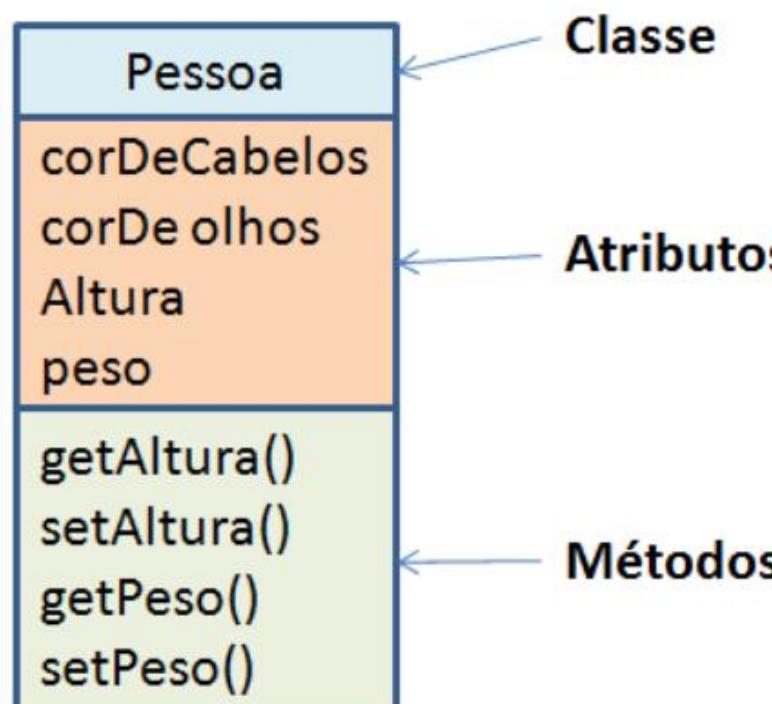
É um **paradigma de programação** onde se usam **classes** e **objetos** criados para representar e processar dados.

Os **modelos** na programação orientada objetos são chamadas de **classes**

Classes são estruturas que contém um determinado modelo.

- Os dados devem ser representados
- As operações devem ser efetuadas entre esses dados

# Classes e objetos



## Classes

Os modelos na POO são chamados de classes

## Objetos



Objetos são instâncias de classes



# Motivação da programação orientada a objetos

---

## Facilidade

Facilidade maior para modelagem de objetos do mundo real

## Necessidade

Necessidade de construção de softwares mais complexos

## Extensibilidade

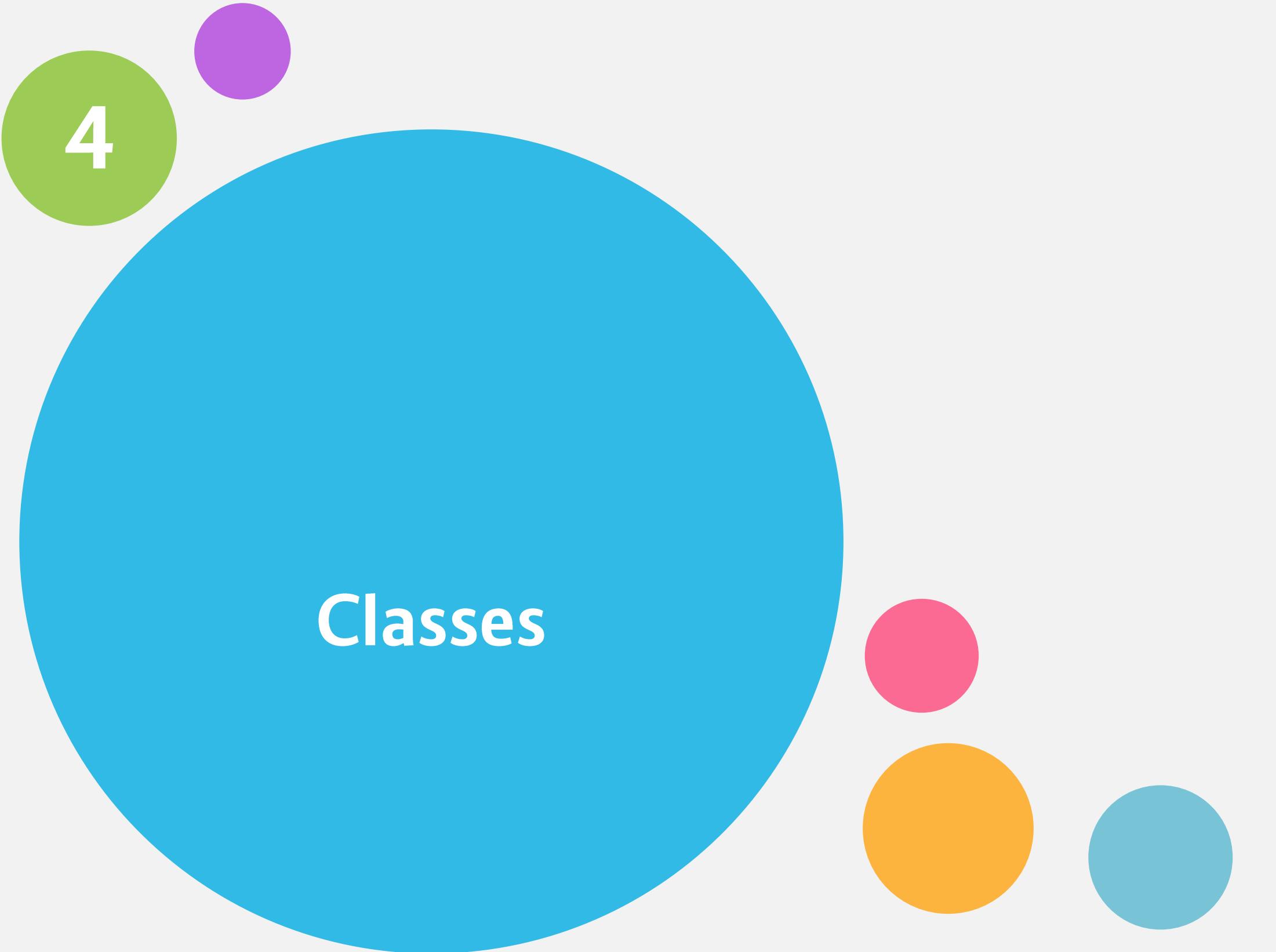
Maior extensibilidade: facilidade de incrementar funcionalidades

## Manutenção

Facilidade de manutenção

## Reuso

Maior reuso





# Classes x Objetos

---



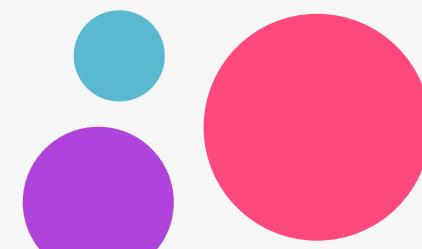
## Valores têm tipos primitivos

- 123 é um valor inteiro;
- True é um valor booleano;
- 12.3 é um valor real.



## Objetos pertencem a classes

- Zé, João e Severino são da classe Pessoa;
- Fusca e Kombi e Chevette são da classe Carro;
- Campinense, Treze e Perilima são da classe Time



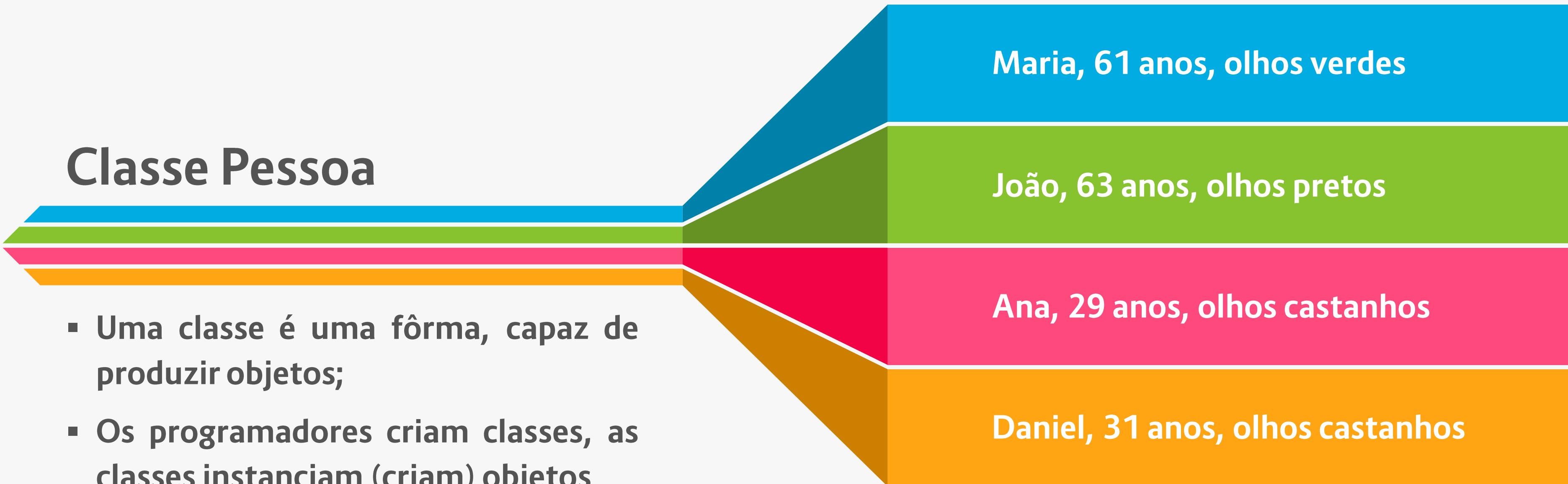


# Classes x Objetos

---

## Classe Pessoa

- Uma classe é uma forma, capaz de produzir objetos;
- Os programadores criam classes, as classes instanciam (criam) objetos



Maria, 61 anos, olhos verdes

João, 63 anos, olhos pretos

Ana, 29 anos, olhos castanhos

Daniel, 31 anos, olhos castanhos



# Classes

---

A classe descreve as características e comportamentos dos objetos

Cada objeto pertence a uma única classe

O objeto possuirá os atributos e métodos definidos na classe.

O objeto é chamado de instância de sua classe

A classe é o bloco básico para construção de programas orientados à objetos



# Comunicação entre objetos

*Exemplo: Compra no supermercado*

Quais os objetos participantes do cálculo do total da compra?

Pedido: 12345  
Cliente: João da Silva  
Endereço: Rua dos Bobos, número zero

Item	Produto	Preço	Quantidade	Subtotal
1	Açúcar	R\$ 2,00	5	R\$ 10,00
2	Macarrão	R\$ 2,50	2	R\$ 5,00
3	Feijão	R\$ 3,00	3	R\$ 9,00
TOTAL				R\$ 24,00

# Comunicação entre objetos

*Exemplo: compra no supermercado*

Pedido: 12345

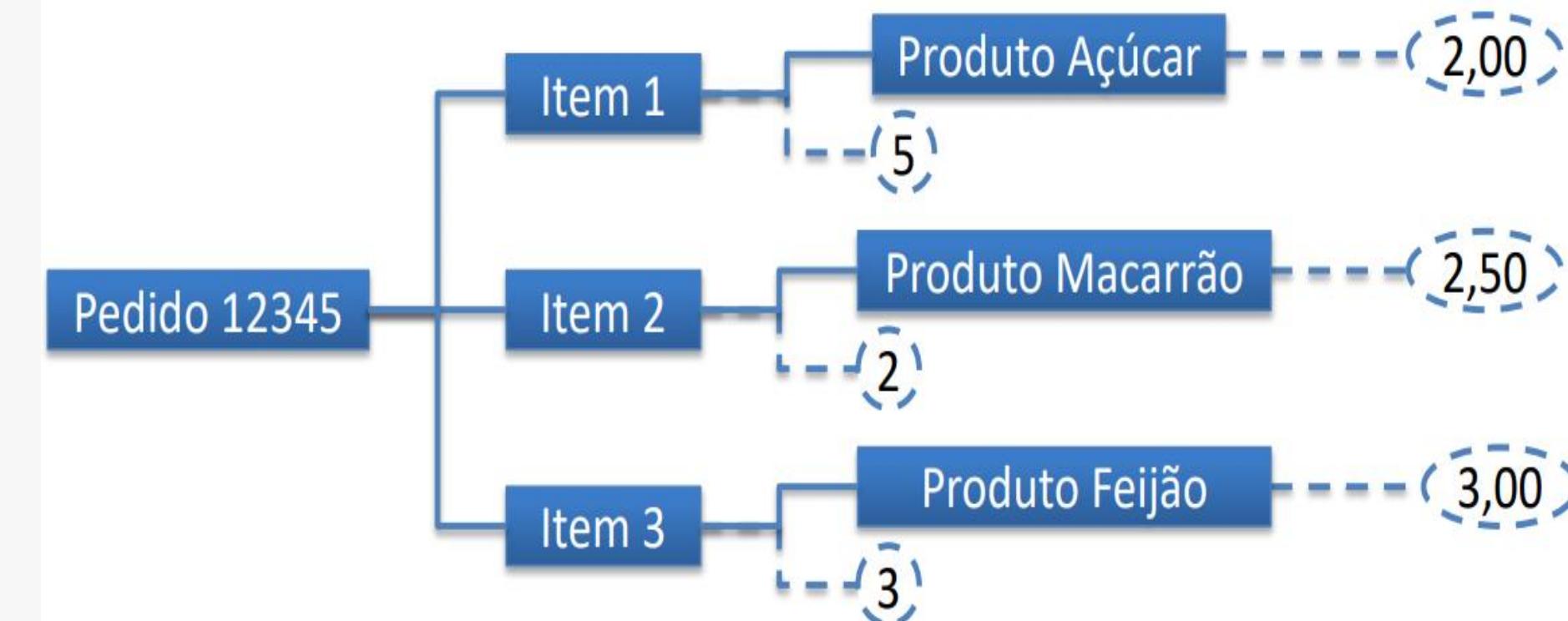
Cliente: João da Silva

Endereço: Rua dos Bobos, número zero

Item	Produto	Preço	Quantidade	Subtotal
1	Açúcar	R\$ 2,00	5	R\$ 10,00
2	Macarrão	R\$ 2,50	2	R\$ 5,00
3	Feijão	R\$ 3,00	3	R\$ 9,00
TOTAL				R\$ 24,00

Tabela das compras

## Comunicação entre objetos

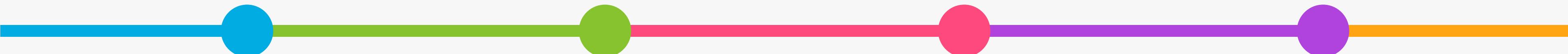


# Comunicação entre objetos

O objeto Caixa pediria ao objeto Pedido seu valor total

**Caixa -> pedido**

**Passo 1**



**Passo 2**

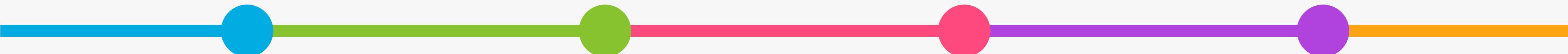
**Pedido -> item**

O objeto Pedido, por sua vez, percorreria todos os seus objetos Item perguntando seu valor subtotal e somaria esses valores para responder ao objeto Caixa

Cada objeto Item perguntaria ao objeto Produto, o seu preço e multiplicaria esse preço pela quantidade que está sendo comprada, para responder ao objeto Pedido

**Item -> produto**

**Passo 3**

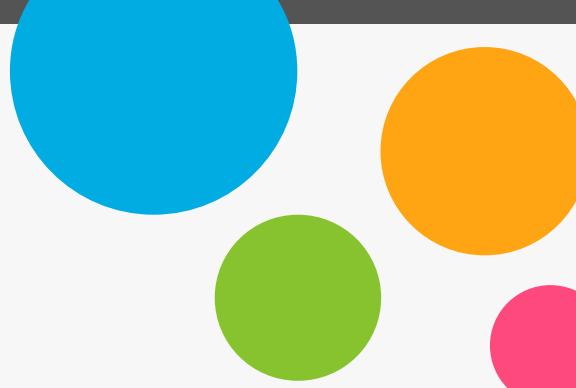


**Passo 4**

**Produto <- item**

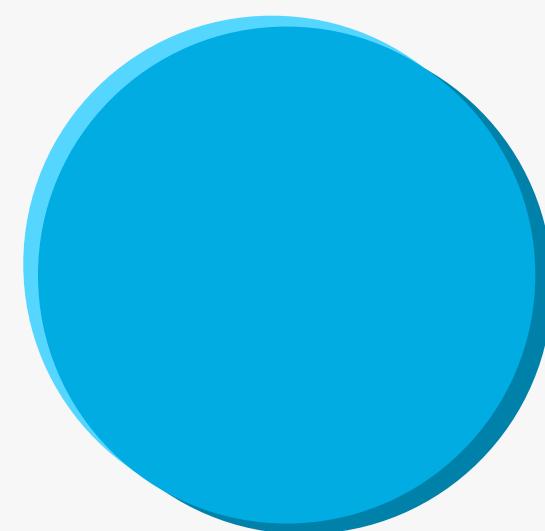
**Item <- pedido**

**Pedido <- caixa**



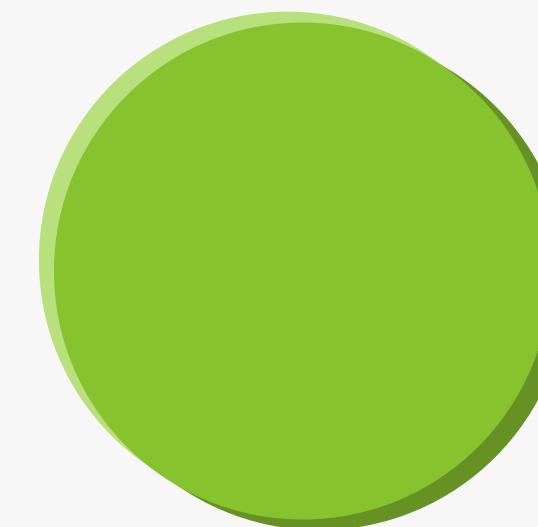
# Princípios da orientação a objetos

---



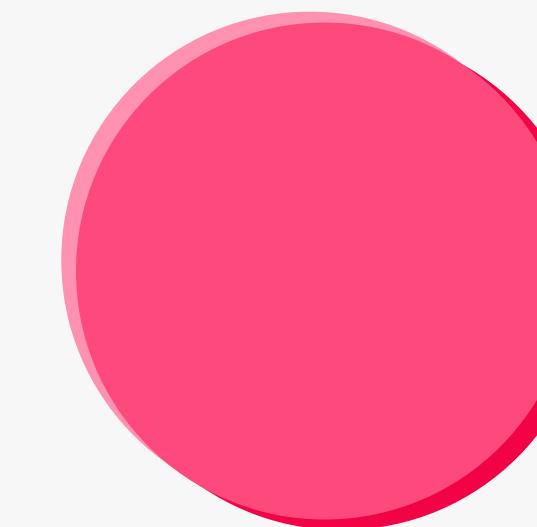
Abstração

---



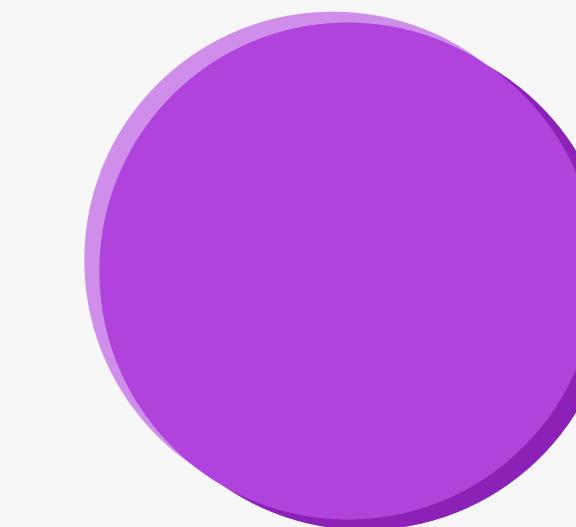
Encapsulamento

---



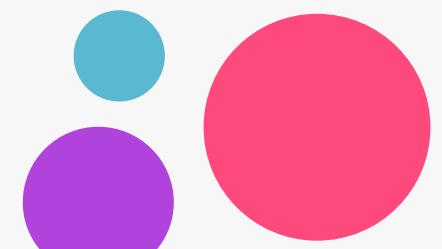
Modularidade

---



Hierarquia

---





5

Abstração



# Abstração

---

As pessoas tipicamente tentam compreender o mundo, construindo modelos mentais de partes dele.

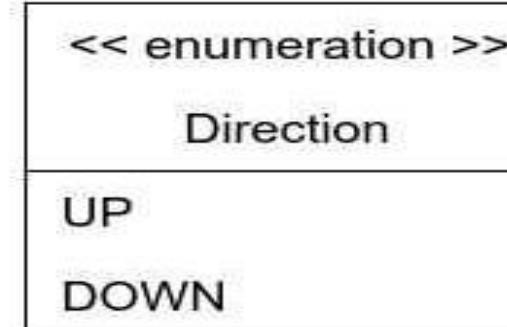
Um modelo mental abstrai apenas as características relevantes de um sistema para seu entendimento

# Abstração

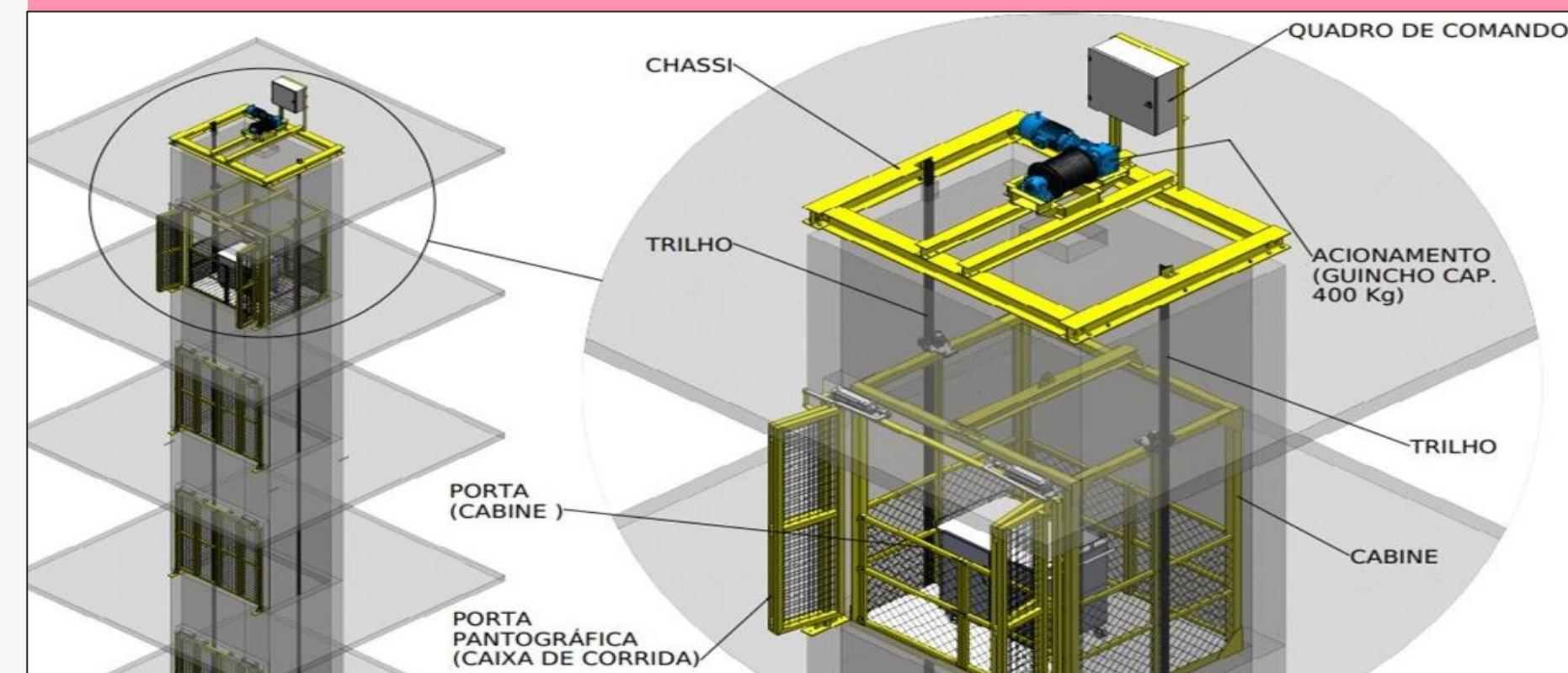
*Exemplo de abstração de um elevador, por um engenheiro civil e um analista de sistema*

## Elevator

```
- direction : Direction  
- requests : vector<int>  
- min_floor : int  
- max_floor : int  
- current_floor : int  
- passengers : size_t  
- capacity : size_t  
  
+ Elevator(int min_floor, max_floor, size_t capacity)  
+ start_elevator() : void  
- set_request() : void  
- check_request(int floor) : int
```



## Engenheiro Civil



## Analista de sistema

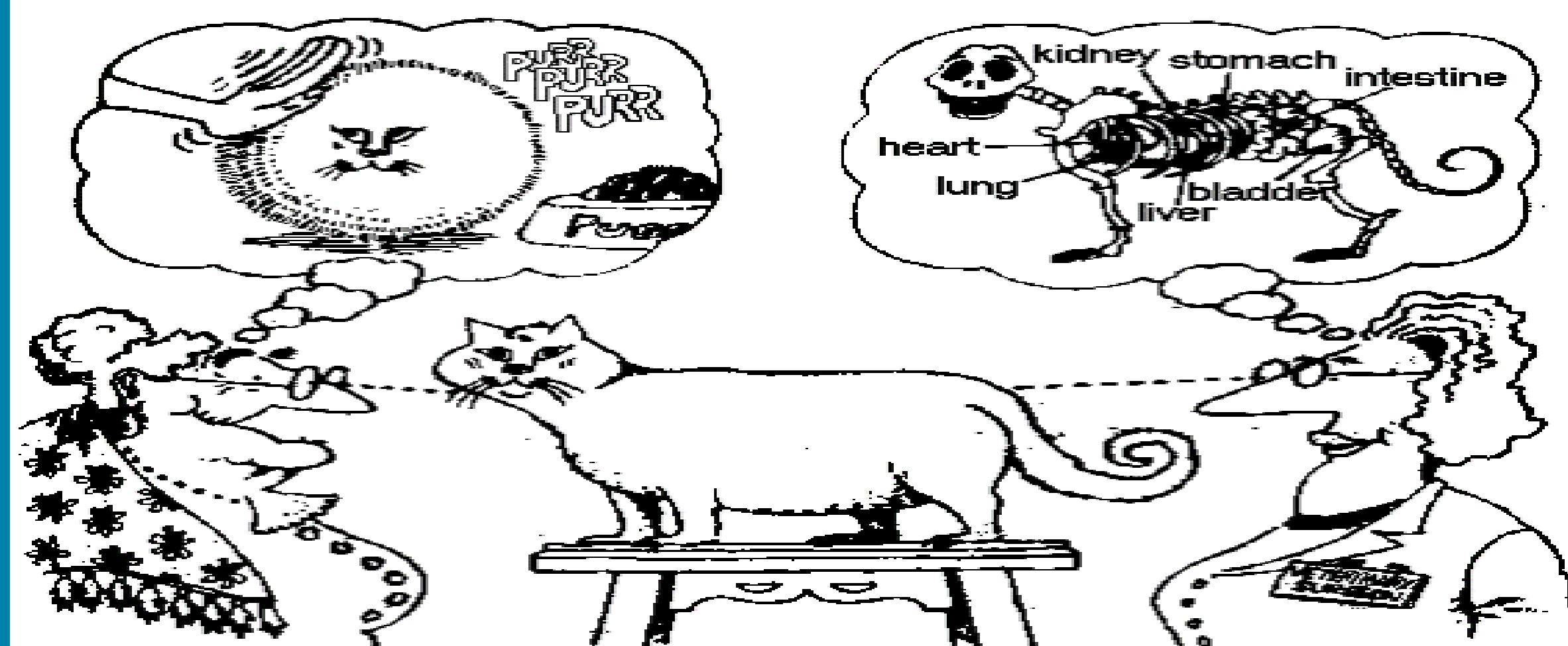
Quais as características que devem ser consideradas: estado (subindo, descendo, parado), porta aberta ou fechado, qual andar que o elevador esta, indicar o andar atual no painel. Abrir porta, Fechar porta, subir, descer, parar, atualizar indicador de andar.

- Qual a Marca?
- Qual a capacidade?
- Qual a Velocidade?
- Indicado para que tipo de edifício?
- Qual a largura?
- Qual a Altura?

# Abstração

*Mas o que é abstração?*

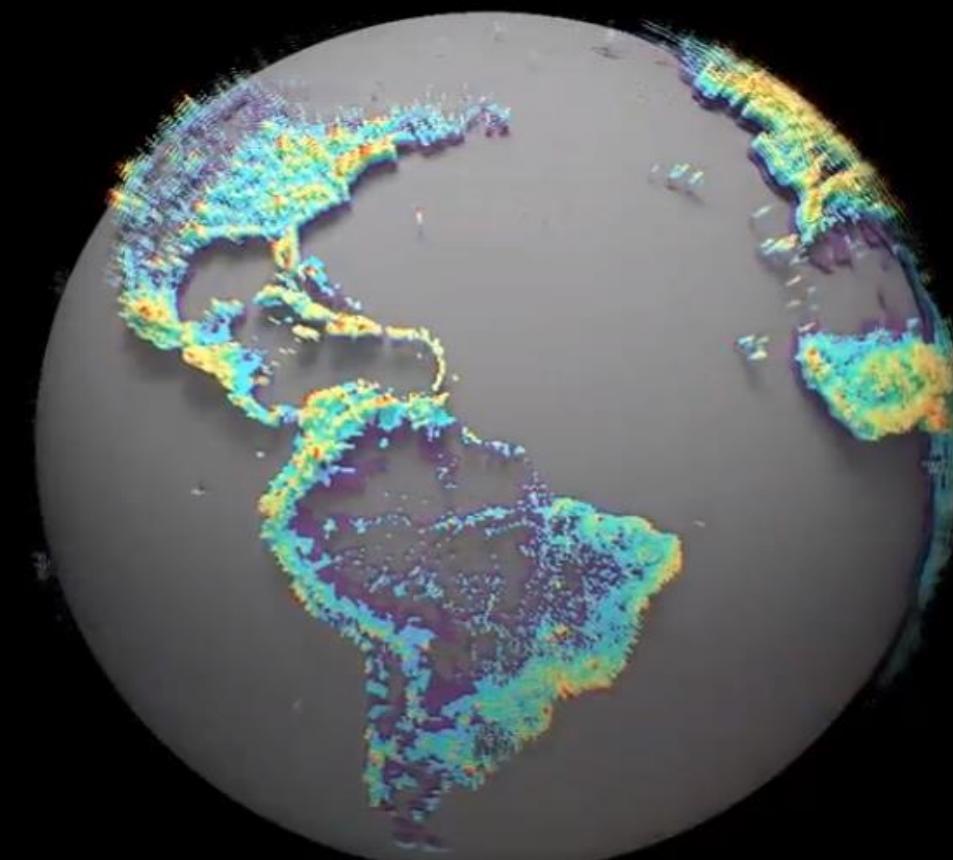
Descrição de um problema ou um objeto a partir da identificação de suas características principais.



# Abstração

Técnica para lidar com a complexidade de um problema.

The Humanity Globe: World Population Density, 30km<sup>2</sup> Grid



- Destaca os aspectos importantes do objeto real abstraído, segundo perspectiva do observador
- Ignora os detalhes não relevantes para o observador

# Abstração

*Abstração de uma casa*



**Planta de uma casa**

**Maquete de uma casa**



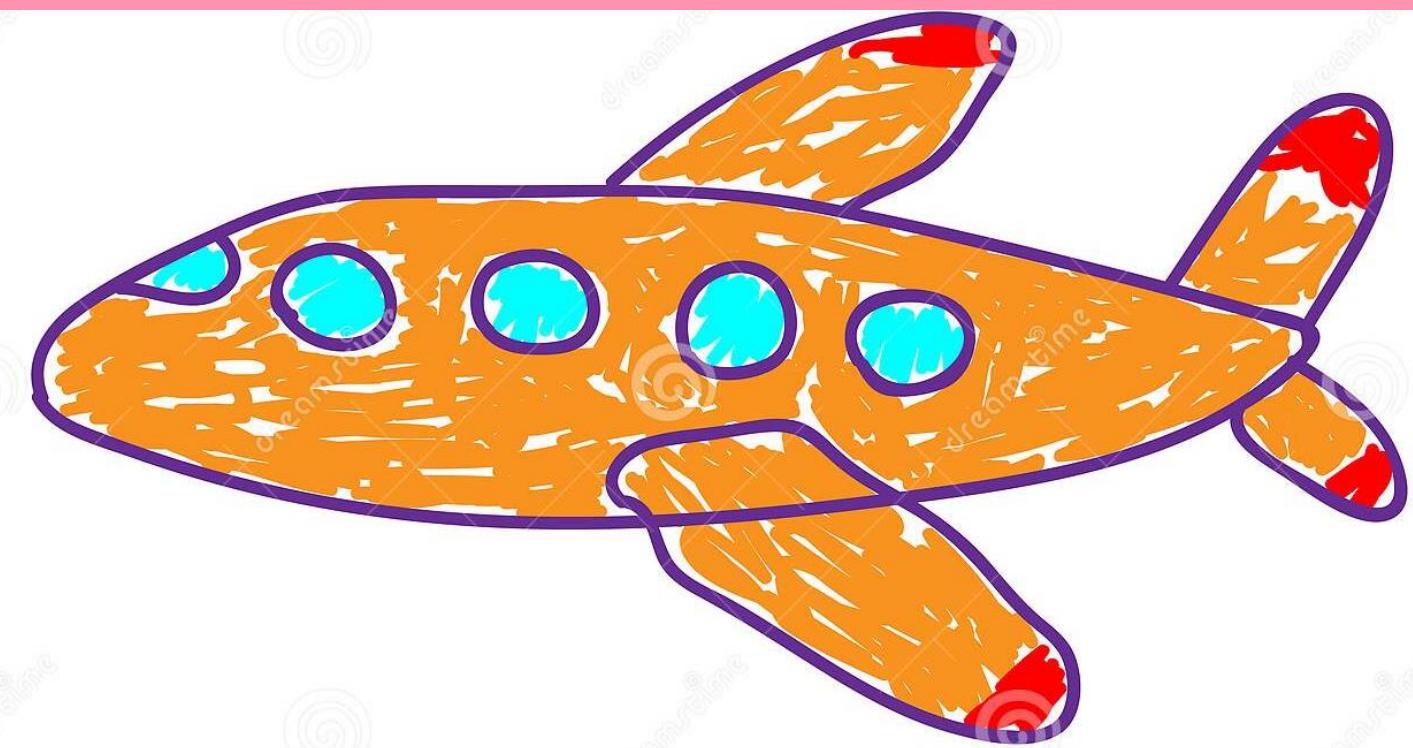
# Abstração



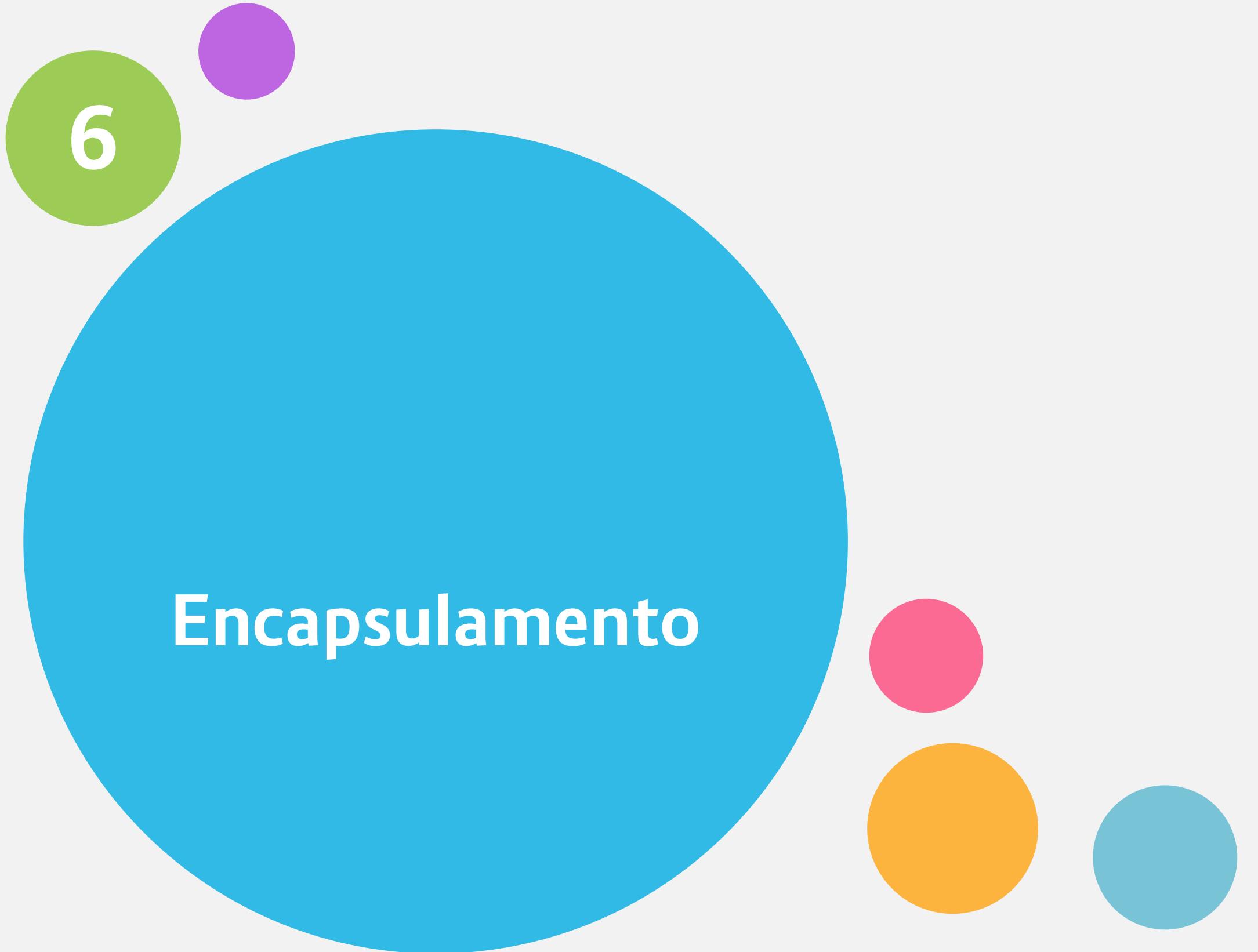
## Concreto

- Todos os detalhes da realidade
- Quanto mais concreto, mais específico
- O avião acima é tão específico, com tantos detalhes, que só representa ele mesmo.

## Abstrato



- Só características importantes
- Quanto mais abstrato mais geral.
- A figura acima poderia representar qualquer avião



6

Encapsulamento

# Encapsulamento

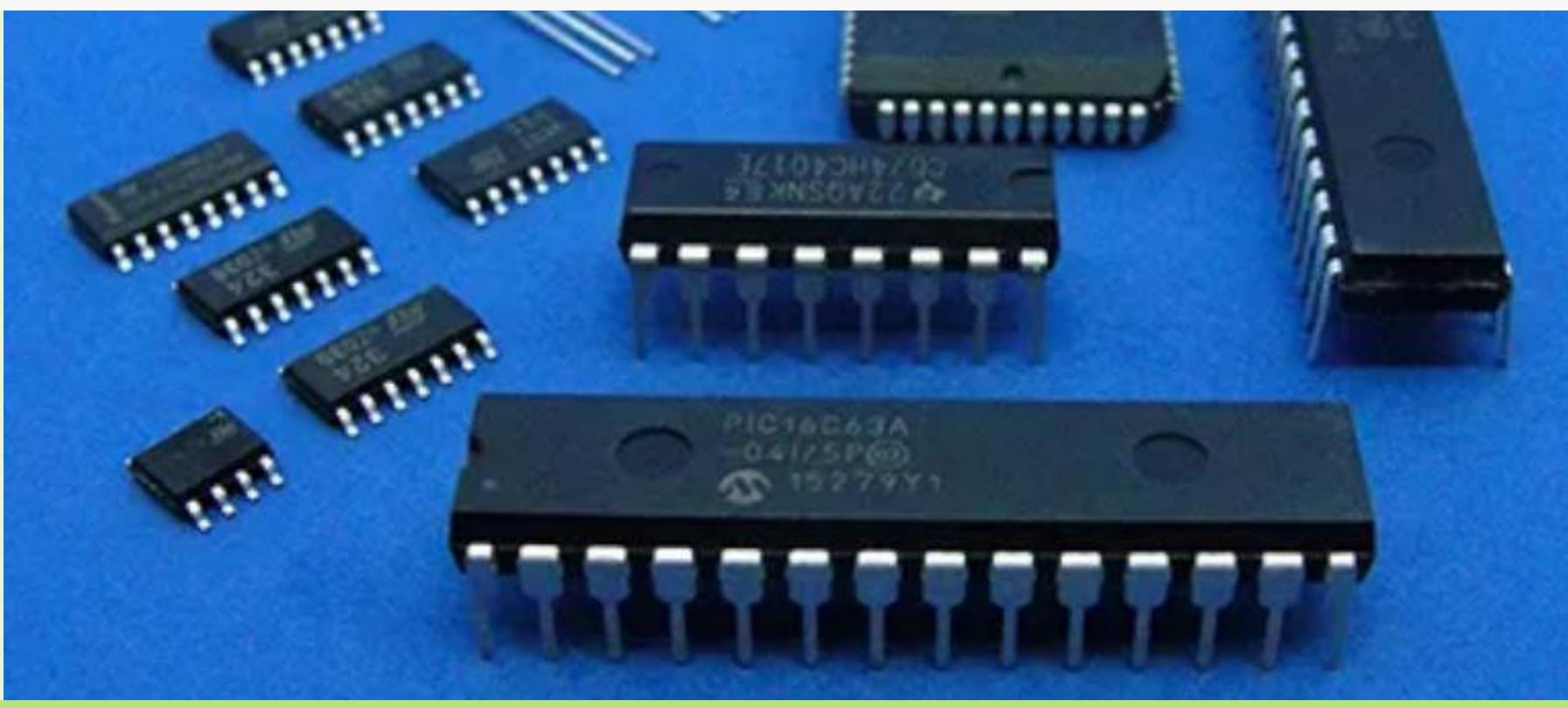
Omissão dos detalhes sobre o funcionamento interno de um objeto ou sistema.



Antigamente, o teor alcoólico do Biotônico Fontoura era de 9,5%.  
A cerveja tem um teor alcoólico entre 4 e 10%.

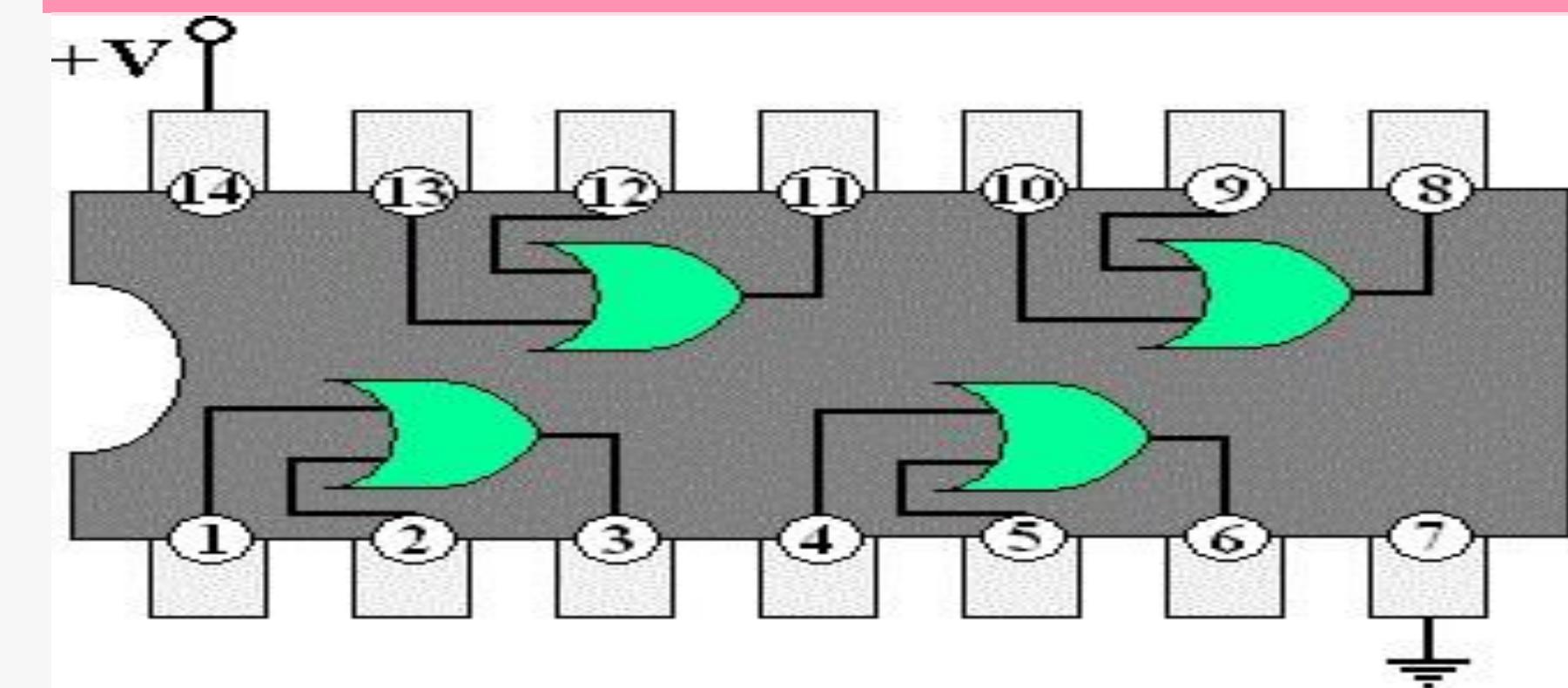
# Encapsulamento

*Exemplos*



Encapsulamento do circuito integrado

Arquitetura do circuito integrado



# Encapsulamento

## *Exemplos*



### Encapsulamento da pele

Alguma dúvida sobre os benefícios do encapsulamento humano através da pele?

### Encapsulamento de uma CPU



# Encapsulamento

## *Exemplos*

Encapsulamento  
comprometido!

Hummm, tô vendo como ele  
foi implementado! Não é  
humano!



# Encapsulamento

O importante é saber o que o objeto faz e não como ele faz



# Encapsulamento

Encapsular é esconder como as coisas funcionam por trás de uma interface externa



Ex: caixa automático

- Como ele é implementado internamente?
- Utilizamos através de operações bem conhecidas

# Encapsulamento



## Ideia de uma “caixa preta”

**Em muitos casos, seria desejável que os dados não possam ser acessados ou usados diretamente, mas somente através das operações**



# Encapsulamento

---



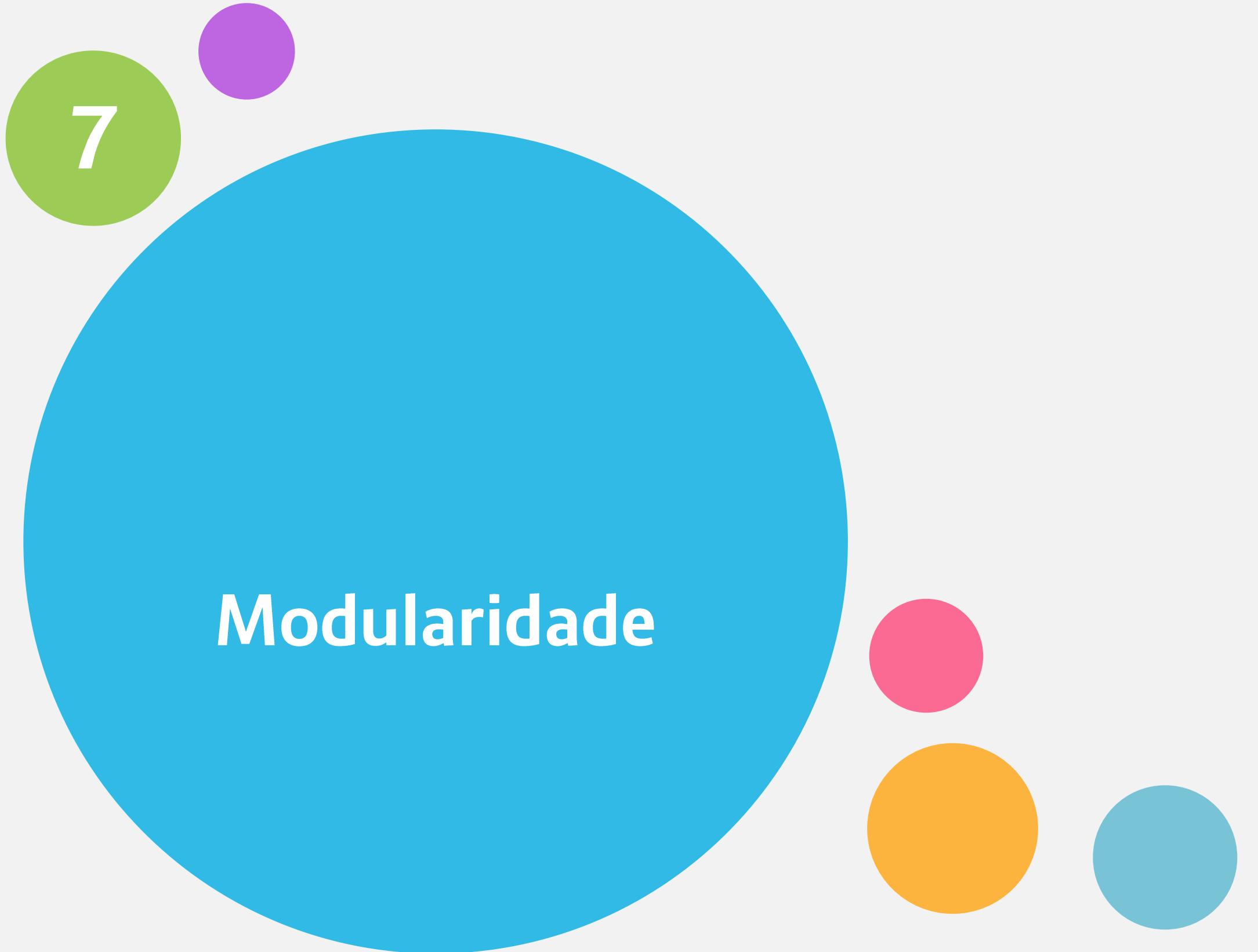
**Grady Booch**

- “A habilidade de mudar a representação de uma abstração sem perturbar quaisquer de seus clientes é o principal benefício do encapsulamento”
- “O encapsulamento não impede o programador de fazer coisas estúpidas”



**Bjarne Stroustrup**

“O encapsulamento previne acidentes, não fraudes”

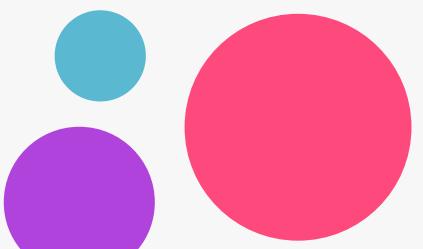
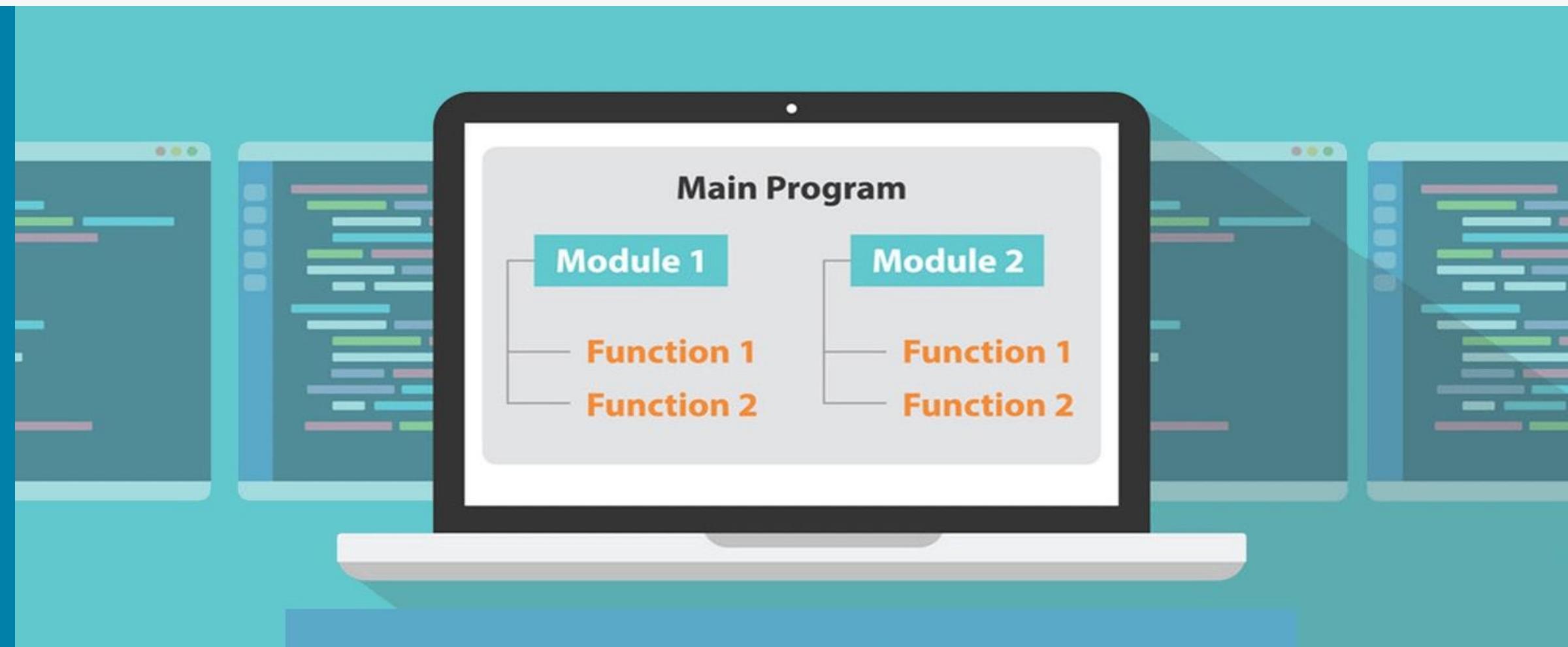


7

Modularidade

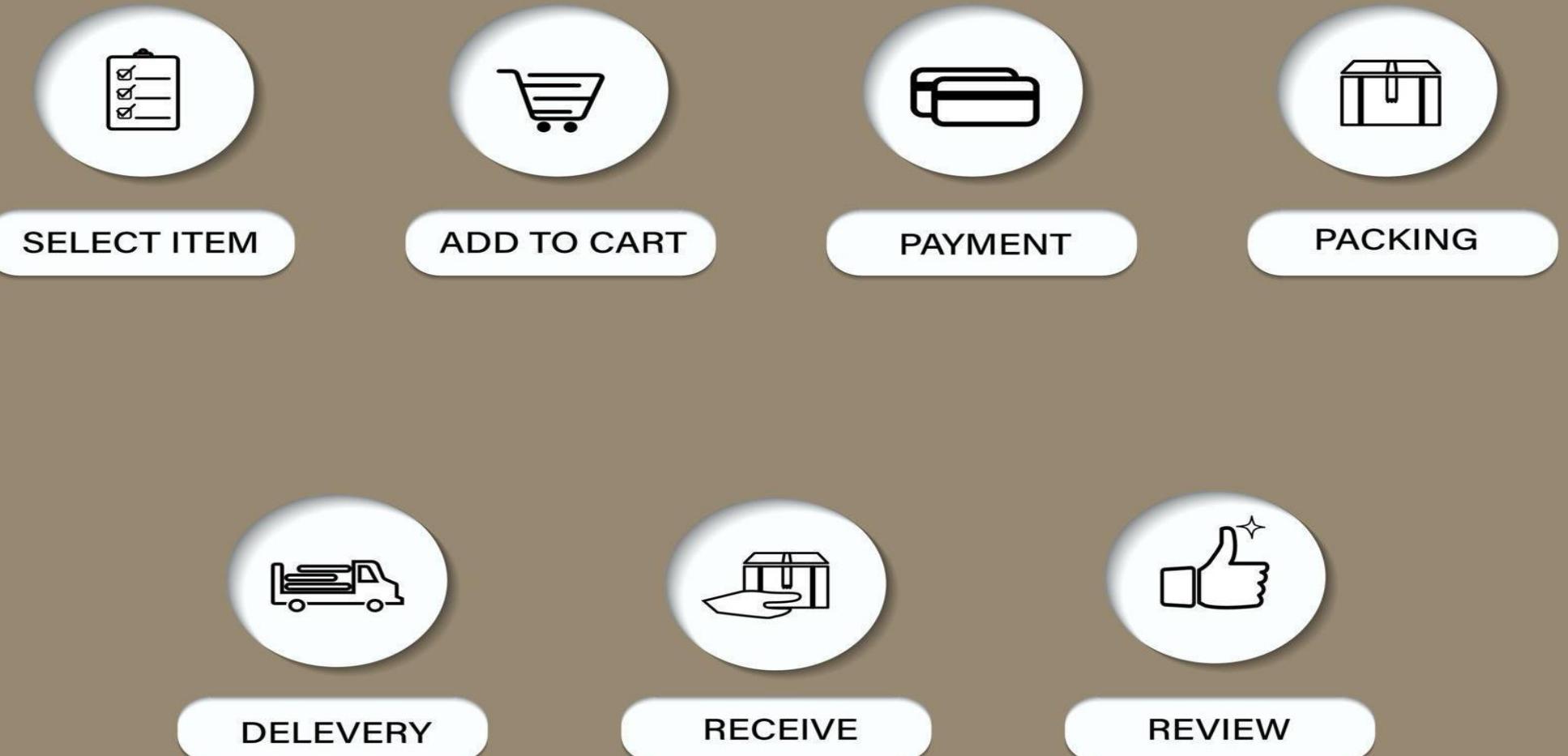
# Modularidade

Quebra de um problema complexo em problemas mais simples



# Modularidade

Como realizar o processamento de um pedido em uma loja?

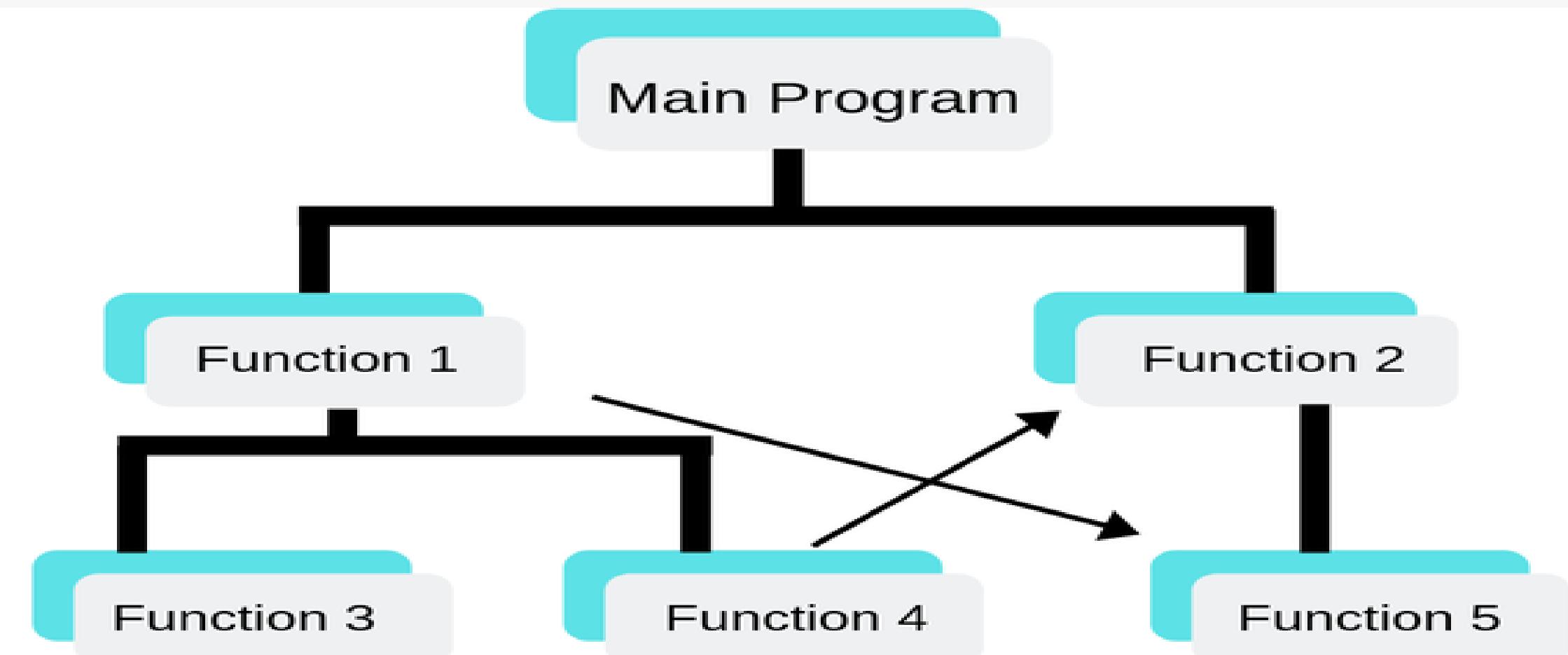


# Modularidade

*Propriedade de construção de sistemas através de módulos*

A coesão de um módulo está relacionada a inter-relação entre os seus componentes.

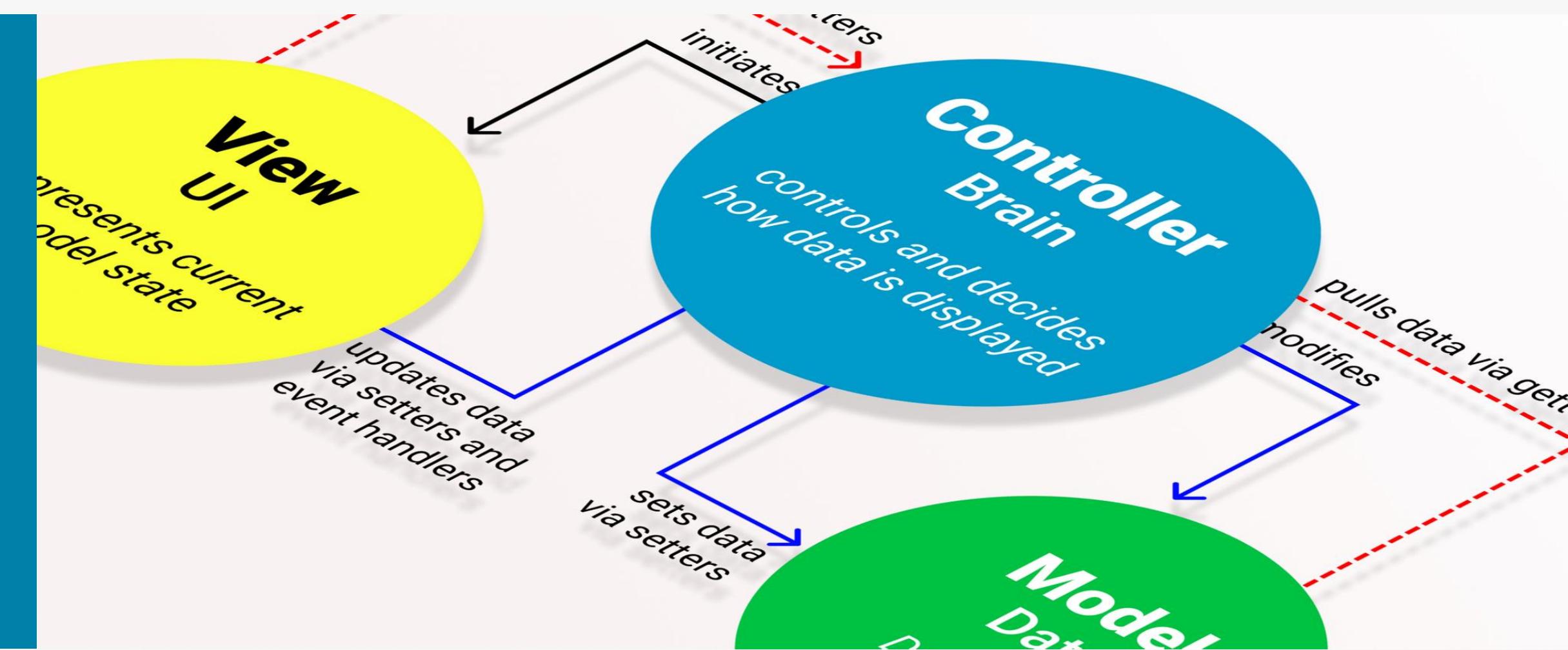
Uma classe não deve assumir responsabilidades que não são suas.



- O acoplamento entre os módulos é o grau de interdependência entre eles
- Ideal: Alta coesão e Baixo acoplamento

# Modularidade

Modularização é o processo de dividir um todo em partes bem definidas, que podem ser construídas e examinadas separadamente



- Essas partes interagem entre si, fazendo com que o sistema funcione de forma adequada
- Particionar um programa em componentes individuais, pode reduzir a complexidade



8

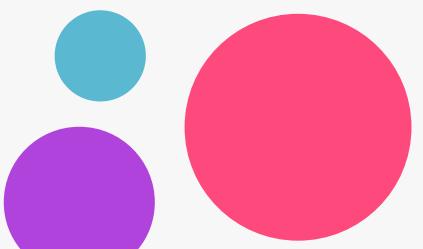
Hierarquia

# Hierarquia

Quando um sistema tem muitos detalhes, ele pode ser decomposto em uma hierarquia de abstrações

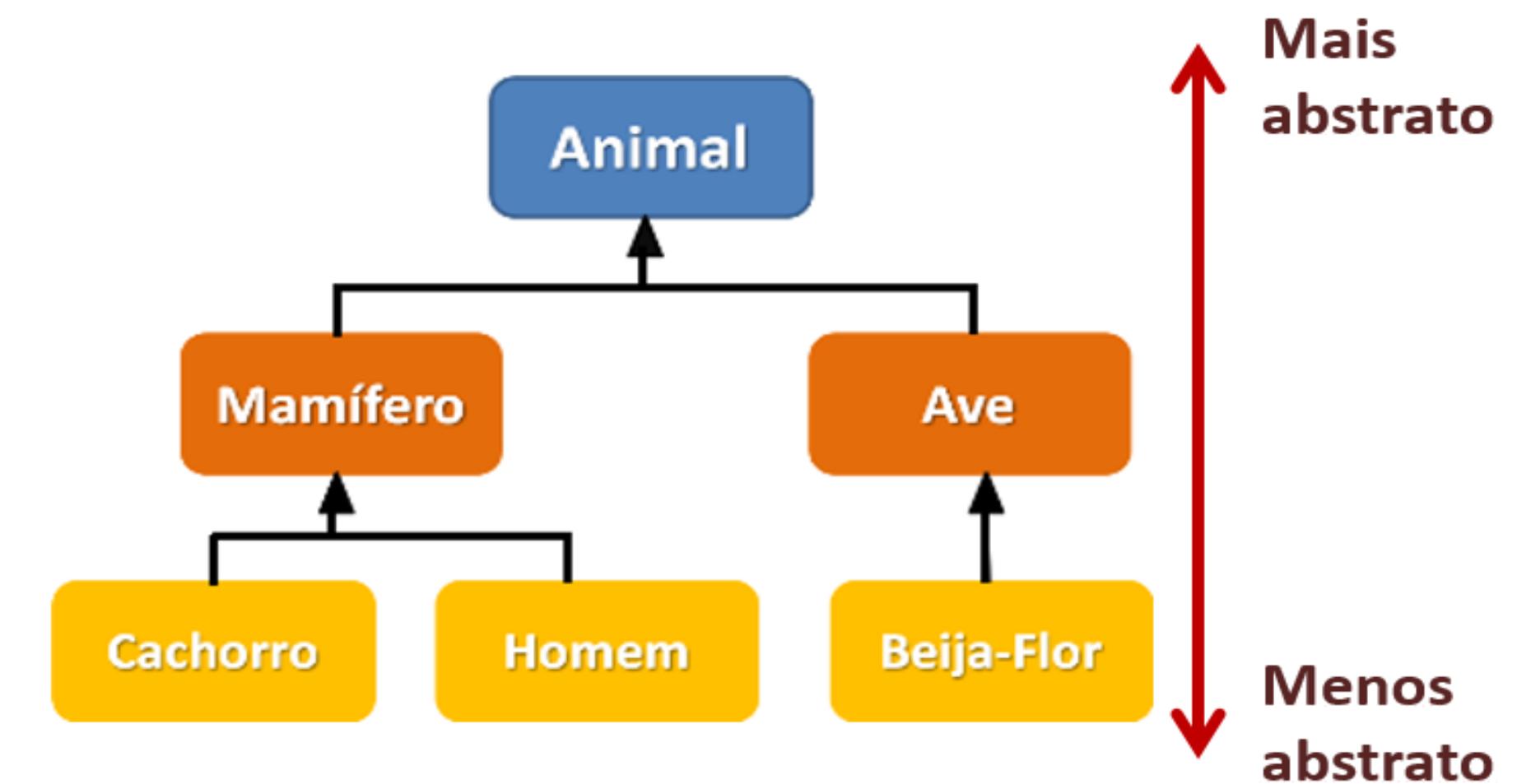


- Permite que detalhes relevantes sejam introduzidos de uma maneira controlada
- É um “ranking” ou uma ordenação de abstrações



# Hierarquia

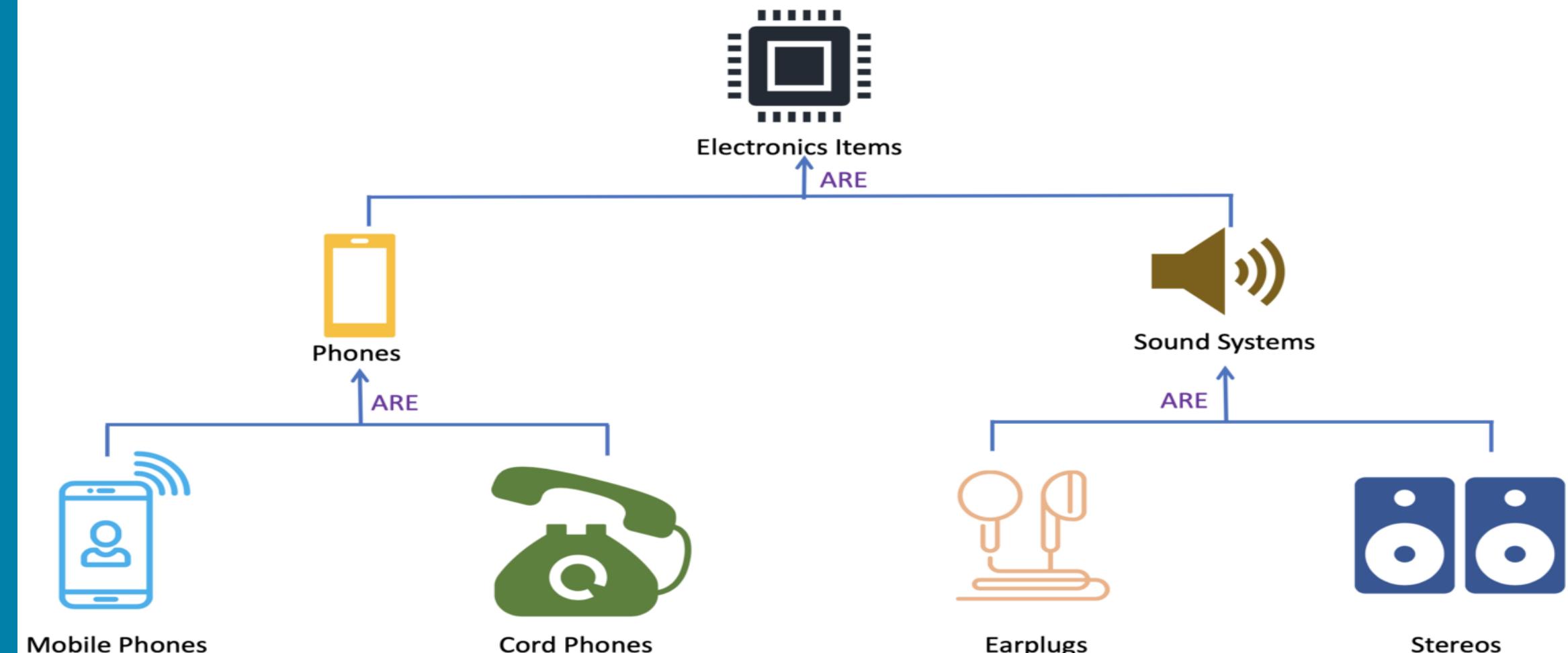
Nível de abstração de um  
objeto



# Hierarquia

*A herança define um tipo de relacionamento entre as classes.*

- Generalização: remover restrições para obter abstrações mais genéricas
- Especialização: buscar características que diferenciem abstrações afins

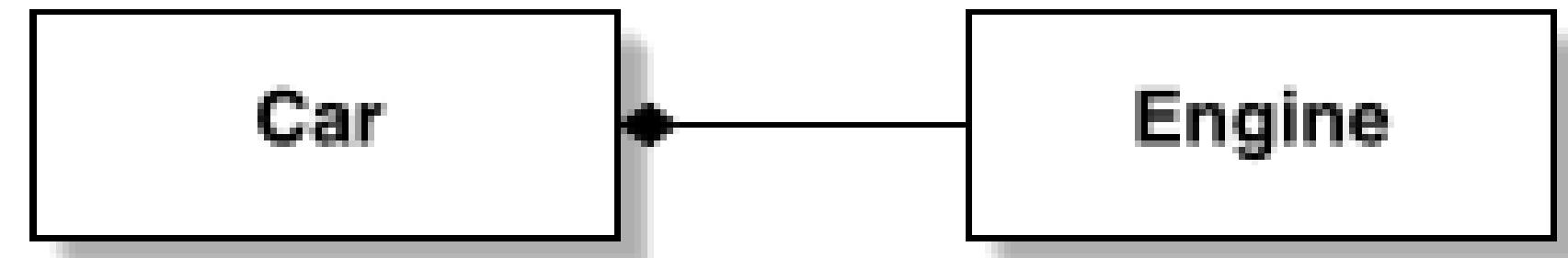


Ligaçāo Dinâmica é um conceito importante que faz com a que a herança e polimorfismo funcionem dinamicamente, isto é, em tempo de execução

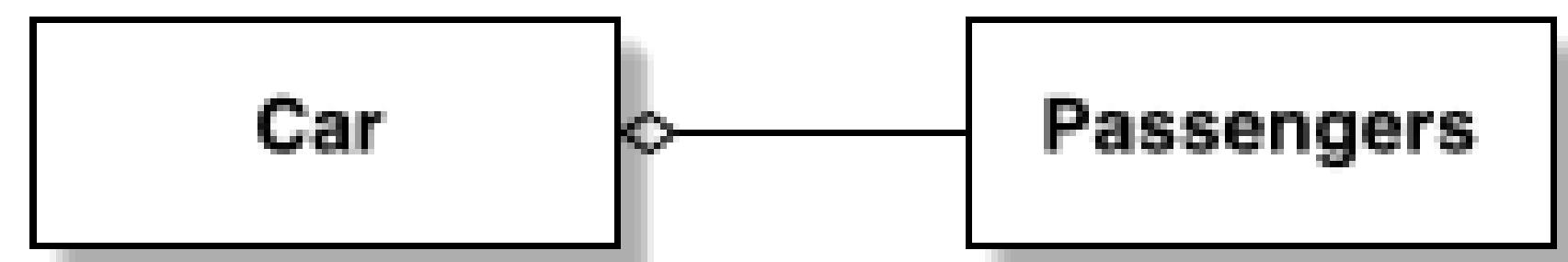
# Hierarquia

*Outros tipos de relacionamento entre classes/objetos.*

- Agregação: combinar abstrações para obter estruturas e comportamentos mais complexos
- Composição: significa detalhar uma abstração dividindo-a nos seus elementos mais constituintes



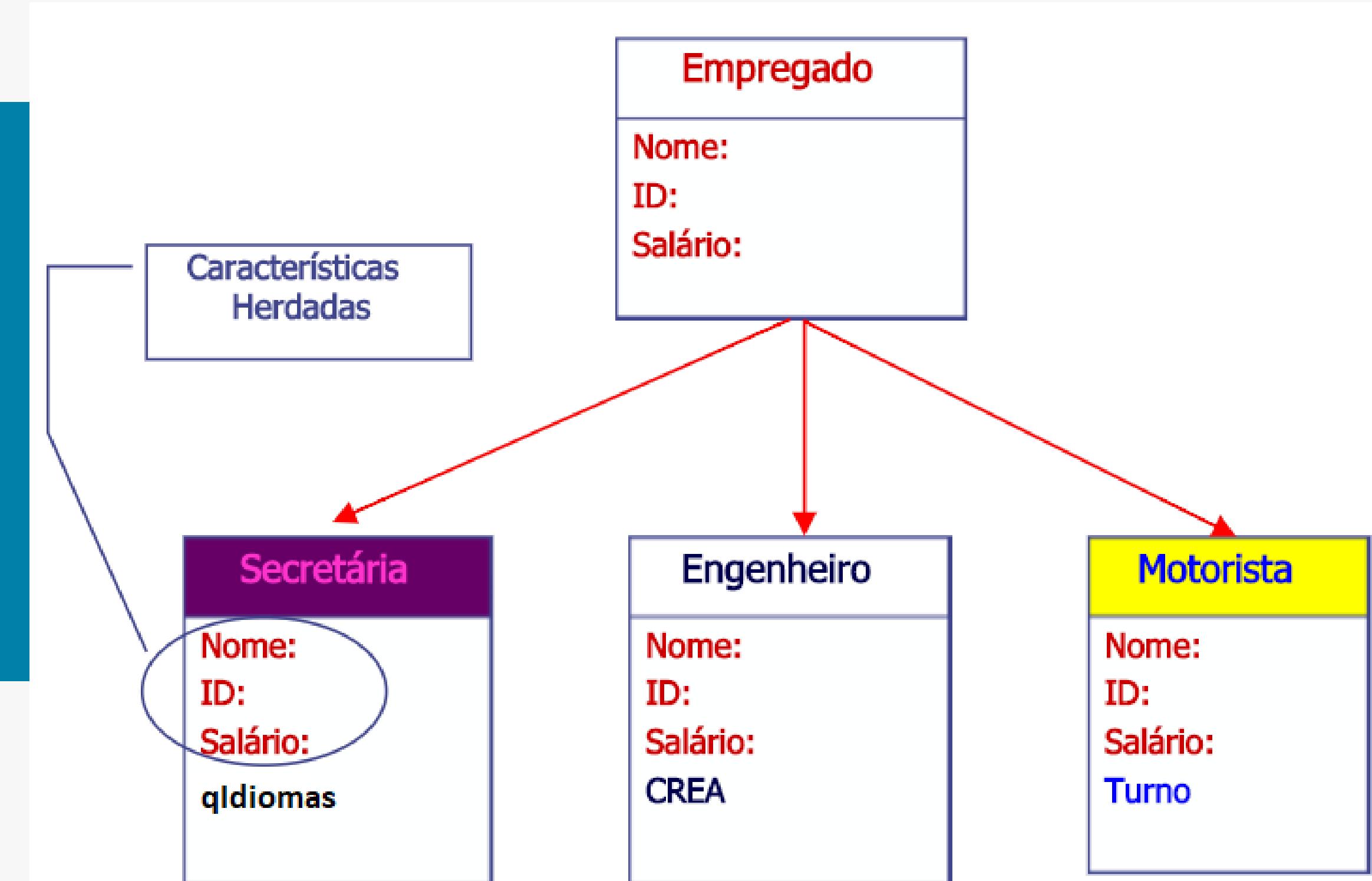
Composition: every car has an engine.



Aggregation: cars may have passengers, they come and go

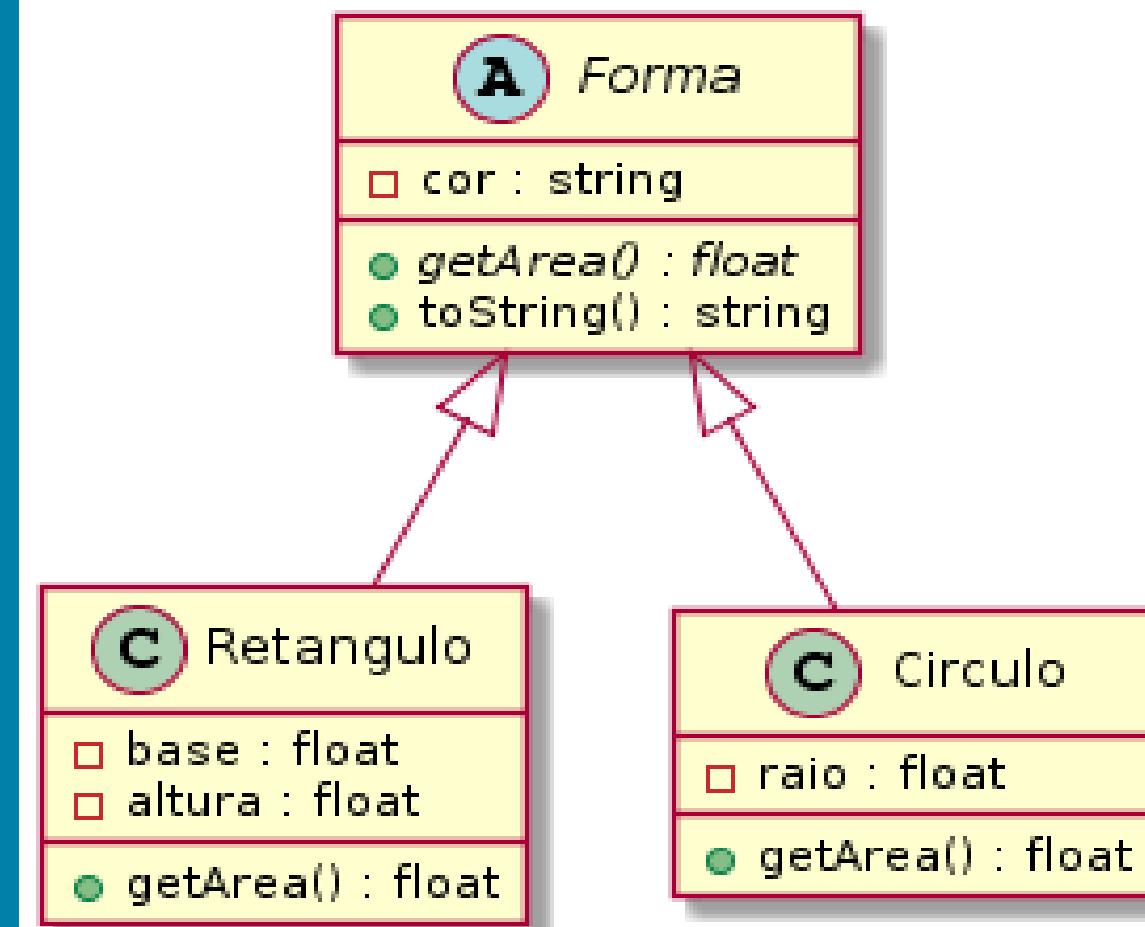
# Hierarquia

Observe que as classes **Secretária**, **Engenheiro** e **Motorista** herdam os atributos **Nome**, **ID** e **Salário**. Isto é, reutilizam algo já implementado anteriormente



# Hierarquia

Na prática



# Revisão

*Programar no paradigma orientada a objetos é*

**Abstrair**

Abstrair a sua visão do problema para um conjunto de objetos interagindo.

**Classes**

Definir classes (estrutura + operações) para os elementos do domínio do problema em questão.

**Composições**

A criação de uma classe pode usar a definição de classes prévias, criando composições de classes

**Herança**

Relacionar classes por meio de herança

**Vinculação dinâmica**

Entender o mecanismo de vinculação dinâmica da linguagem.

# Alguma dúvida?

**Não guardem dúvidas, perguntam**

• • •

# Referências

- 1 DA COSTA, Anderson Fabiano F. **Fundamentos de C++**. Instituto Federal da Paraíba. 2022.
- 2 Materiais de aula dos professores Guillermo Camara-Chavez, Tiago Maritan, Fred Guedes Pereira e Danielle Chaves.
- 3 HORSTMANN, C. **Conceitos de Computação com o Essencial de C++**, 3<sup>a</sup> edição, Bookman, 2005.
- 4 DEITEL, **C++ Como Programar**, 5<sup>a</sup> edição, Editora Prentice Hall, 2006
- 5