

*introdução à*  
**ORIENTAÇÃO**  
*a* **OBJETOS**

BY RAFA

# *introdução à* **ORIENTAÇÃO** *a* **OBJETOS**

**REVISÃO**

**PARADIGMAS DE PROGRAMAÇÃO**

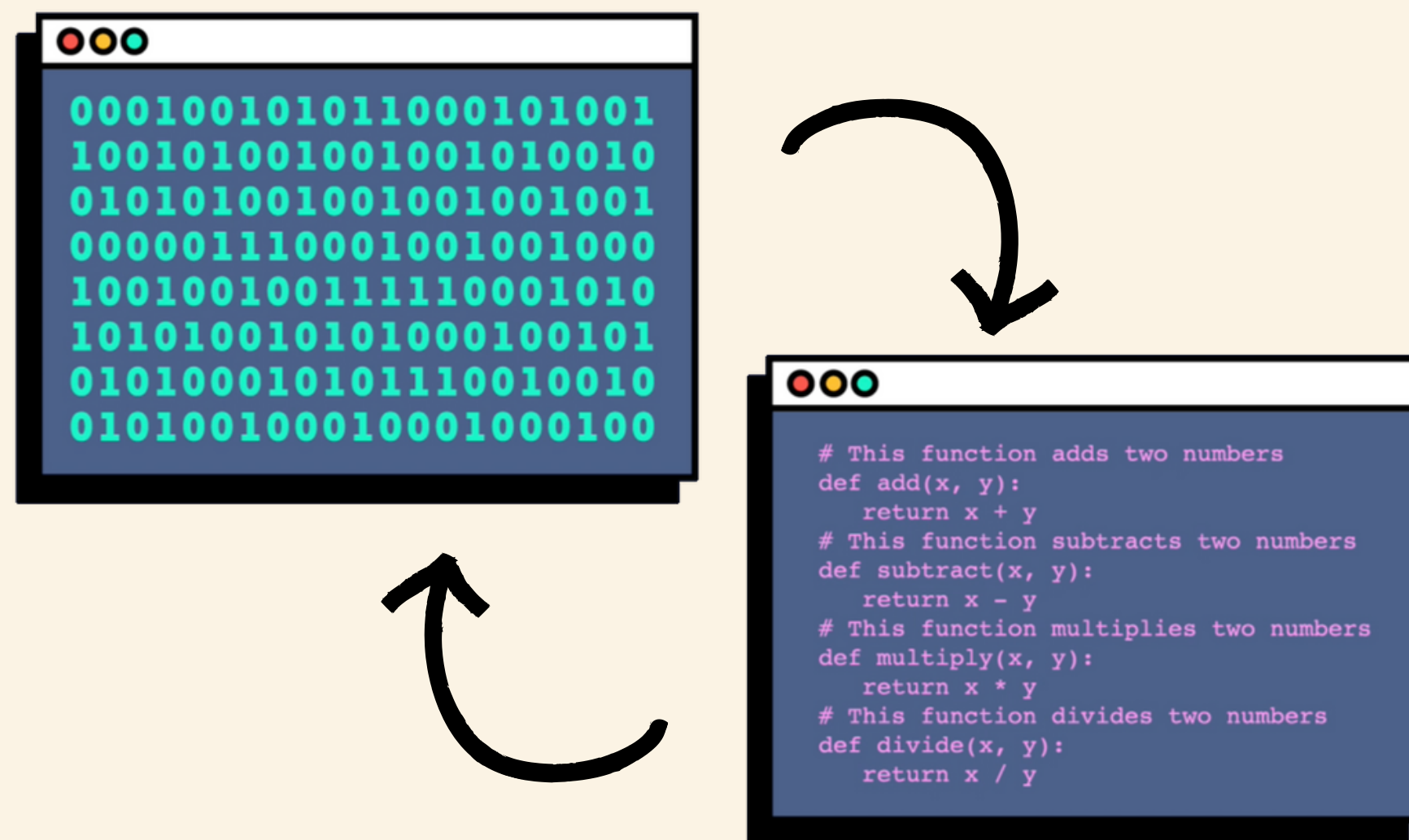
**ORIENTAÇÃO A OBJETOS**

**ABSTRAÇÃO**

# O QUE É UMA LINGUAGEM DE PROGRAMAÇÃO?



# Linguagens de PROGRAMAÇÃO



## O que são?

Conjunto de sintaxes utilizados para criar comandos na comunicação humano-máquina

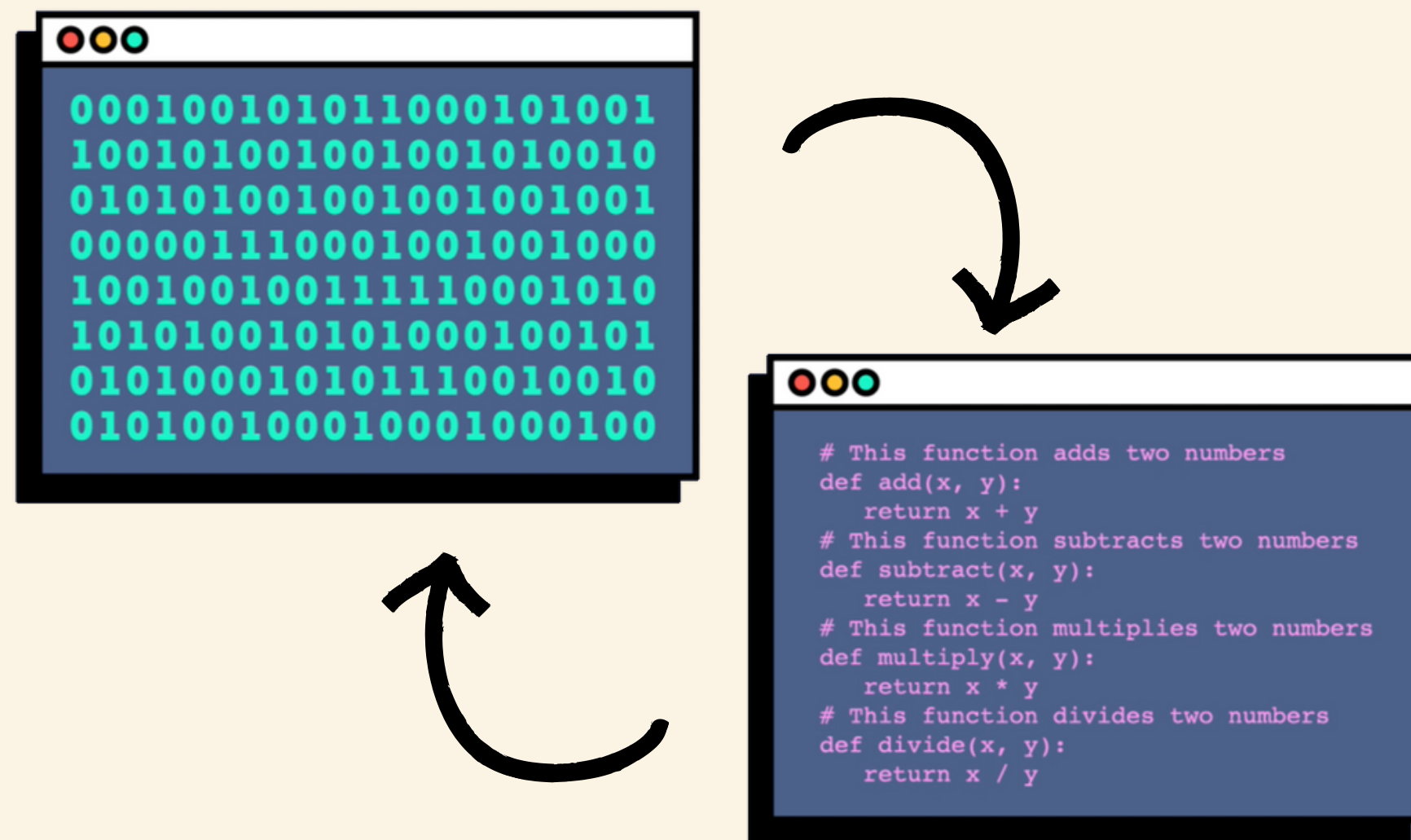
## Como classificar?

São classificadas de BAIXO a ALTO nível sendo que, quanto mais alto, mais próximo à linguagem humana

## Como compreender?

Existem 3 tipos de tradutores: Compiladores, Transpiladores e Interpretadores

# Linguagens de PROGRAMAÇÃO



## O que são?

Conjunto de sintaxes utilizados para criar comandos na comunicação humano-máquina

## Como classificar?

São classificadas de BAIXO a ALTO nível sendo que, quanto mais alto, mais próximo à linguagem humana

## Como compreender?

Existem 3 tipos de tradutores: Compiladores, Transpiladores e Interpretadores

---

*paradigmas de*  
**PROGRAMAÇÃO**

---



paradigma

*substantivo masculino*

1. um exemplo que serve como modelo; padrão.
2. **GRAMÁTICA**  
conjunto de formas vocabulares que servem de modelo para um sistema de flexão ou de derivação (p.ex.: na declinação, na conjugação etc.); padrão.

paradigma

---

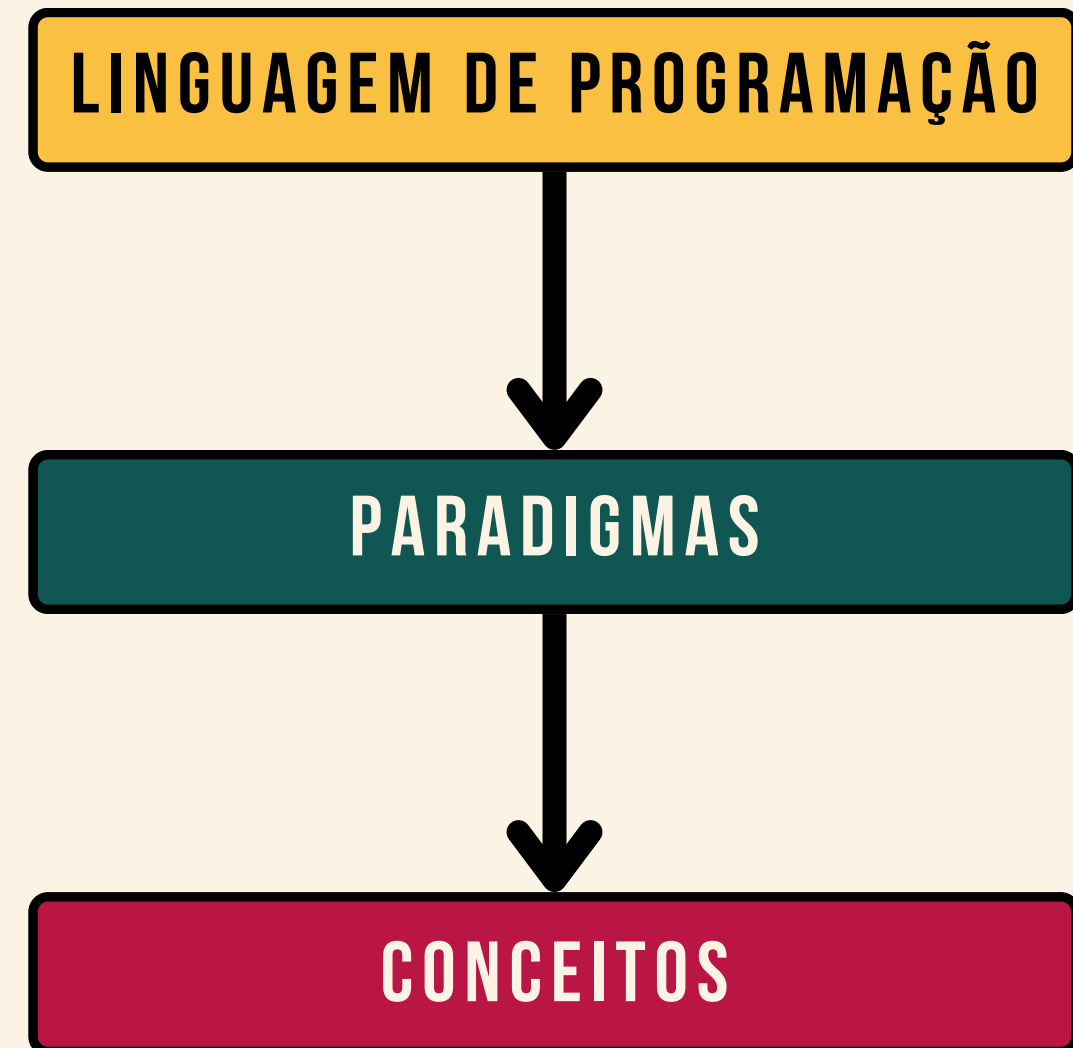
# paradigmas de PROGRAMAÇÃO

“ Um padrão que serve como **forma de pensar** para a programação. ”

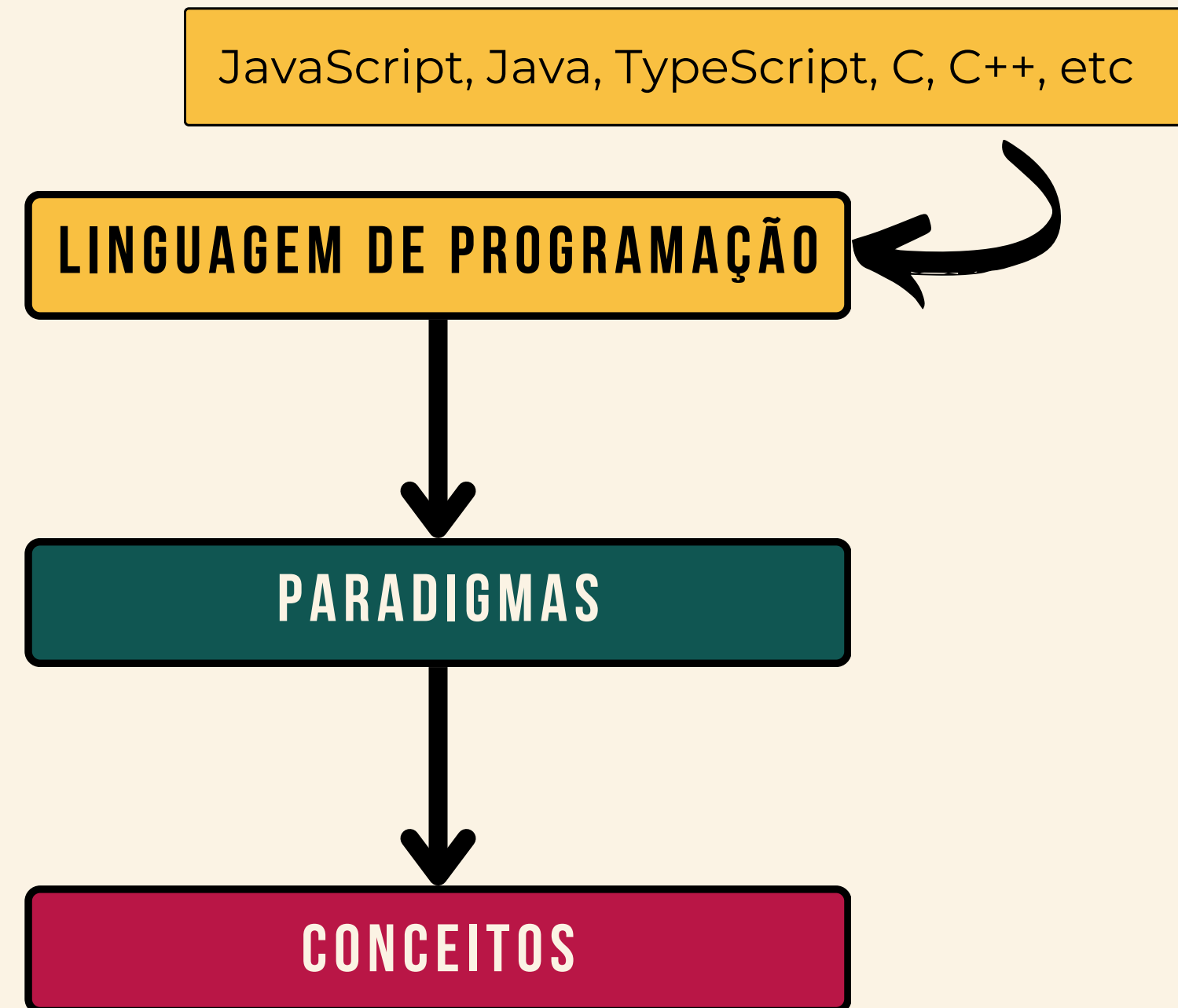
BY KURT NØRMARK



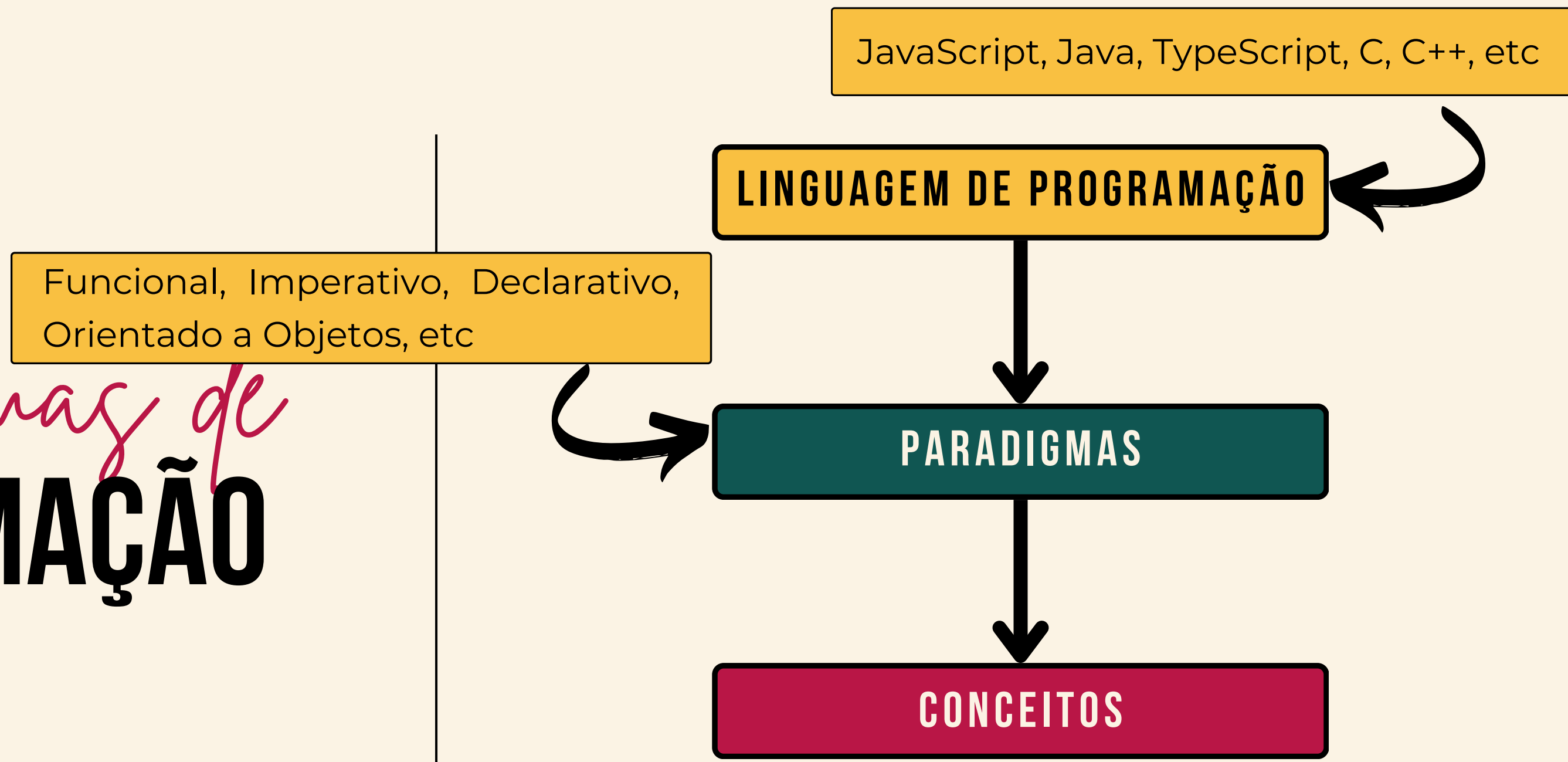
# *paradigmas de* **PROGRAMAÇÃO**



# *paradigmas de* **PROGRAMAÇÃO**



# *paradigmas de* **PROGRAMAÇÃO**



# *paradigmas de* **PROGRAMAÇÃO**

Funcional, Imperativo, Declarativo,  
Orientado a Objetos, etc

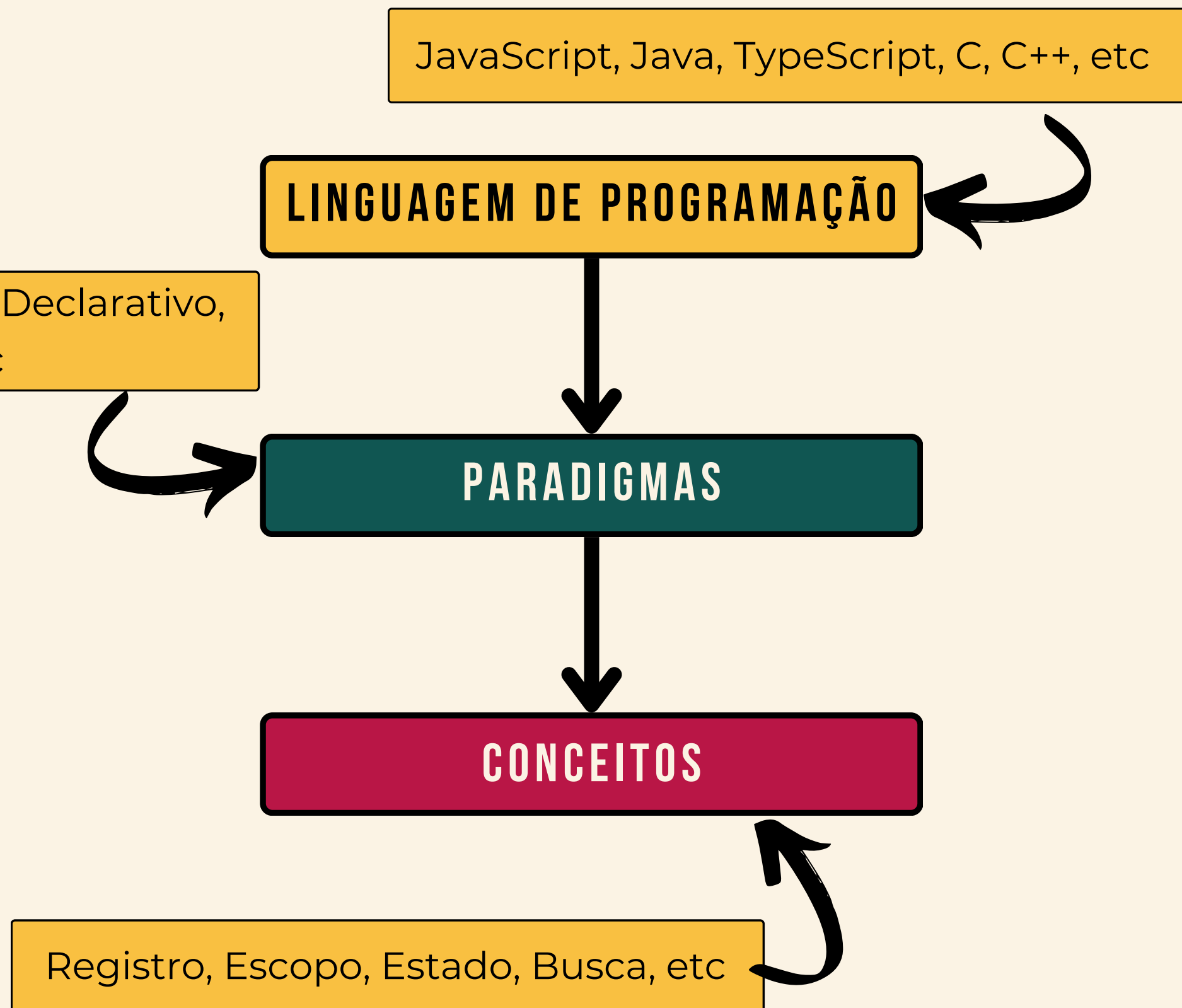
LINGUAGEM DE PROGRAMAÇÃO

JavaScript, Java, TypeScript, C, C++, etc

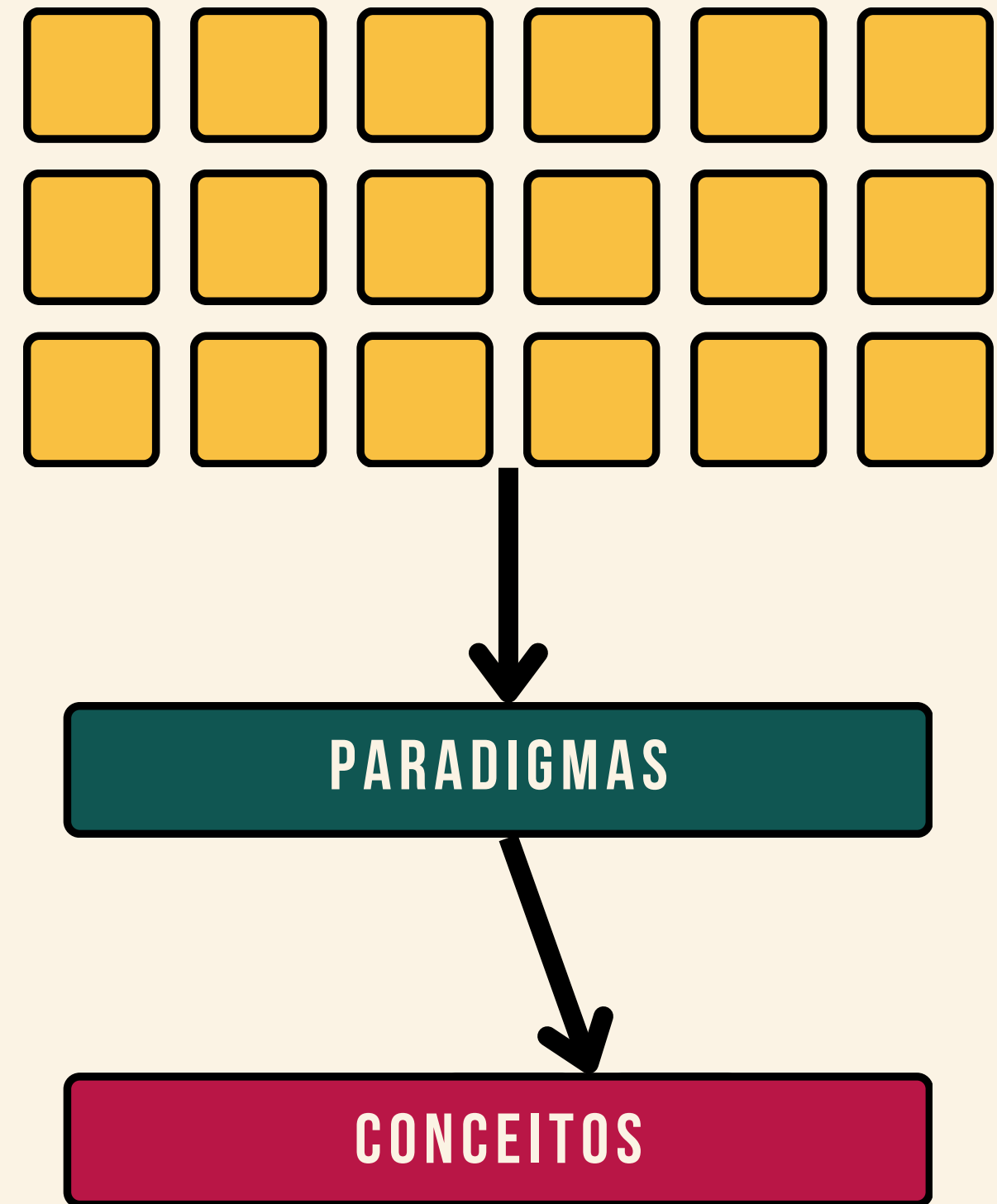
PARADIGMAS

CONCEITOS

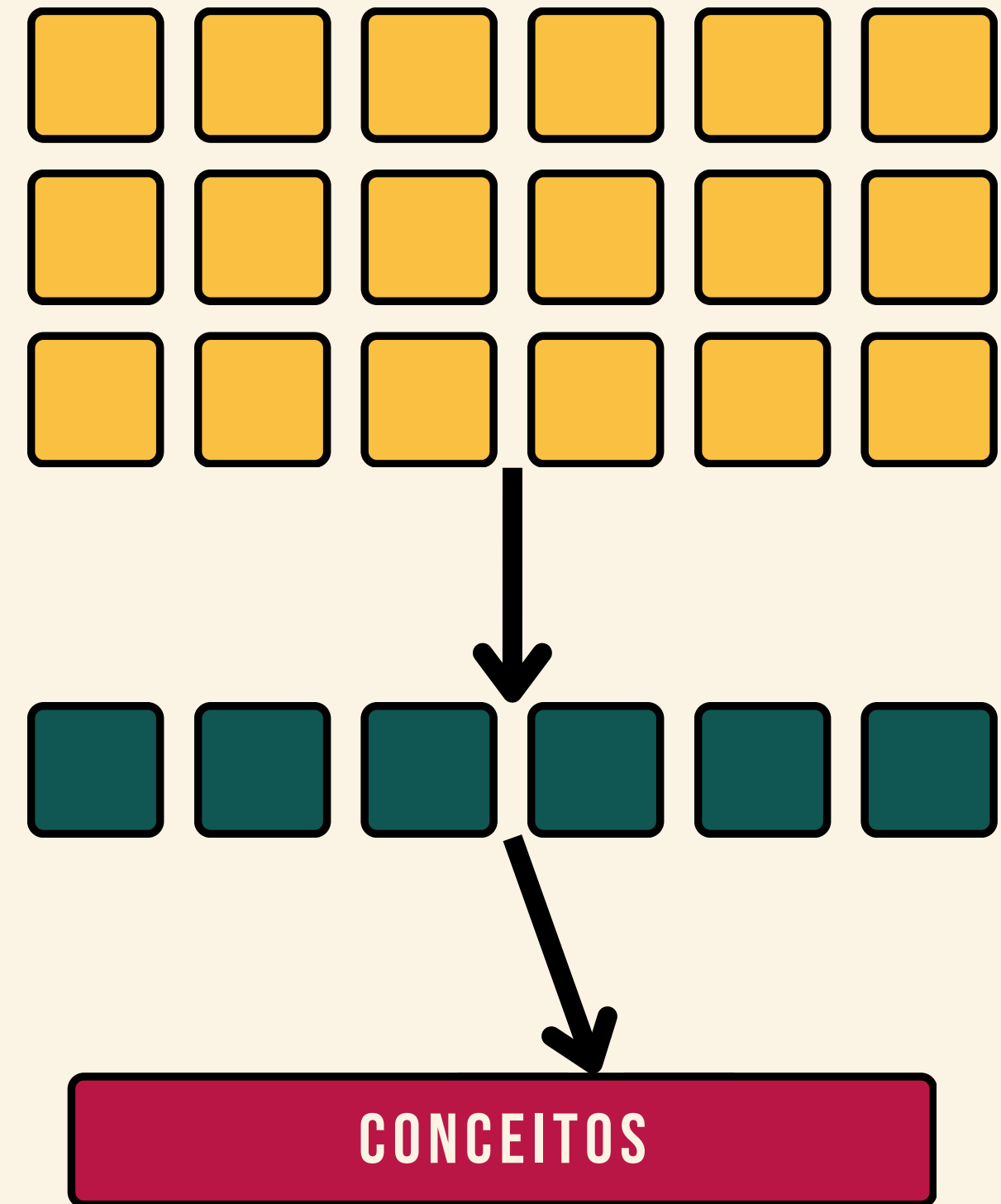
Registro, Escopo, Estado, Busca, etc



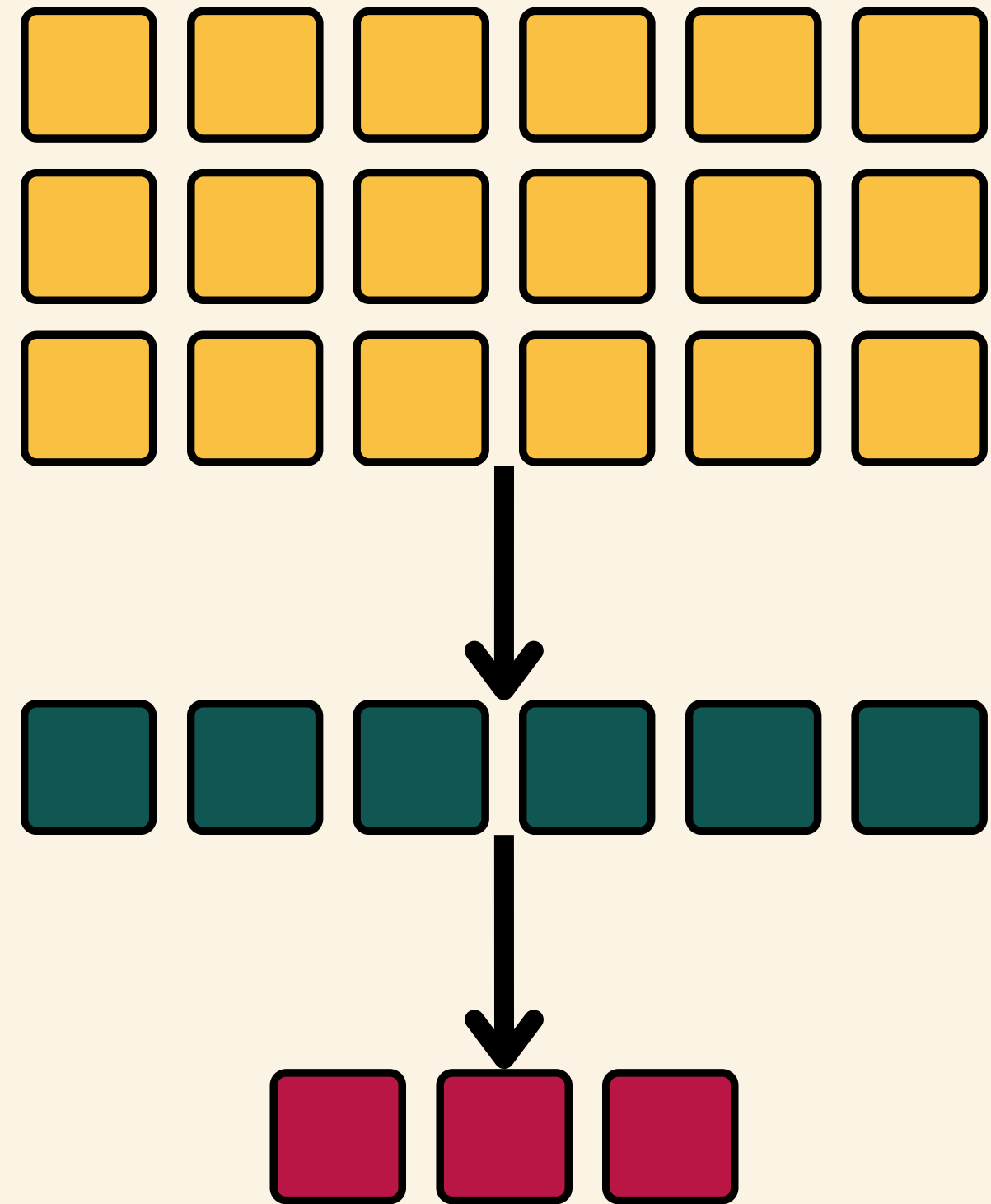
# *paradigmas de* **PROGRAMAÇÃO**



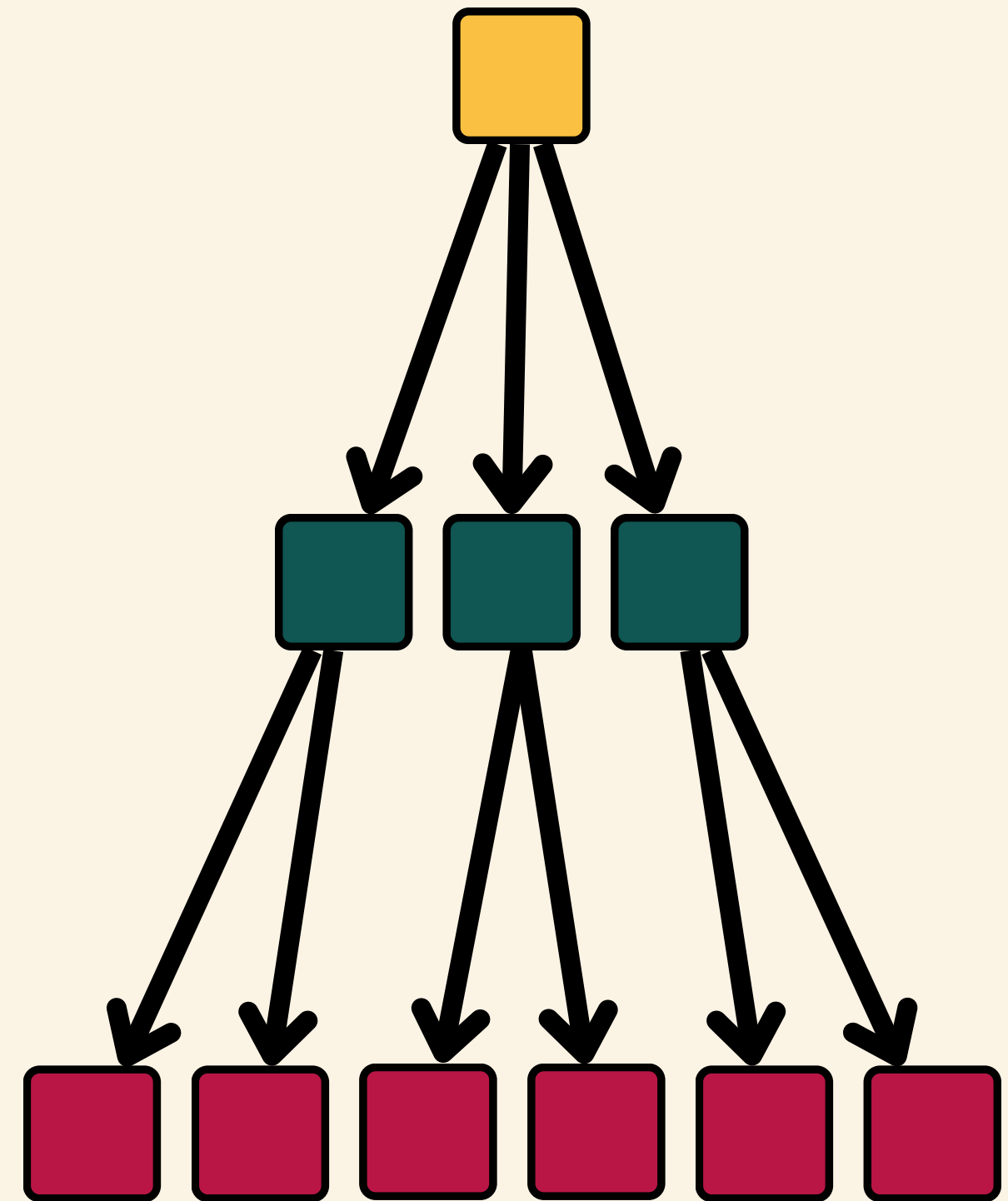
# *paradigmas de* **PROGRAMAÇÃO**



# *paradigmas de* **PROGRAMAÇÃO**

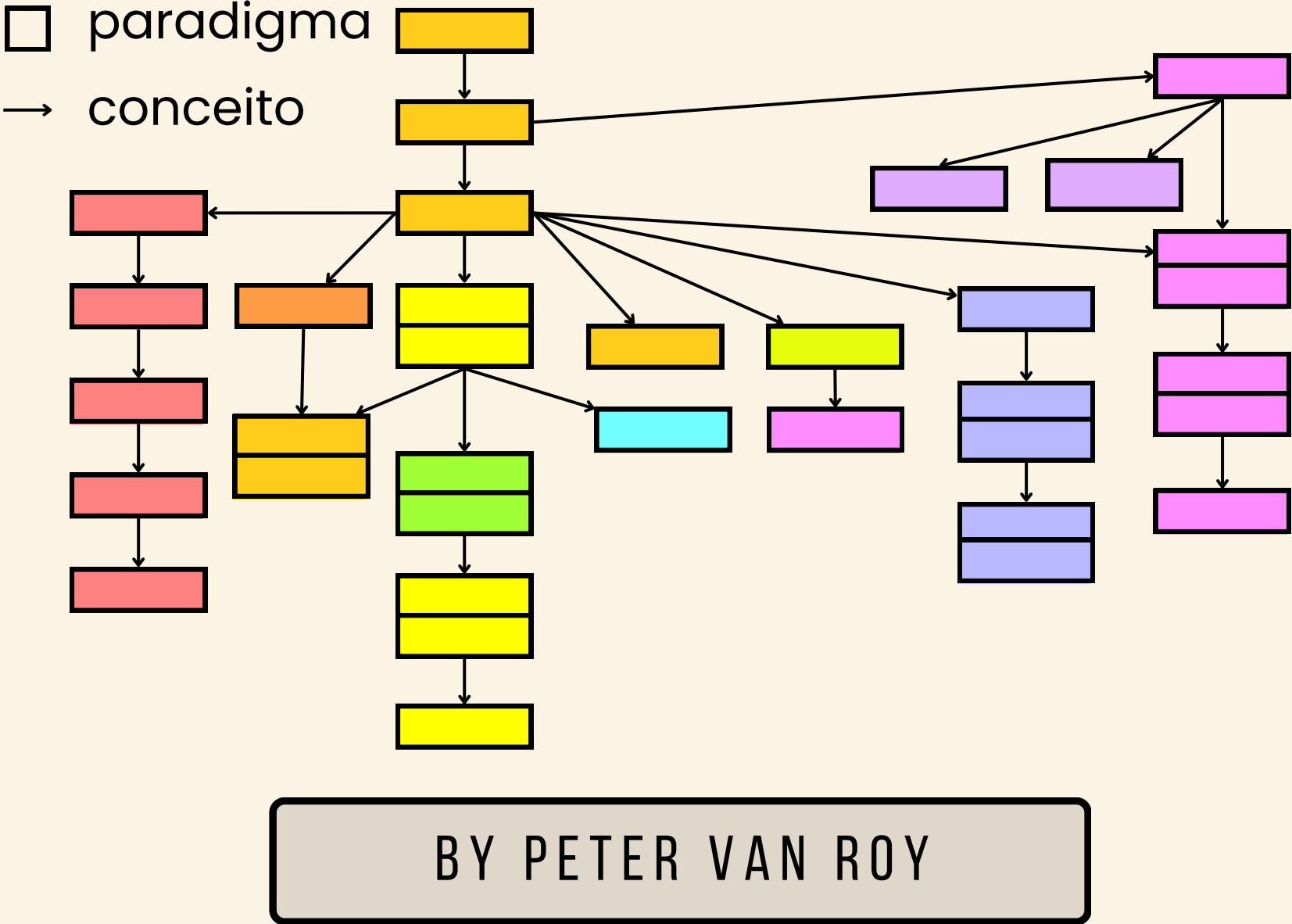


# *paradigmas de* **PROGRAMAÇÃO**

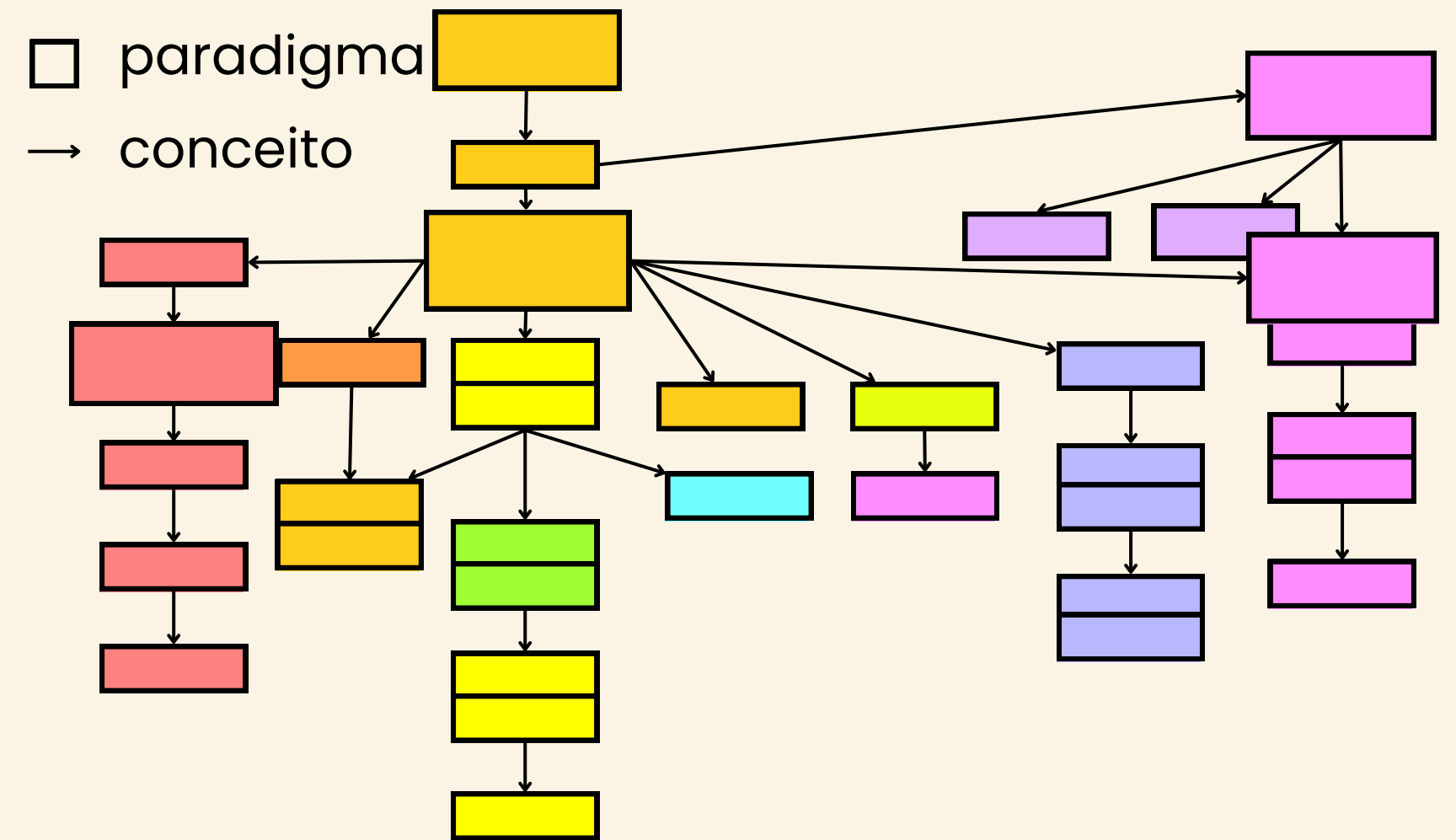




# paradigmas de PROGRAMAÇÃO

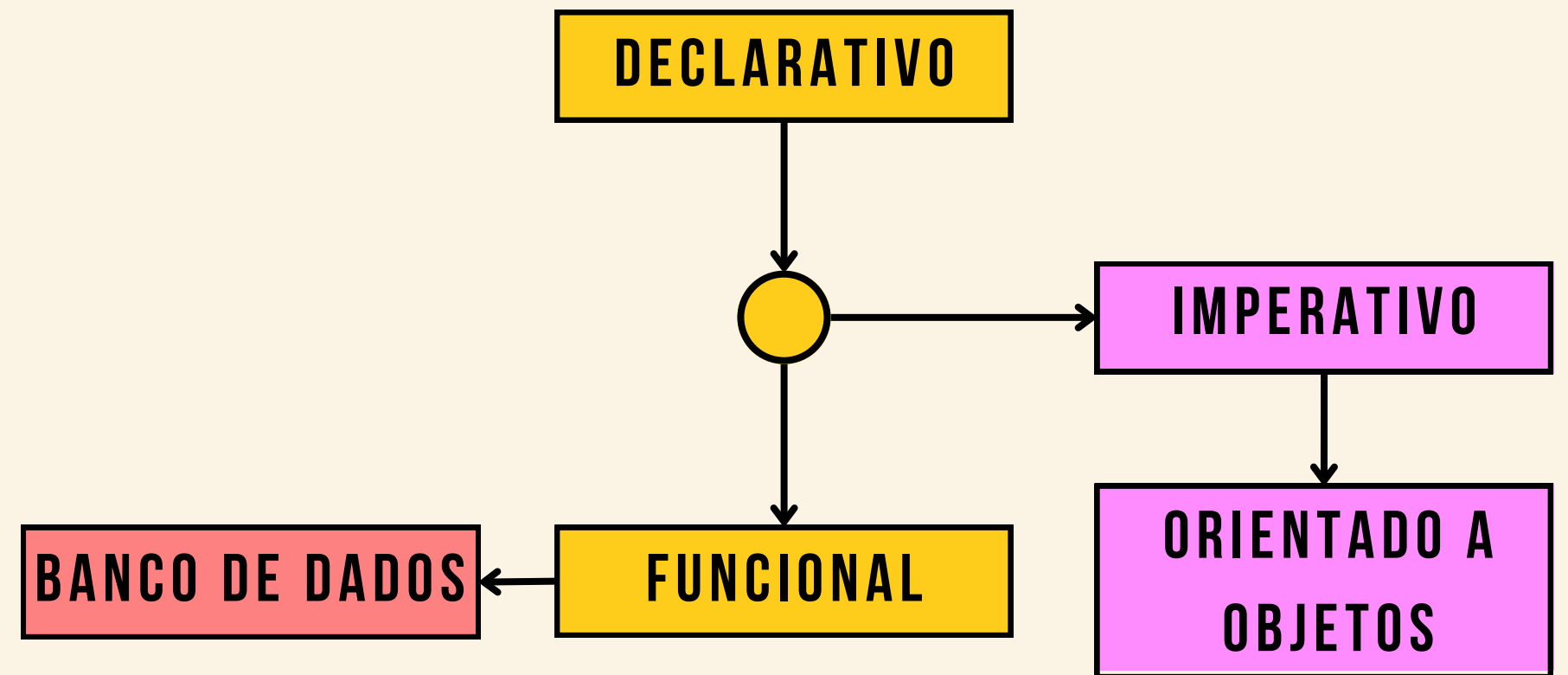


# *paradigmas de* **PROGRAMAÇÃO**

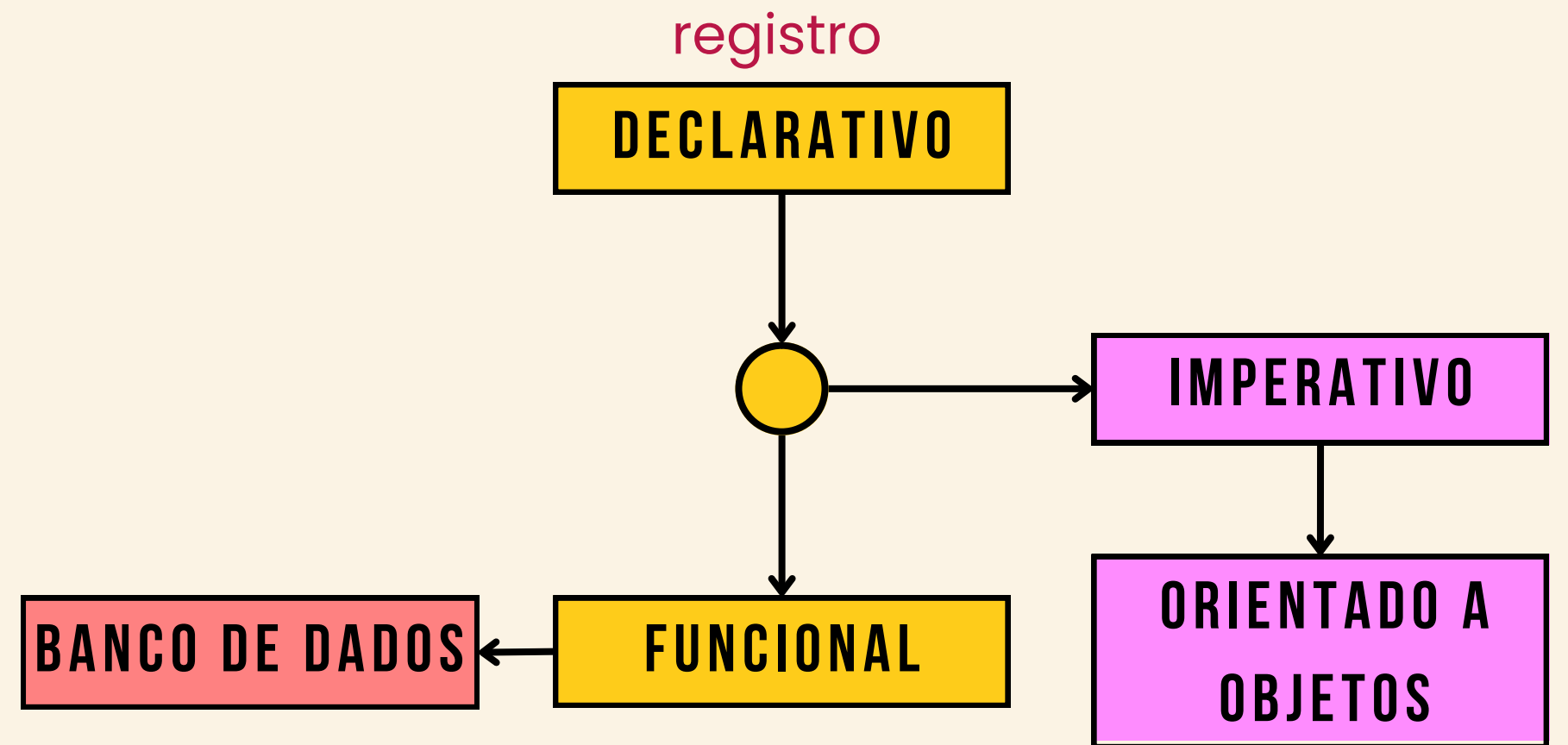


BY PETER VAN ROY

# *paradigmas de* **PROGRAMAÇÃO**



# *paradigmas de* **PROGRAMAÇÃO**



# *paradigmas de* **PROGRAMAÇÃO**

**DECLARATIVO**

# *paradigmas de* **PROGRAMAÇÃO**

## **DECLARATIVO**

Meramente um a **declaração** de algo

# *paradigmas de* **PROGRAMAÇÃO**

## DECLARATIVO

Meramente um a **declaração** de algo

Mais focado no **o que** que no **como**

# *paradigmas de* **PROGRAMAÇÃO**

## DECLARATIVO

Meramente um a **declaração** de algo

Mais focado no **o que** que no **como**

Define verdades **imutáveis** levando aos  
mesmos resultados **sempre**



# *paradigmas de* **PROGRAMAÇÃO**

## DECLARATIVO

Meramente um a **declaração** de algo

Mais focado no **o que** que no **como**

Define verdades **imutáveis** levando aos  
mesmos resultados **sempre**

Exemplos: HTML, XML

# DECLARATIVO

Meramente um a **declaração** de algo

Mais focado no **o que** que no **como**

Define verdades **imutáveis** levando aos  
mesmos resultados **sempre**

Exemplos: HTML, XML

```
<note>
  <to>Claudio</to>
  <from>Rafael</from>
  <heading>Lembrete</heading>
  <body>Lavar a louça</body>
</note>
```

# DECLARATIVO

Meramente um a **declaração** de algo

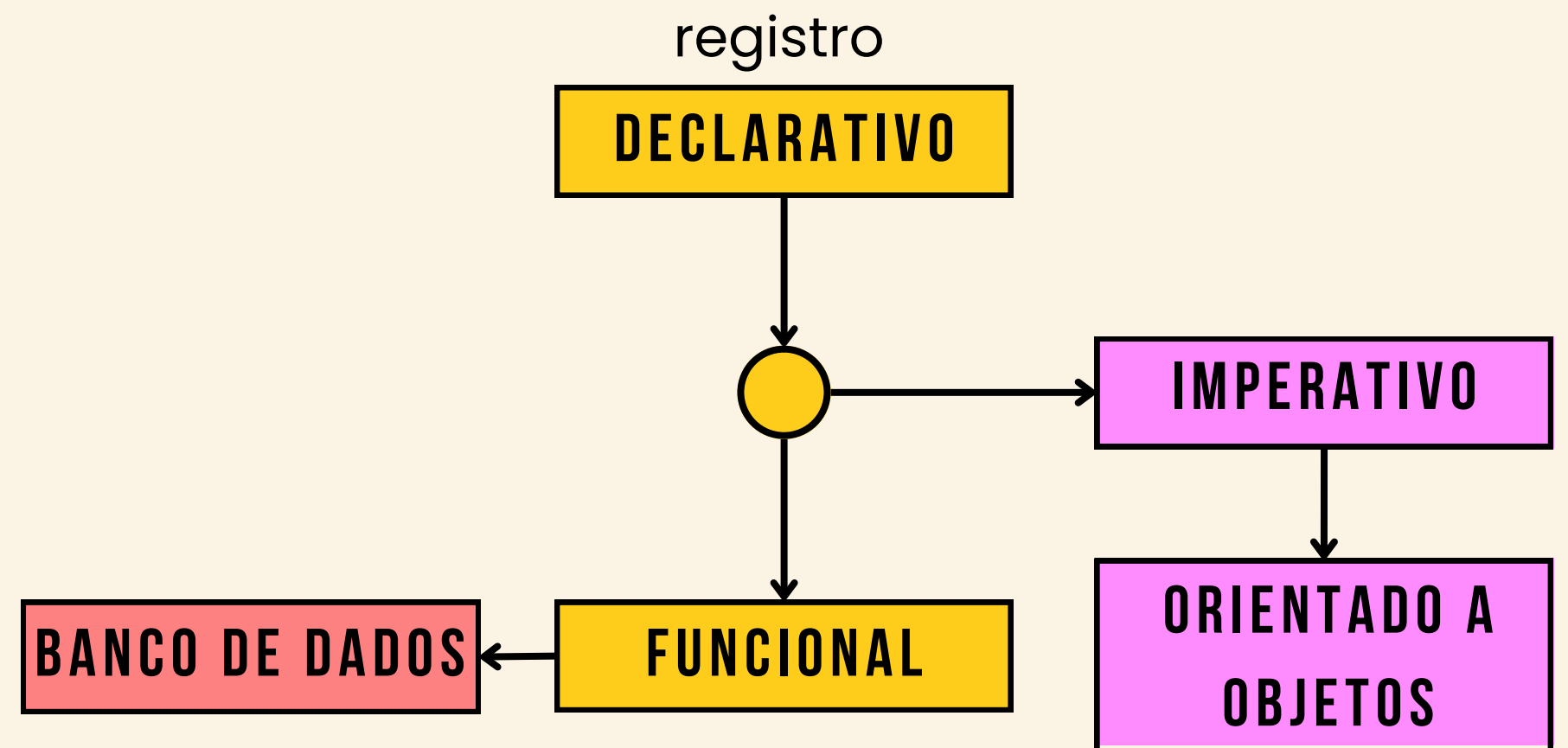
Mais focado no **o que** que no **como**

Define verdades **imutáveis** levando aos  
mesmos resultados **sempre**

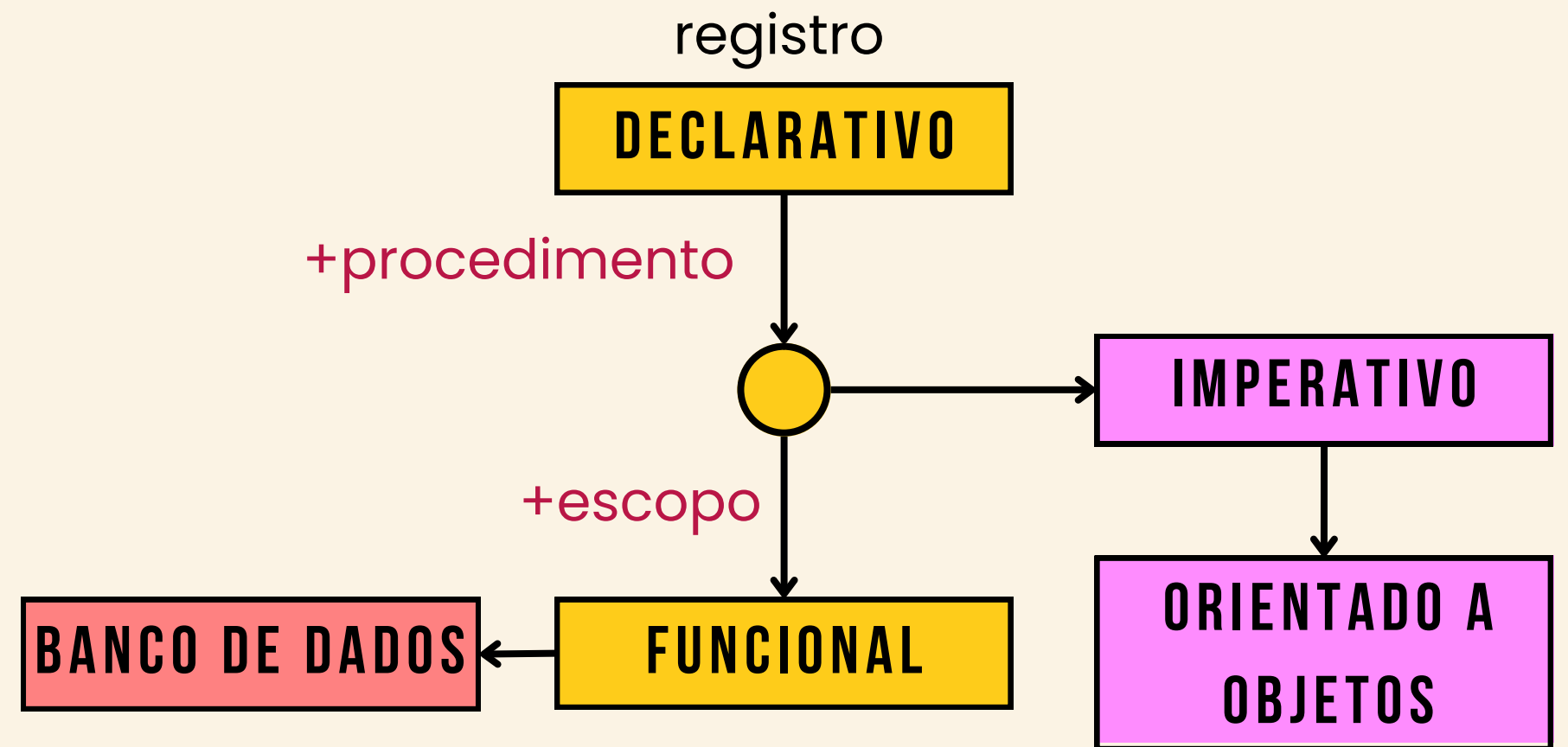
Exemplos: HTML, XML

```
<!DOCTYPE html>
<html>
  <head>
    <title>Título</title>
  </head>
  <body>
    <h1>Cabeçalho</h1>
    <p>Um parágrafo simples</p>
  </body>
</html>
```

# *paradigmas de* **PROGRAMAÇÃO**



# *paradigmas de* **PROGRAMAÇÃO**



# *paradigmas de* **PROGRAMAÇÃO**

**FUNCIONAL**

# *paradigmas de* **PROGRAMAÇÃO**

## **FUNCIONAL**

Sintaxe semelhante a **funções matemáticas**

# *paradigmas de* **PROGRAMAÇÃO**

## **FUNCIONAL**

Sintaxe semelhante a **funções matemáticas**

Onde variáveis são **imutáveis**



# *paradigmas de* **PROGRAMAÇÃO**

## **FUNCIONAL**

Sintaxe semelhante a **funções matemáticas**

Onde variáveis são **imutáveis**

Não efetua **mudanças de estado**

# *paradigmas de* **PROGRAMAÇÃO**

## **FUNCIONAL**

Sintaxe semelhante a **funções matemáticas**

Onde variáveis são **imutáveis**

Não efetua **mudanças de estado**

Não possui operações iterativas (**laços**)

# *paradigmas de* **PROGRAMAÇÃO**

## **FUNCIONAL**

Sintaxe semelhante a **funções matemáticas**

Onde variáveis são **imutáveis**

Não efetua **mudanças de estado**

Não possui operações iterativas (**laços**)

Exemplos: Haskell, Javascript

# *paradigmas de* **PROGRAMAÇÃO**

## **FUNCIONAL**

Sintaxe semelhante a **funções matemáticas**

Onde variáveis são **imutáveis**

Não efetua **mudanças de estado**

Não possui operações iterativas (**laços**)

Exemplos: Haskell, Javascript

Em linguagens  
puramente funcionais

# FUNCIONAL

Sintaxe semelhante a **funções matemáticas**

Onde variáveis são **imutáveis**

Não efetua **mudanças de estado**

Não possui operações iterativas (**laços**)

Exemplos: Haskell, Javascript

```
doubleMe x = x + x
doubleUs x y = x*2 + y*2
doubleUs2 x y = doubleMe x + doubleMe y

ghci> doubleMe 9
18
```

# FUNCIONAL

Sintaxe semelhante a **funções matemáticas**

Onde variáveis são **imutáveis**

Não efetua **mudanças de estado**

Não possui operações iterativas (**laços**)

Exemplos: Haskell, Javascript



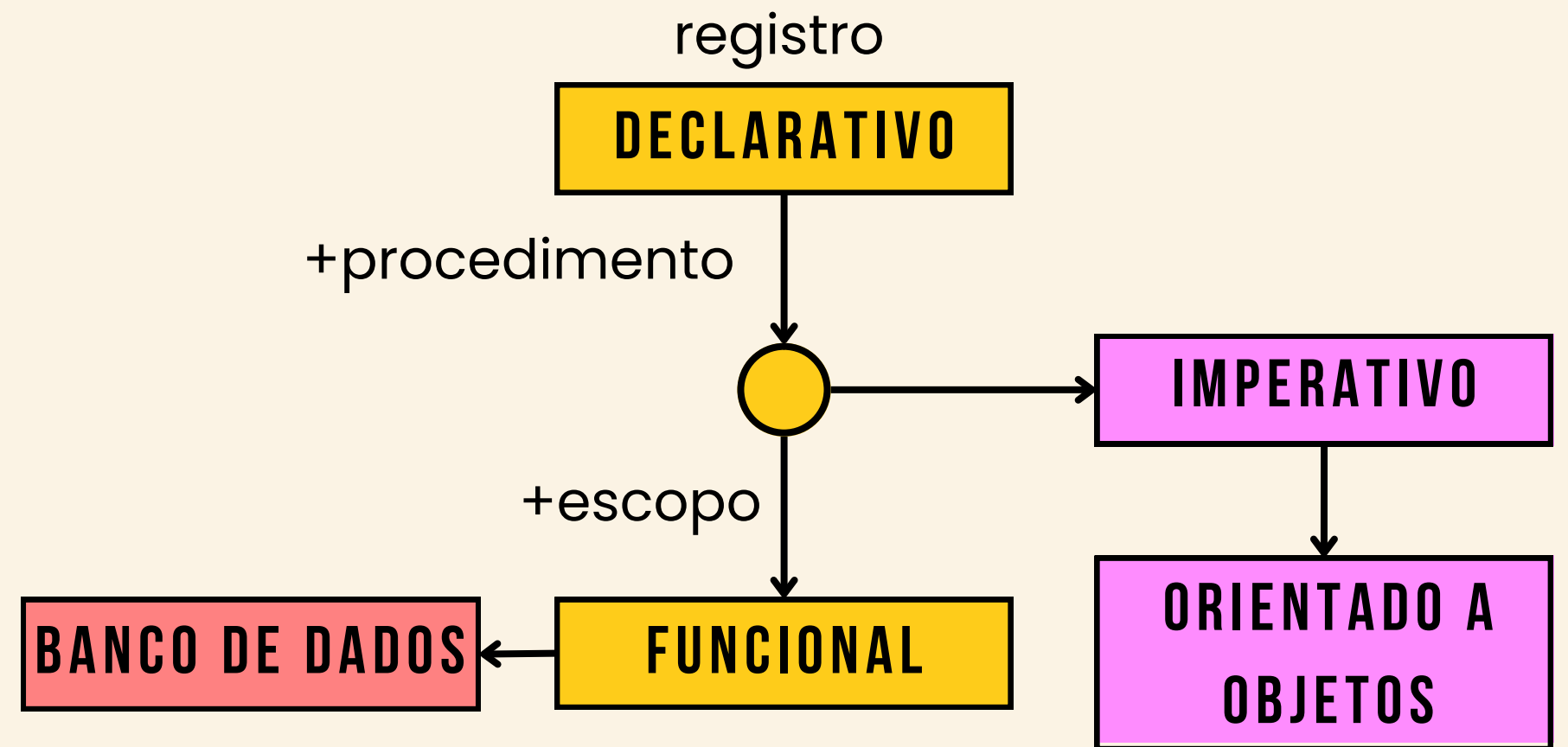
```
const sum = (a,b) => a + b
```

```
const resultSum = sum(1,2)
```

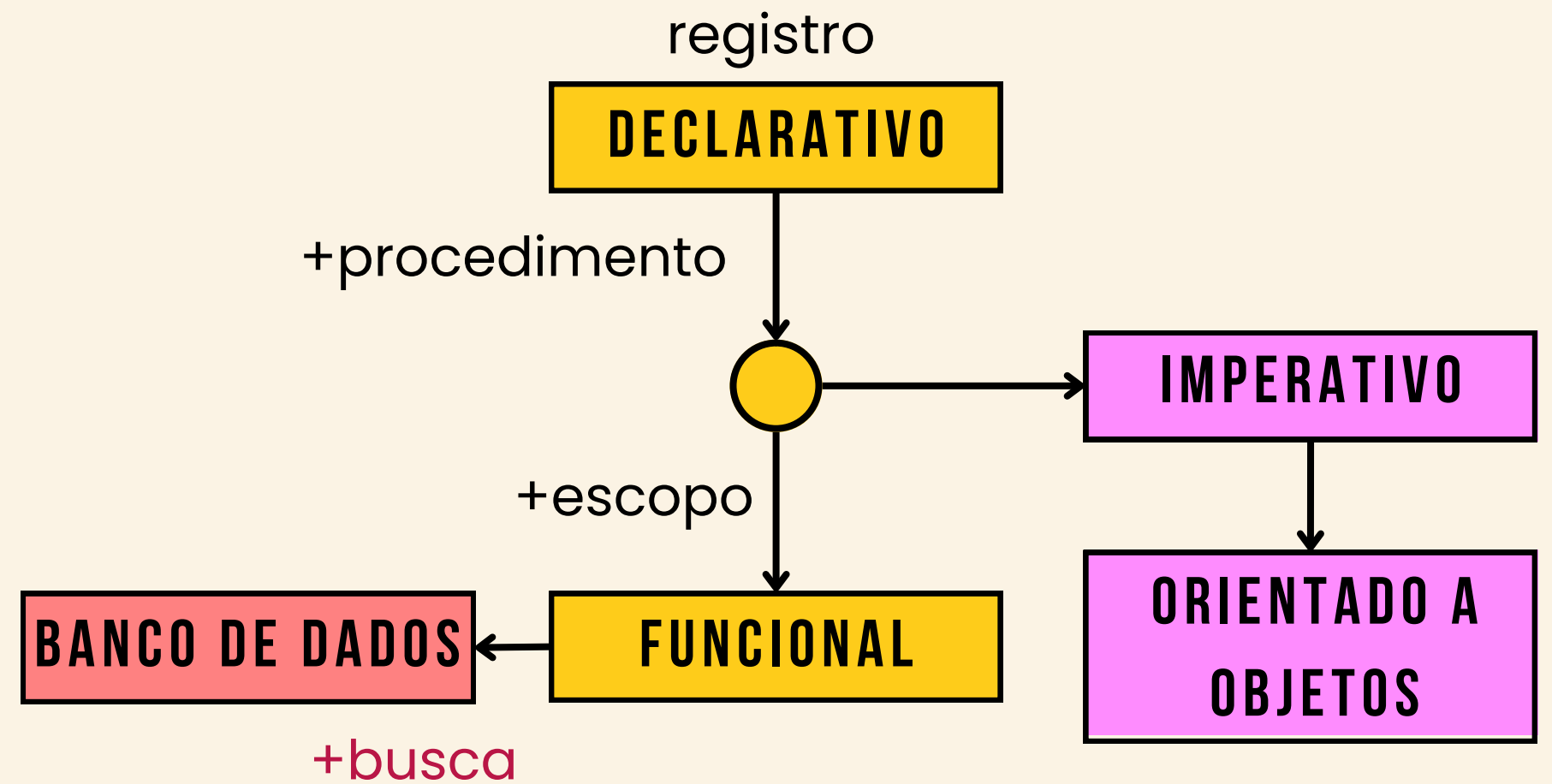
```
const myName = (name) => `Hello ${name}`
```

```
myName("Leonardo")
```

# *paradigmas de* **PROGRAMAÇÃO**



# *paradigmas de* **PROGRAMAÇÃO**





# *paradigmas de* **PROGRAMAÇÃO**

**BANCO DE DADOS**

# *paradigmas de* **PROGRAMAÇÃO**

## **BANCO DE DADOS**

Utilizado para **buscas** em **bancos de dados**

# *paradigmas de* **PROGRAMAÇÃO**

## **BANCO DE DADOS**

Utilizado para **buscas** em **bancos de dados**

Descreve o **resultado** esperado na busca

# *paradigmas de* **PROGRAMAÇÃO**

## **BANCO DE DADOS**

Utilizado para **buscas** em **bancos de dados**

Descreve o **resultado** esperado na busca

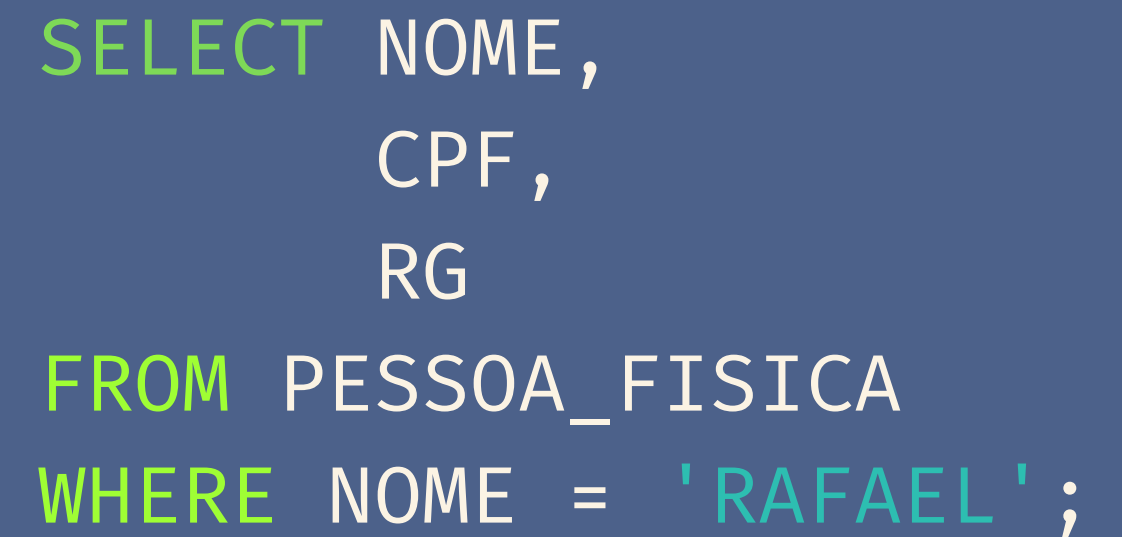
Exemplos: SQL, UnSQL

# BANCO DE DADOS

Utilizado para **buscas** em **bancos de dados**

Descreve o **resultado** esperado na busca

Exemplos: SQL, UnSQL



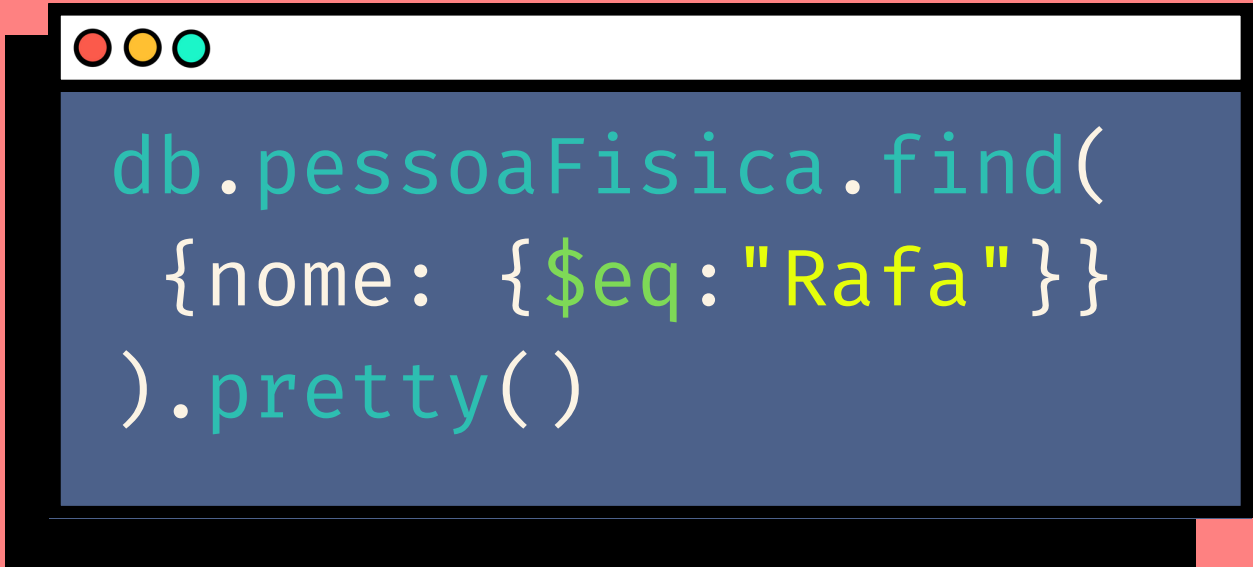
```
SELECT NOME,  
        CPF,  
        RG  
FROM PESSOA_FISICA  
WHERE NOME = 'RAFAEL';
```

# BANCO DE DADOS

Utilizado para **buscas** em **bancos de dados**

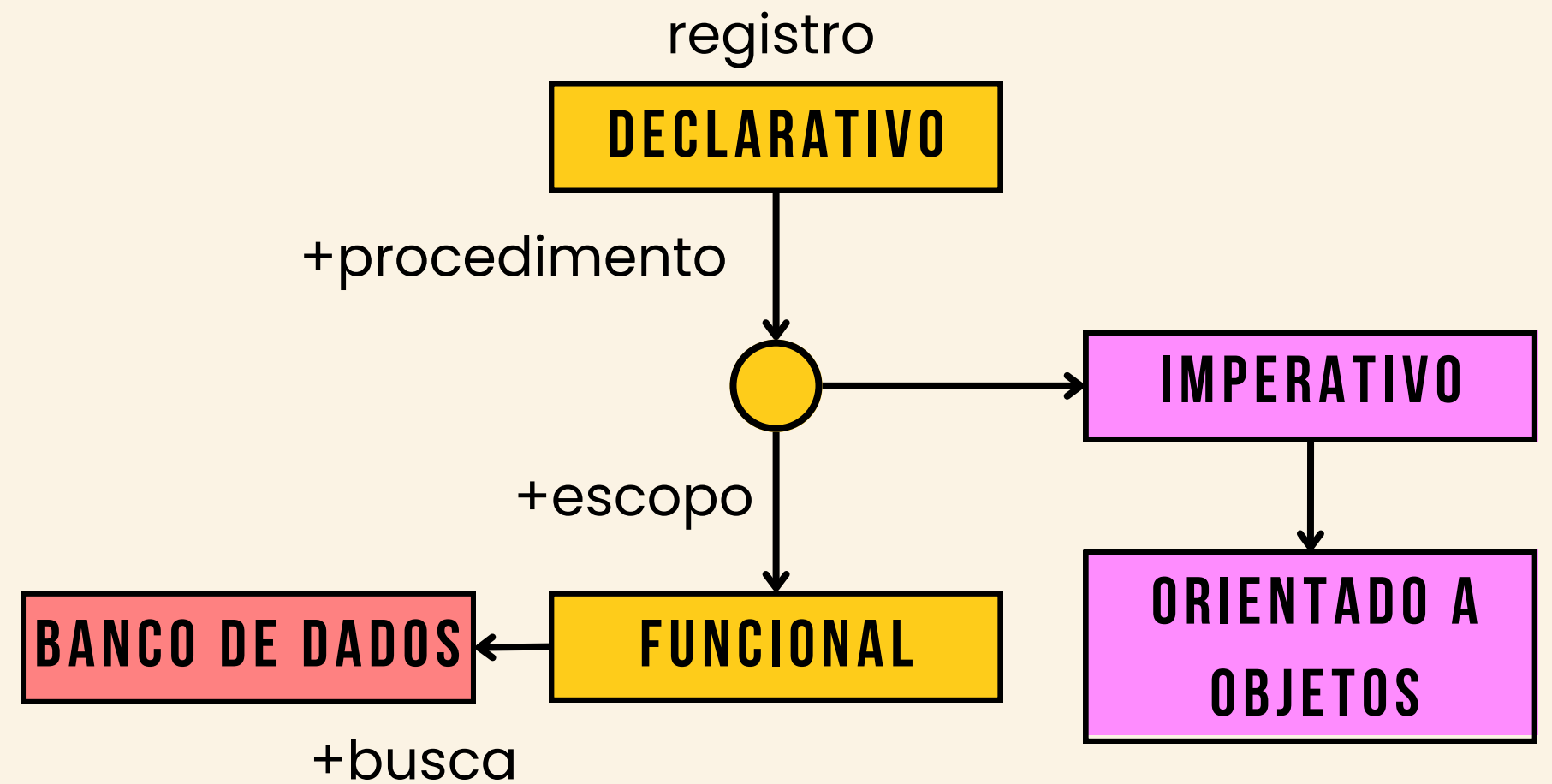
Descreve o **resultado** esperado na busca

Exemplos: SQL, UnSQL

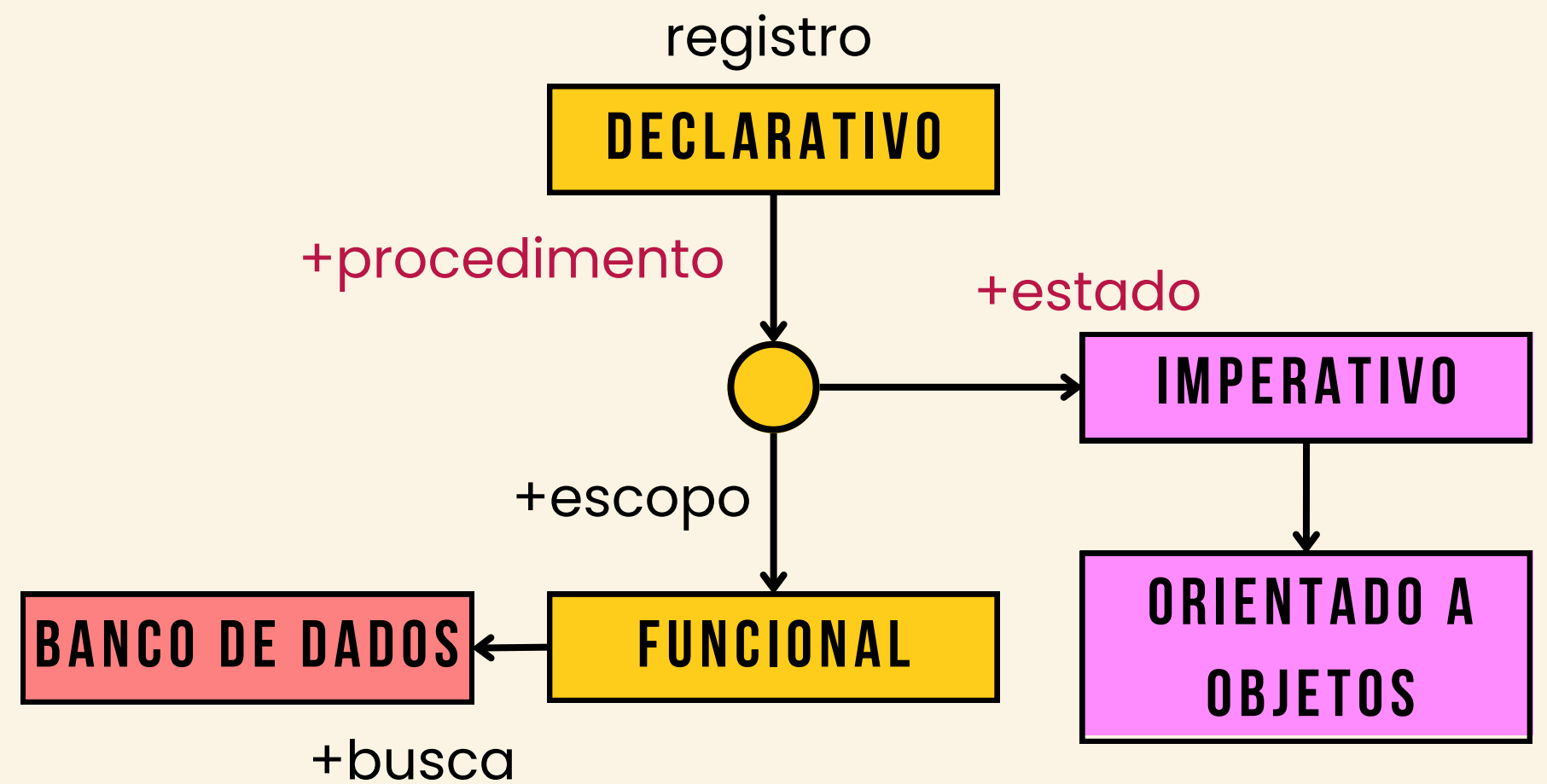


```
db.pessoaFisica.find(  
  {nome: {$eq: "Rafa"}}  
)
```

# *paradigmas de* **PROGRAMAÇÃO**



# *paradigmas de* **PROGRAMAÇÃO**





# *paradigmas de* **PROGRAMAÇÃO**

**IMPERATIVO**

# *paradigmas de* **PROGRAMAÇÃO**

**IMPERATIVO** *ou*

PROCEDURAL

# *paradigmas de* **PROGRAMAÇÃO**

PROCEDURAL

**IMPERATIVO** *ou*

Detalha **passo a passo** o processo

# *paradigmas de* **PROGRAMAÇÃO**

PROCEDURAL

## **IMPERATIVO** *ou*

Detalha **passo a passo** o processo

Baseado em uma **sequência** de instruções

# *paradigmas de* **PROGRAMAÇÃO**

PROCEDURAL

## **IMPERATIVO** *ou*

Detalha **passo a passo** o processo

Baseado em uma **sequência** de instruções

Manipula o **estado** com variáveis

# *paradigmas de* **PROGRAMAÇÃO**

PROCEDURAL

## **IMPERATIVO** *ou*

Detalha **passo a passo** o processo

Baseado em uma **sequência** de instruções

Manipula o **estado** com variáveis

Possui as instruções **if, while, switch** e **for**

# *paradigmas de* **PROGRAMAÇÃO**

PROCEDURAL

## **IMPERATIVO** *ou*

Detalha **passo a passo** o processo

Baseado em uma **sequência** de instruções

Manipula o **estado** com variáveis

Possui as instruções **if, while, switch** e **for**

Exemplos: C, Pascal, Java

# IMPERATIVO

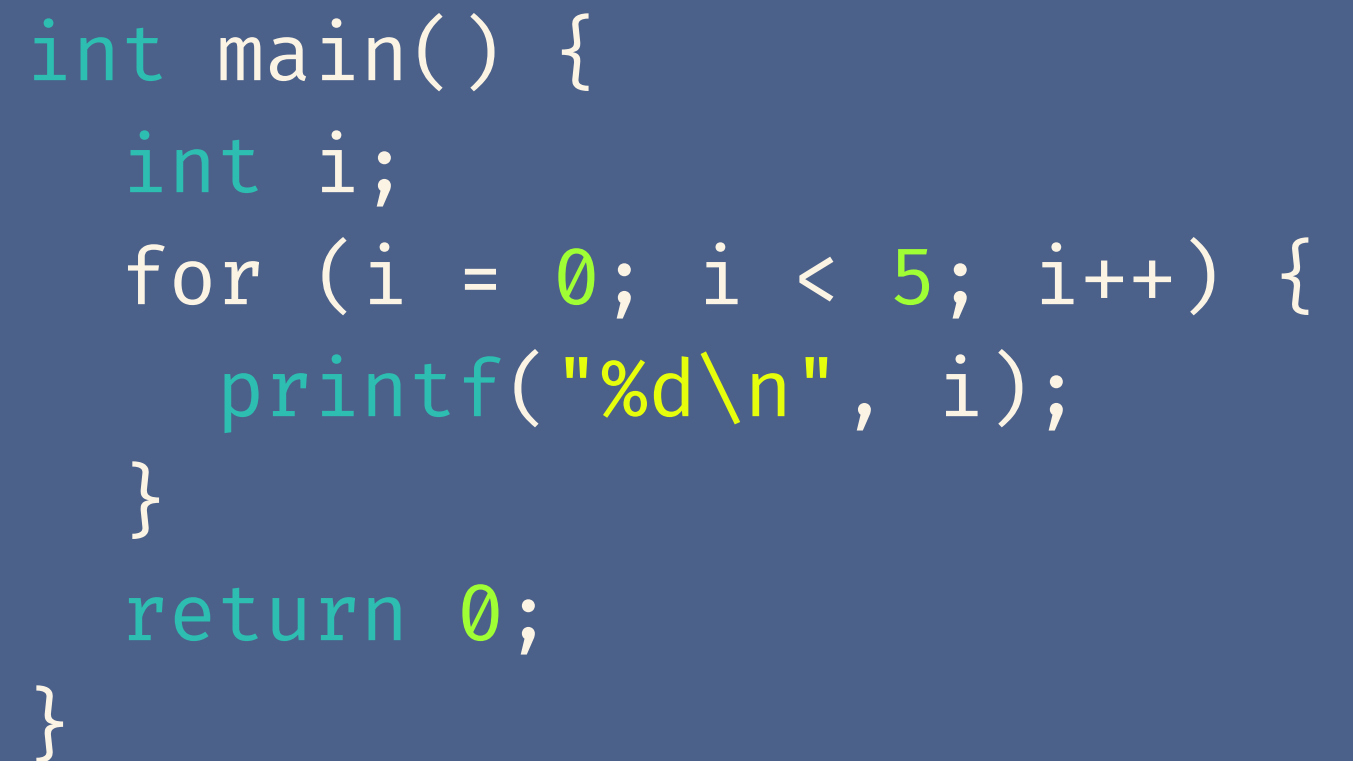
Detalha **passo a passo** o processo

Baseado em uma **sequência** de instruções

Manipula o **estado** com variáveis

Possui as instruções **if, while, switch** e **for**

Exemplos: C, Pascal, Java



```
int main() {  
    int i;  
    for (i = 0; i < 5; i++) {  
        printf("%d\n", i);  
    }  
    return 0;  
}
```



# IMPERATIVO

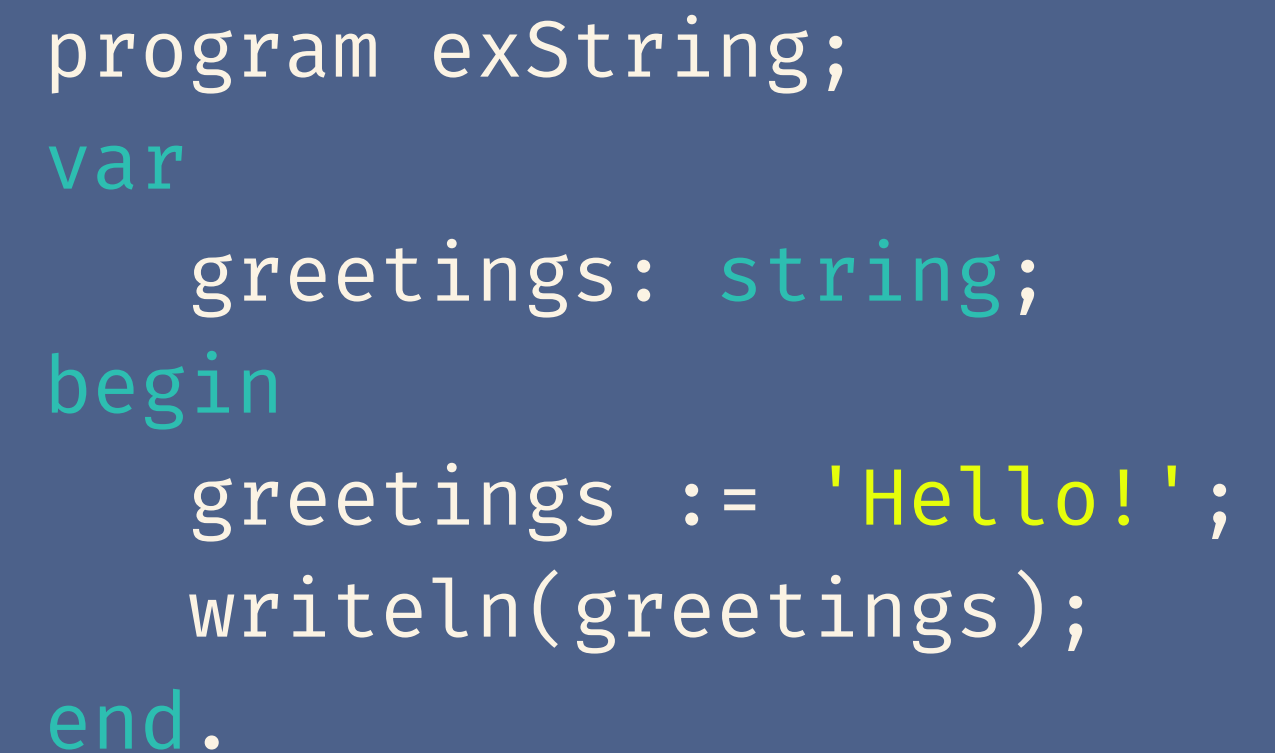
Detalha **passo a passo** o processo

Baseado em uma **sequência** de instruções

Manipula o **estado** com variáveis

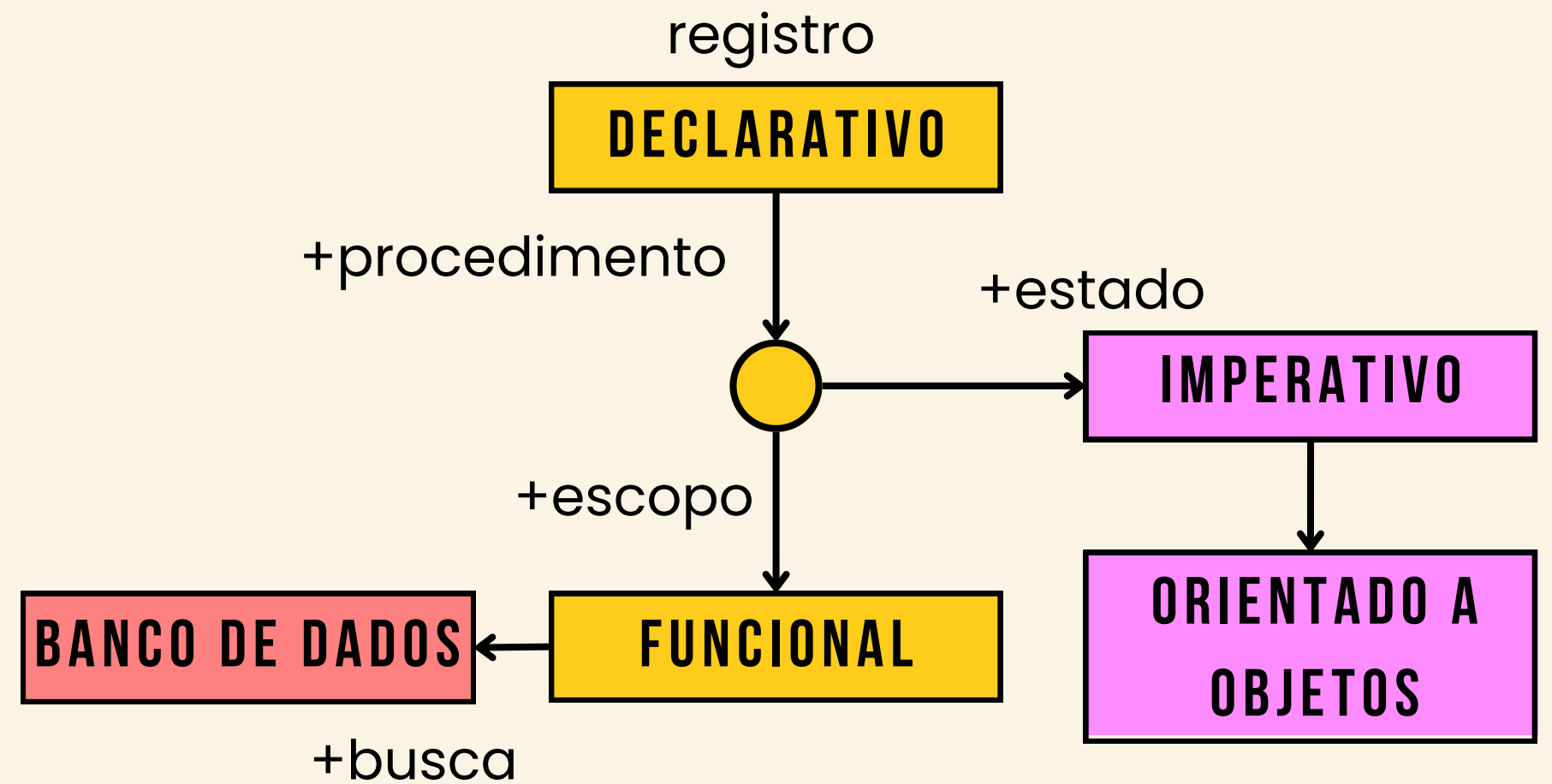
Possui as instruções **if, while, switch** e **for**

Exemplos: C, Pascal, Java

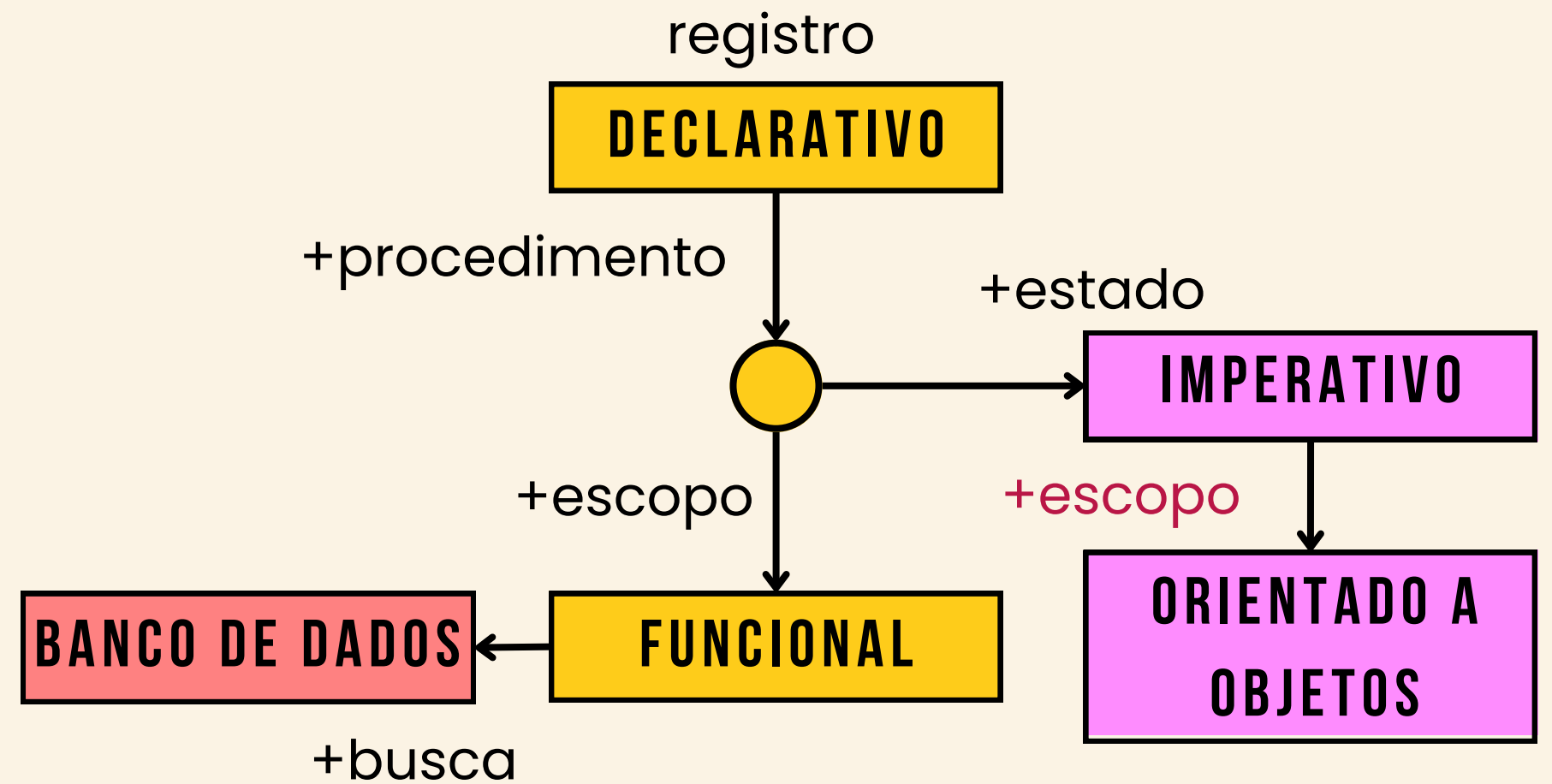


```
program exString;  
var  
    greetings: string;  
begin  
    greetings := 'Hello!';  
    writeln(greetings);  
end.
```

# *paradigmas de* **PROGRAMAÇÃO**



# *paradigmas de* **PROGRAMAÇÃO**



# *paradigmas de* **PROGRAMAÇÃO**

## ORIENTADO A OBJETOS

Baseado na interação entre **objetos**

Utiliza de **abstrações** para solucionar problemas

Busca poupar código através da **herança** de **classes**

Exemplos: Typescript, C#, Java

# ORIENTADO A OBJETOS

Baseado na interação entre **objetos**

Utiliza de **abstrações** para solucionar problemas

Busca poupar código através da **herança** de **classes**

Exemplos: Typescript, C#, Java

```
class Character {  
    protected health: number;  
    protected alive: boolean;  
  
    constructor(health: number,  
                alive: boolean = true) {  
        this.health = health;  
        this.alive = alive;  
    }  
}
```

# ORIENTADO A OBJETOS

Baseado na interação entre **objetos**

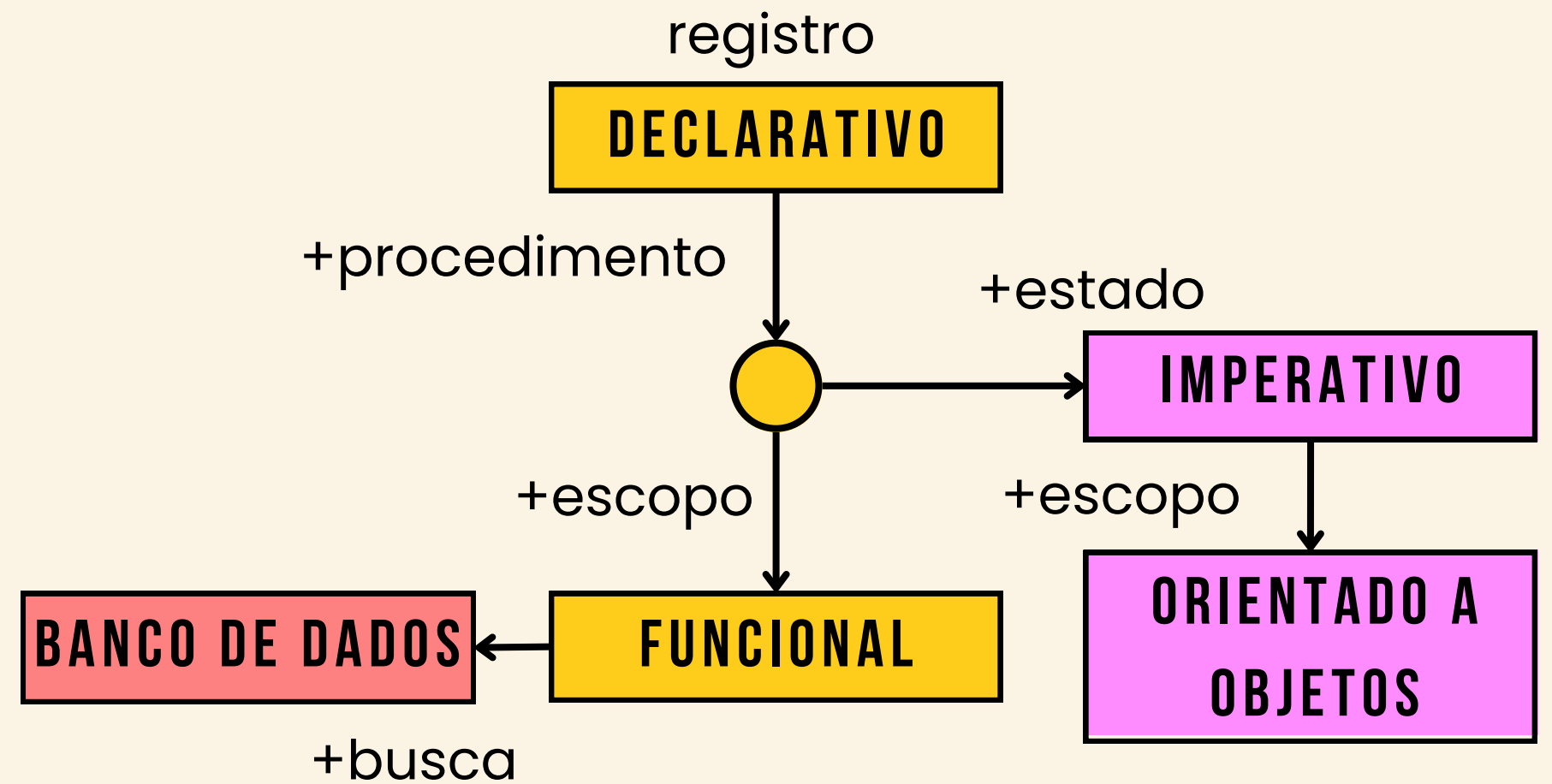
Utiliza de **abstrações** para solucionar problemas

Busca poupar código através da **herança** de **classes**

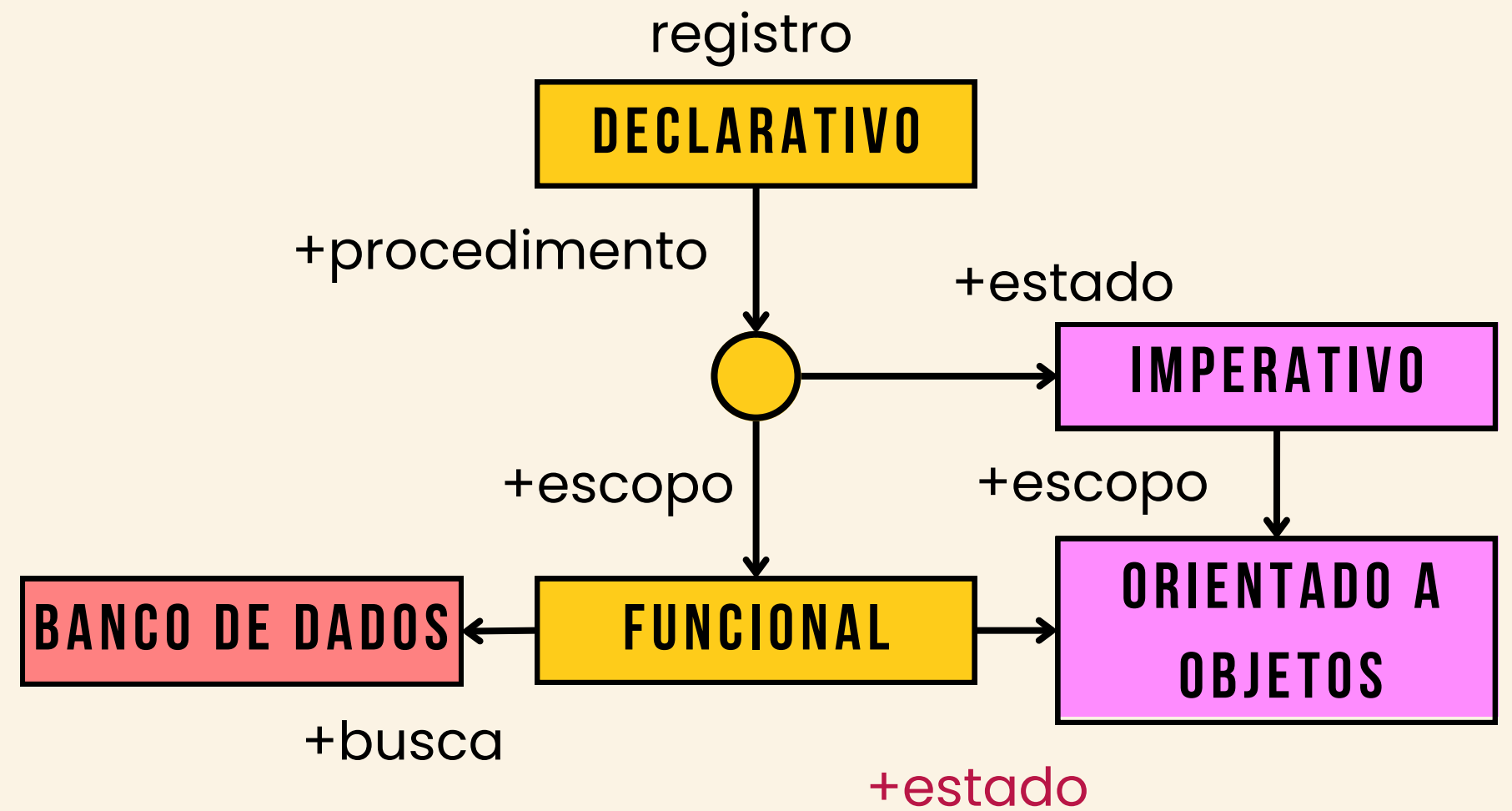
Exemplos: Typescript, C#, Java

```
public class PessoaFisica {  
    protected String nome;  
    protected Integer idade;  
  
    PessoaFisica(String nome,  
                  Integer idade) {  
        this.nome = nome;  
        this.idade = idade;  
    }  
}
```

# *paradigmas de* **PROGRAMAÇÃO**



# *paradigmas de* **PROGRAMAÇÃO**





---

*programação*  
**ORIENTADA**  
*a* **OBJETOS**

---

# *programação* **ORIENTADA** *a* **OBJETOS**



**ABSTRAÇÃO**



**ENCAPSULAMENTO**



**HERANÇA**



**POLIMORFISMO**

*programação*  
**ORIENTADA**  
*a* **OBJETOS**



A diagram showing four OOP concepts arranged in a 2x2 grid. The top row contains 'ABSTRAÇÃO' and 'ENCAPSULAMENTO', while the bottom row contains 'HERANÇA' and 'POLIMORFISMO'. The text is in bold black capital letters. The grid is divided by a horizontal line. The top-left cell has a pink oval around the text, with a pink square at the top and a yellow square at the bottom. The top-right cell has a teal square at the top and a pink square at the bottom. The bottom-left cell has a yellow square at the top and a pink square at the bottom. The bottom-right cell has a pink square at the top and a pink square at the bottom.

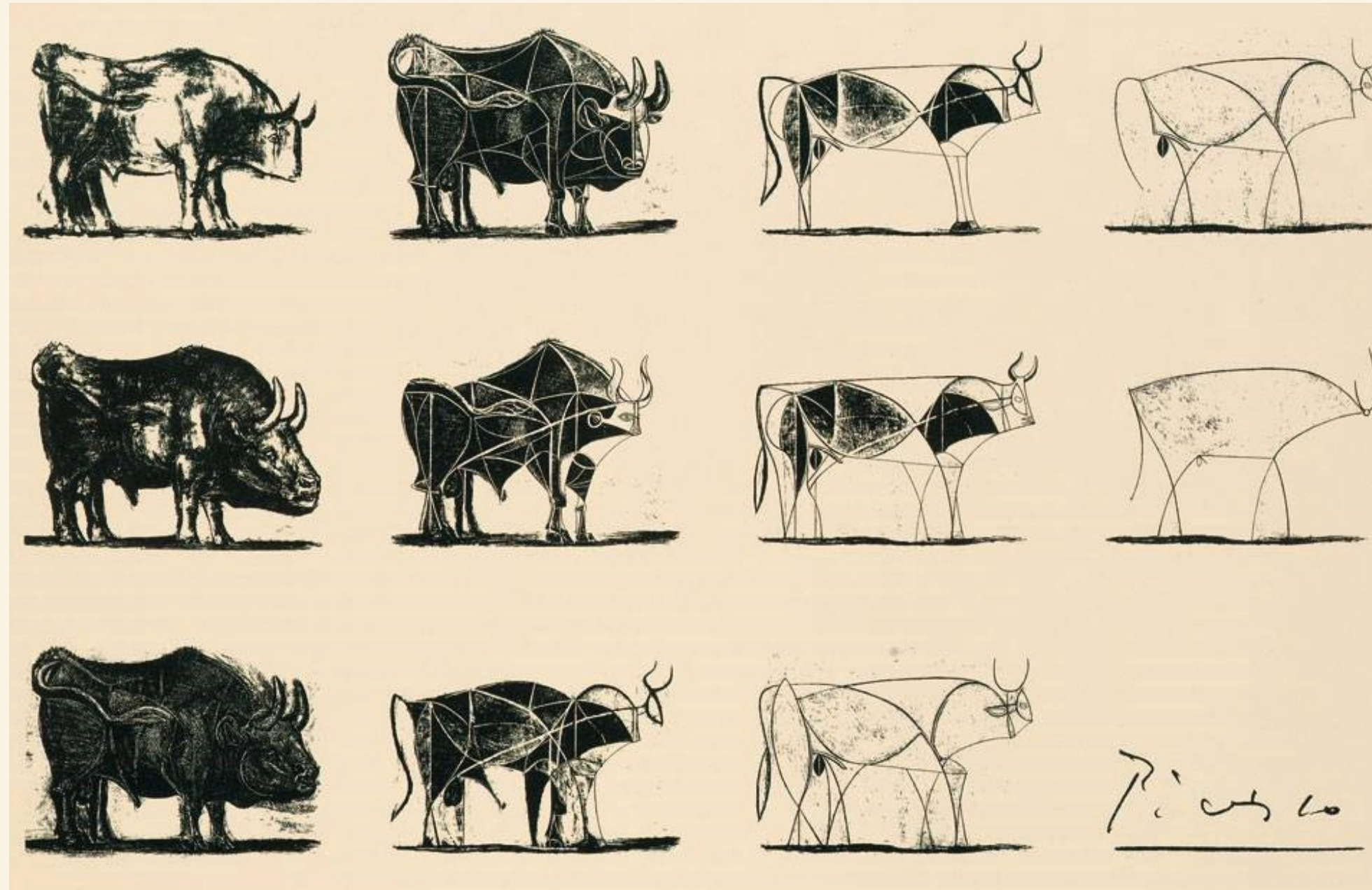
**ABSTRAÇÃO**

**ENCAPSULAMENTO**

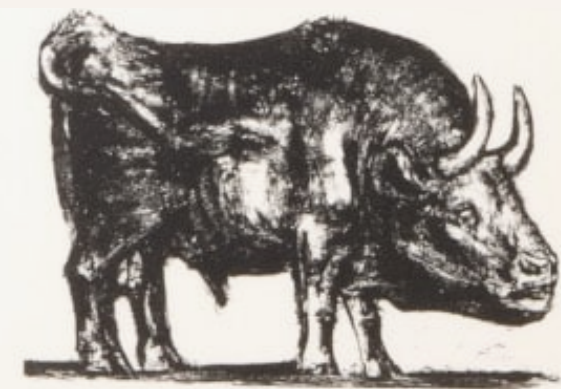
**HERANÇA**

**POLIMORFISMO**

# ABSTRAÇÃO

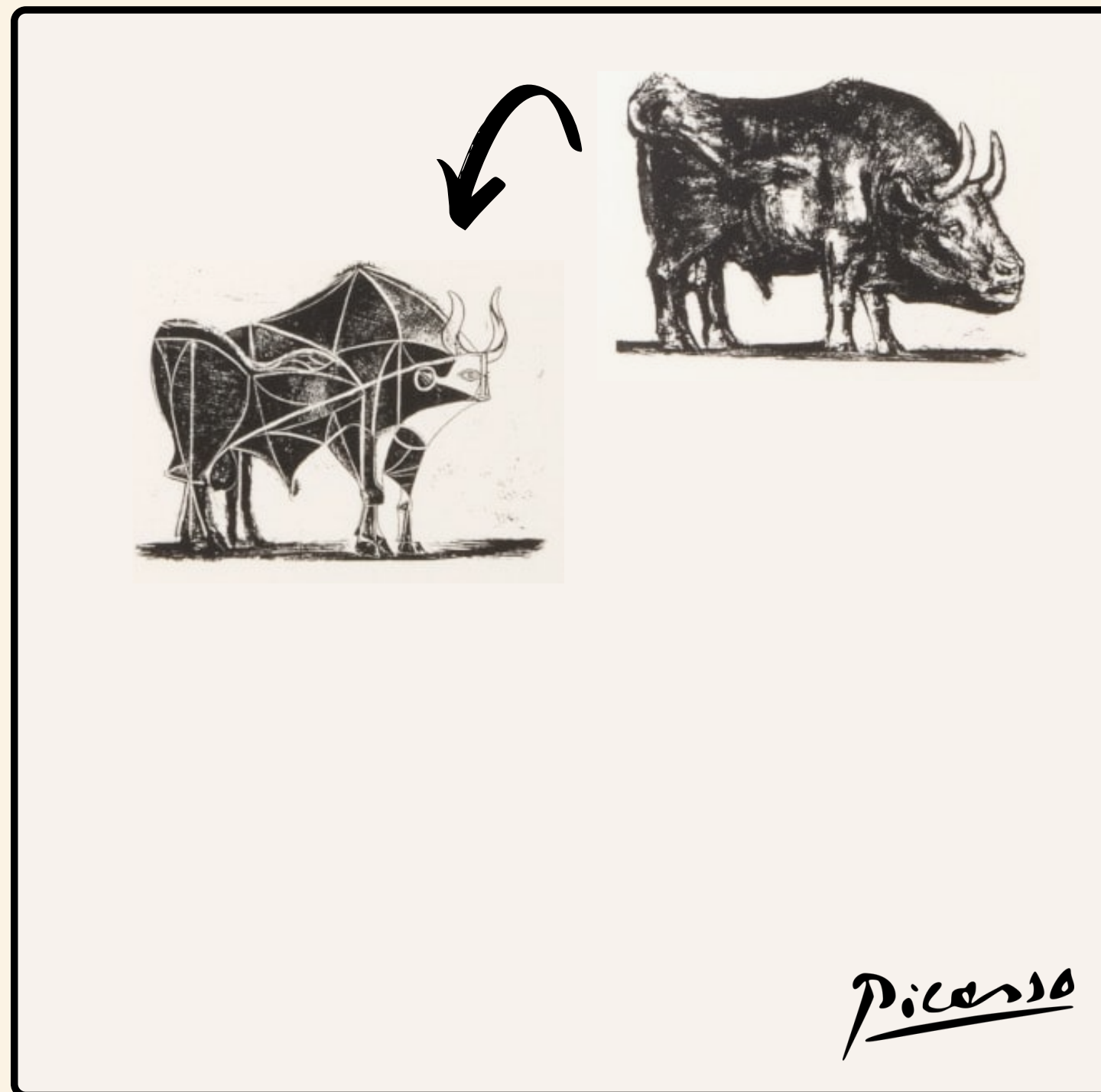


# ABSTRAÇÃO



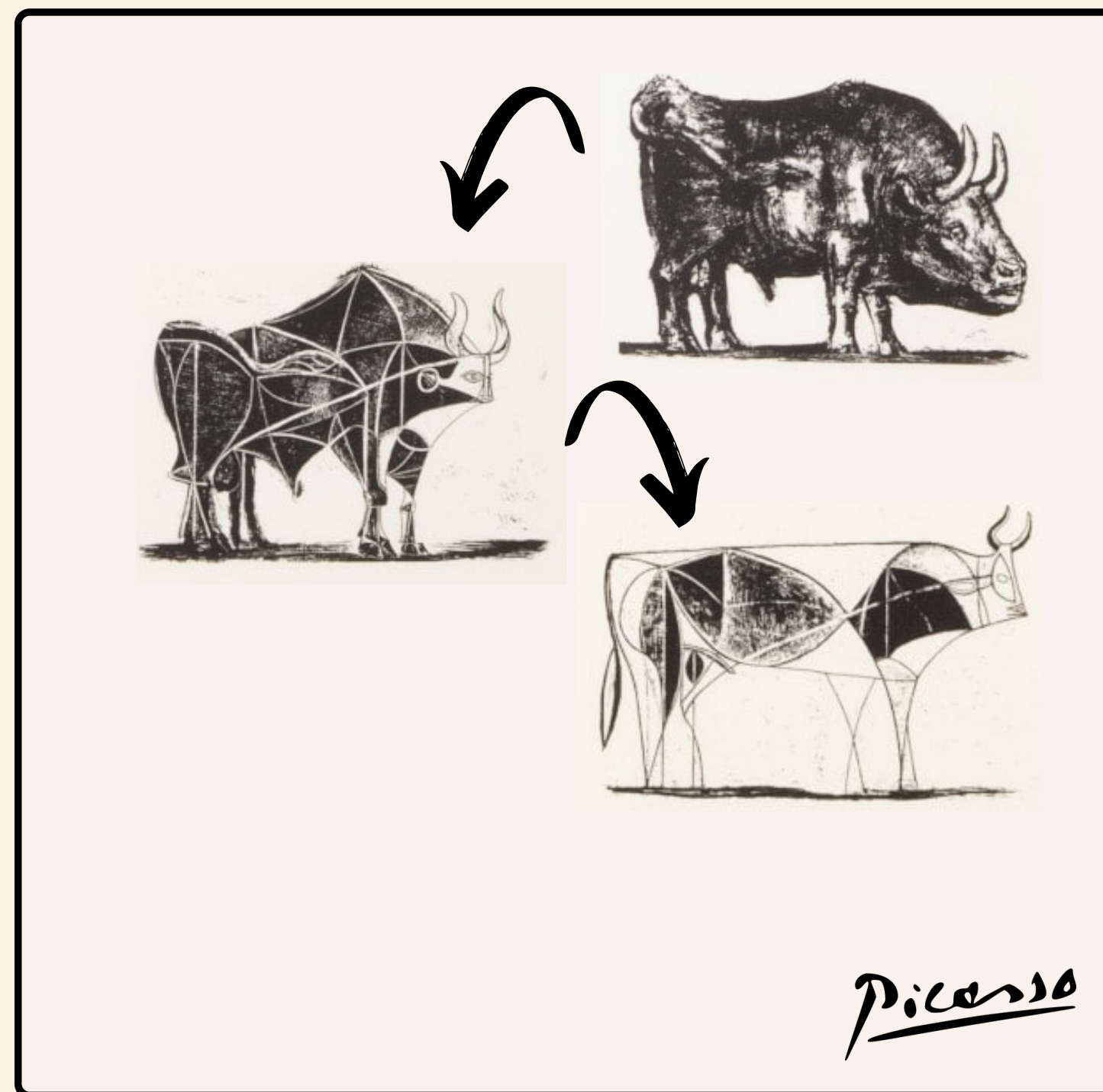
*Picasso*

# ABSTRAÇÃO

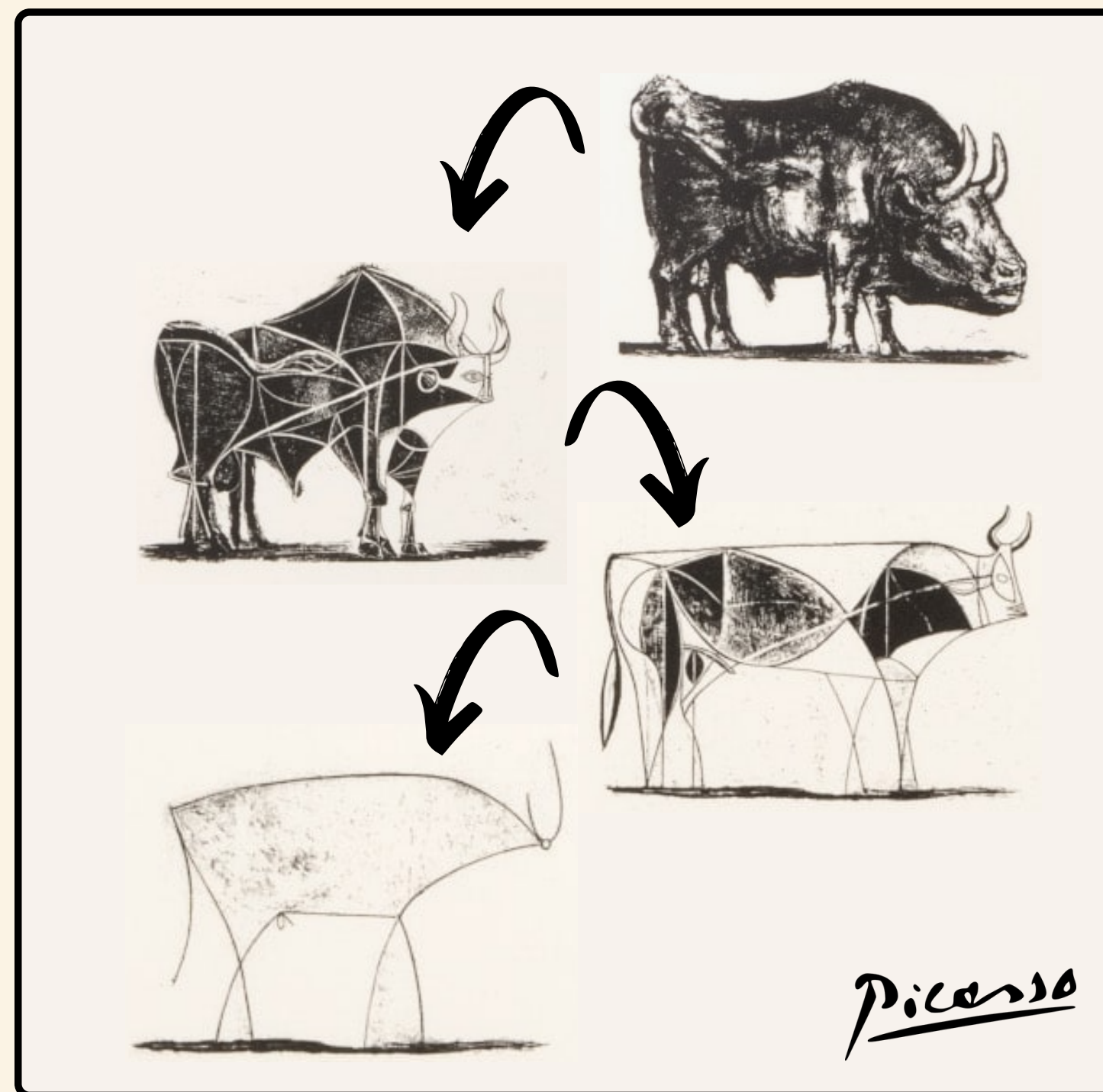




# ABSTRAÇÃO



# ABSTRAÇÃO





# ABSTRAÇÃO



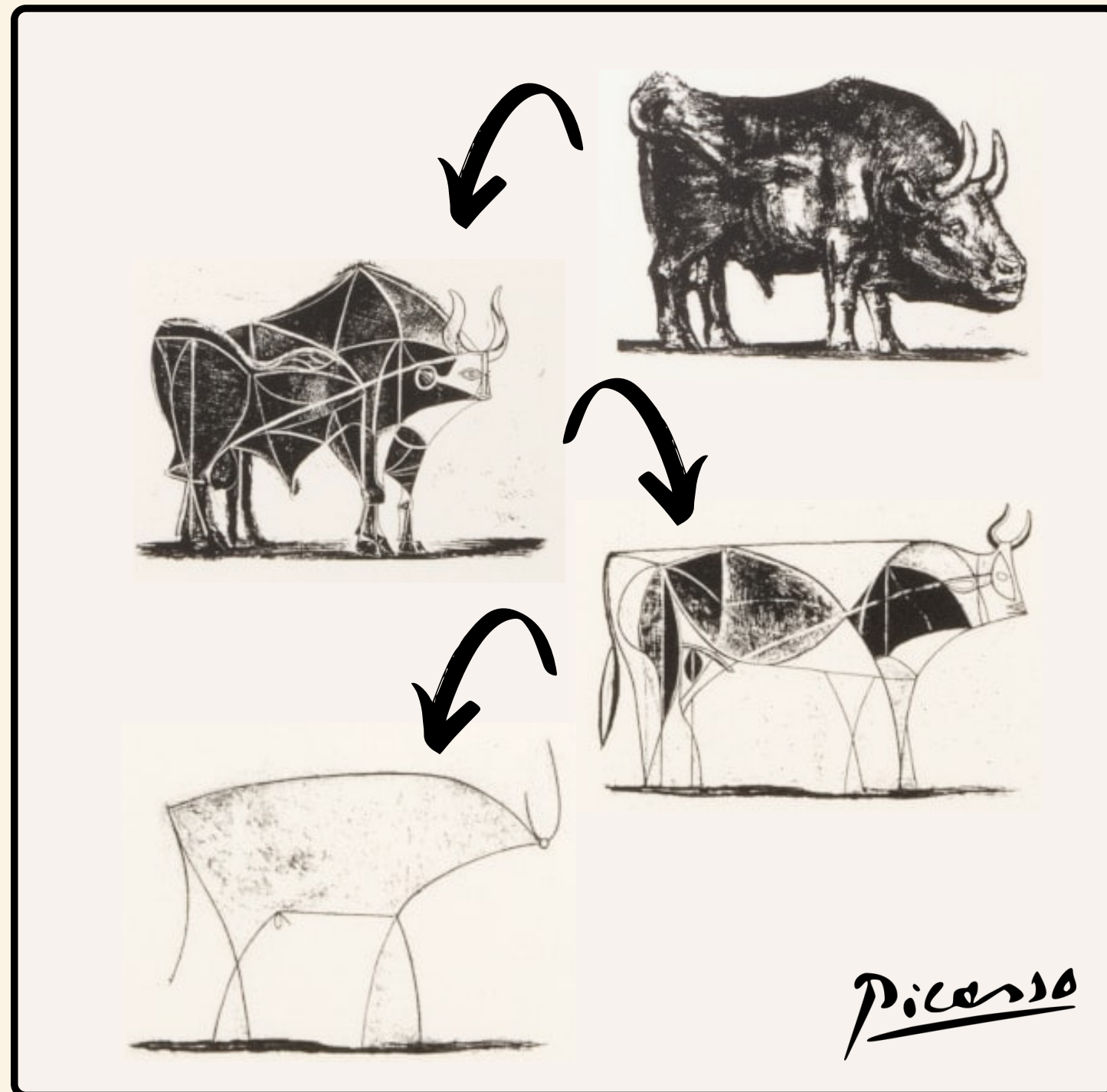
SIMULAÇÃO



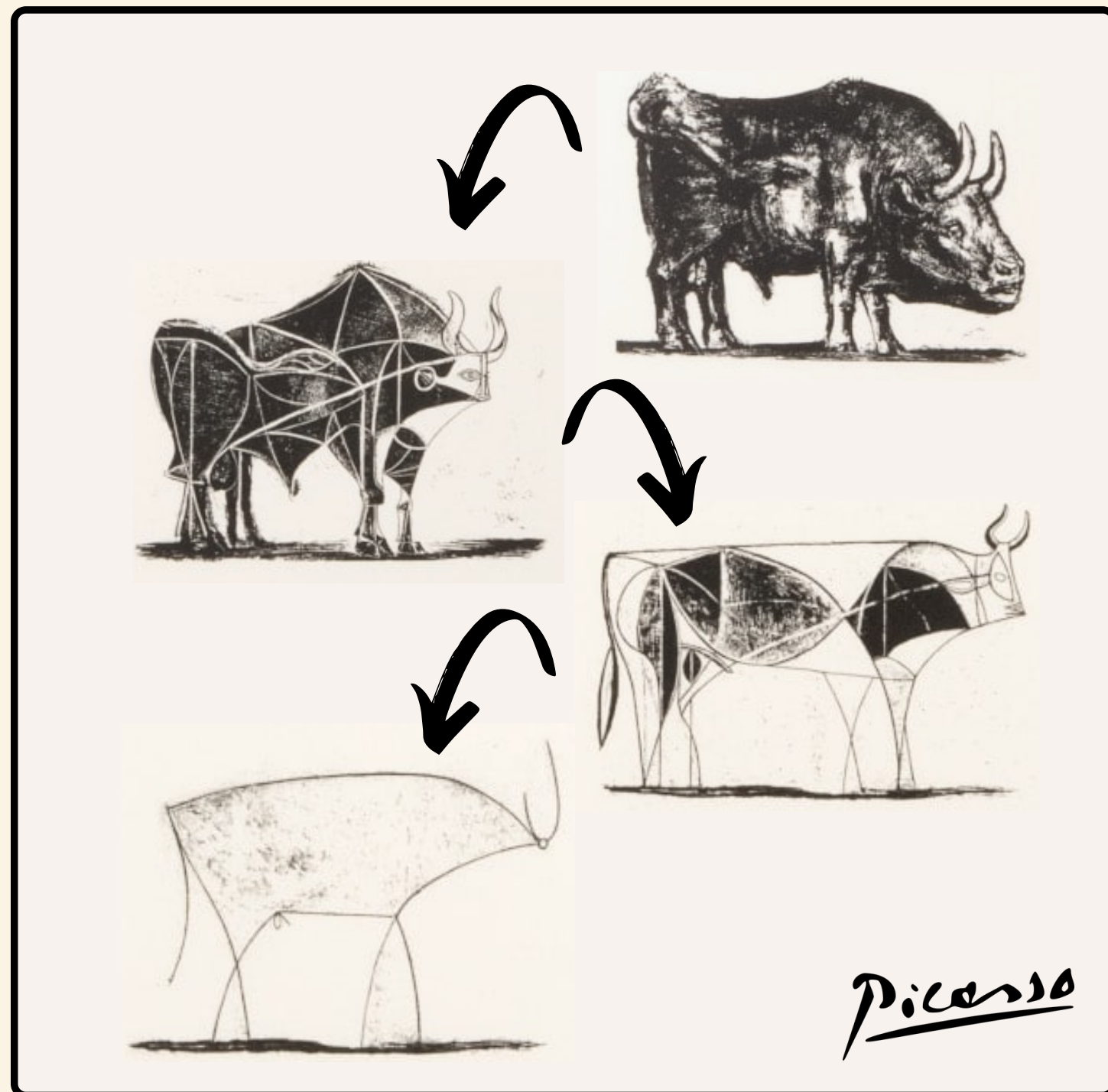
OBJETOS



CLASSES



# ABSTRAÇÃO



## SIMULAÇÃO

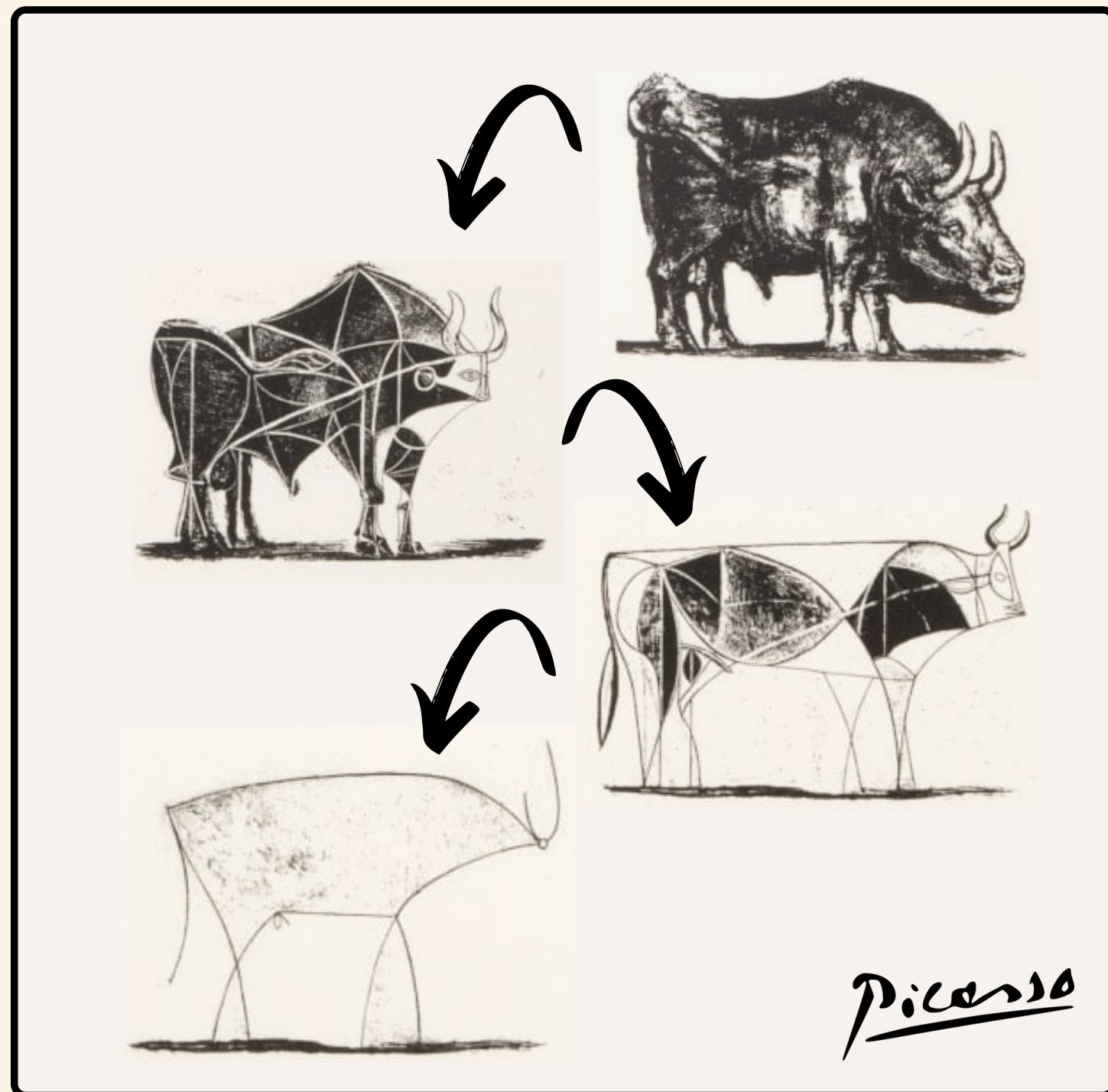
A POO tem como objetivo simular PROCESSOS REAIS a partir da ABSTRAÇÃO de elementos

## OBJETOS

São representações dos elementos contidos no processo a ser simulado. São utilizados para INTERAGIREM com outros objetos SIMULANDO o processo real

## CLASSES

# ABSTRAÇÃO



## SIMULAÇÃO

A POO tem como objetivo simular PROCESSOS REAIS a partir da ABSTRAÇÃO de elementos

## OBJETOS

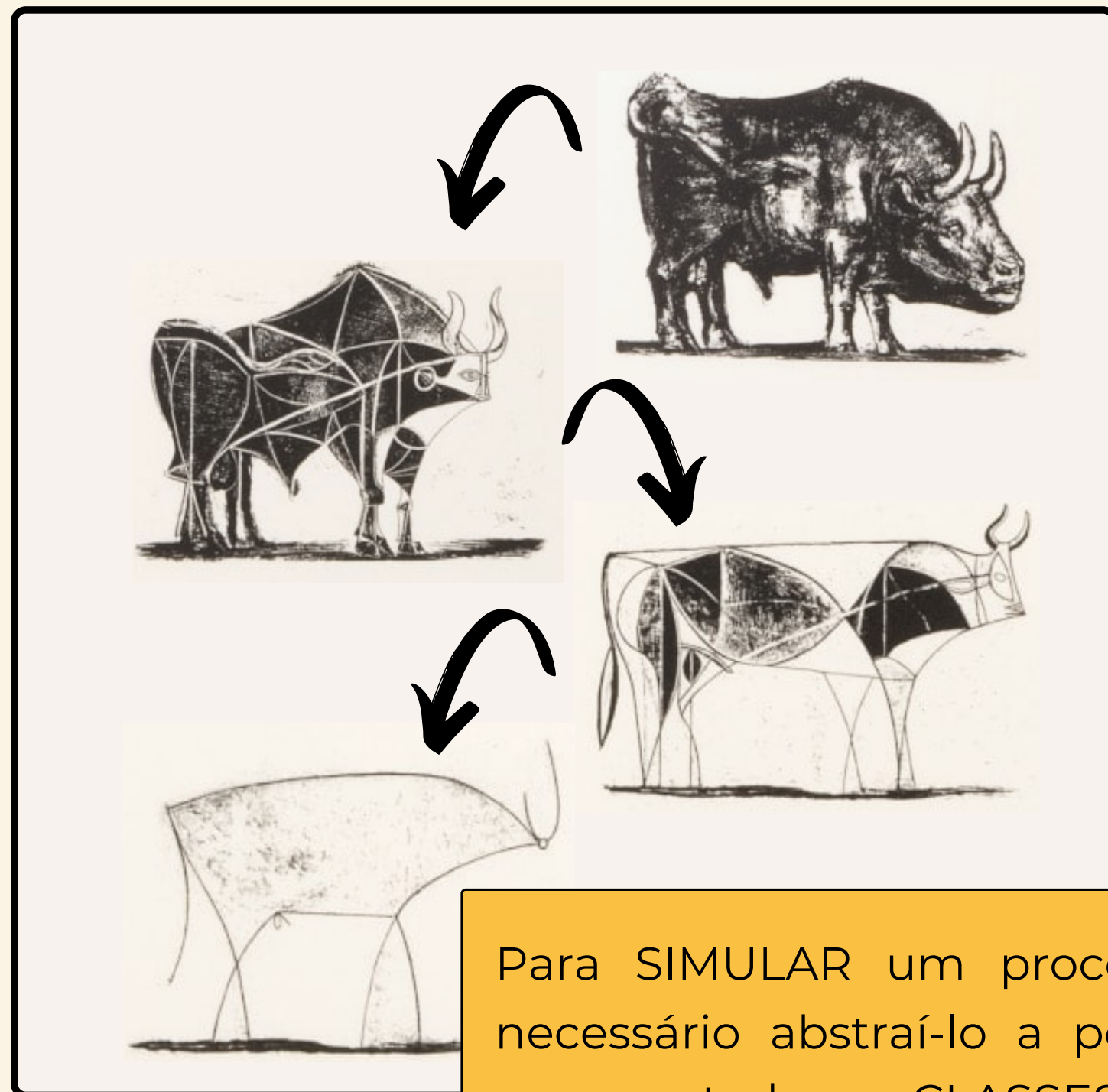
São representações dos elementos contidos no processo a ser simulado. São utilizados para INTERAGIREM com outros objetos SIMULANDO o processo real

## CLASSES

São ABSTRAÇÕES aplicadas aos OBJETOS. Um objeto é a forma "CONCRETA" de uma classe



# ABSTRAÇÃO



Para SIMULAR um processo real, é necessário abstraí-lo a ponto de ser representado por CLASSES e OBJETOS

## SIMULAÇÃO

A POO tem como objetivo simular PROCESSOS REAIS a partir da ABSTRAÇÃO de elementos

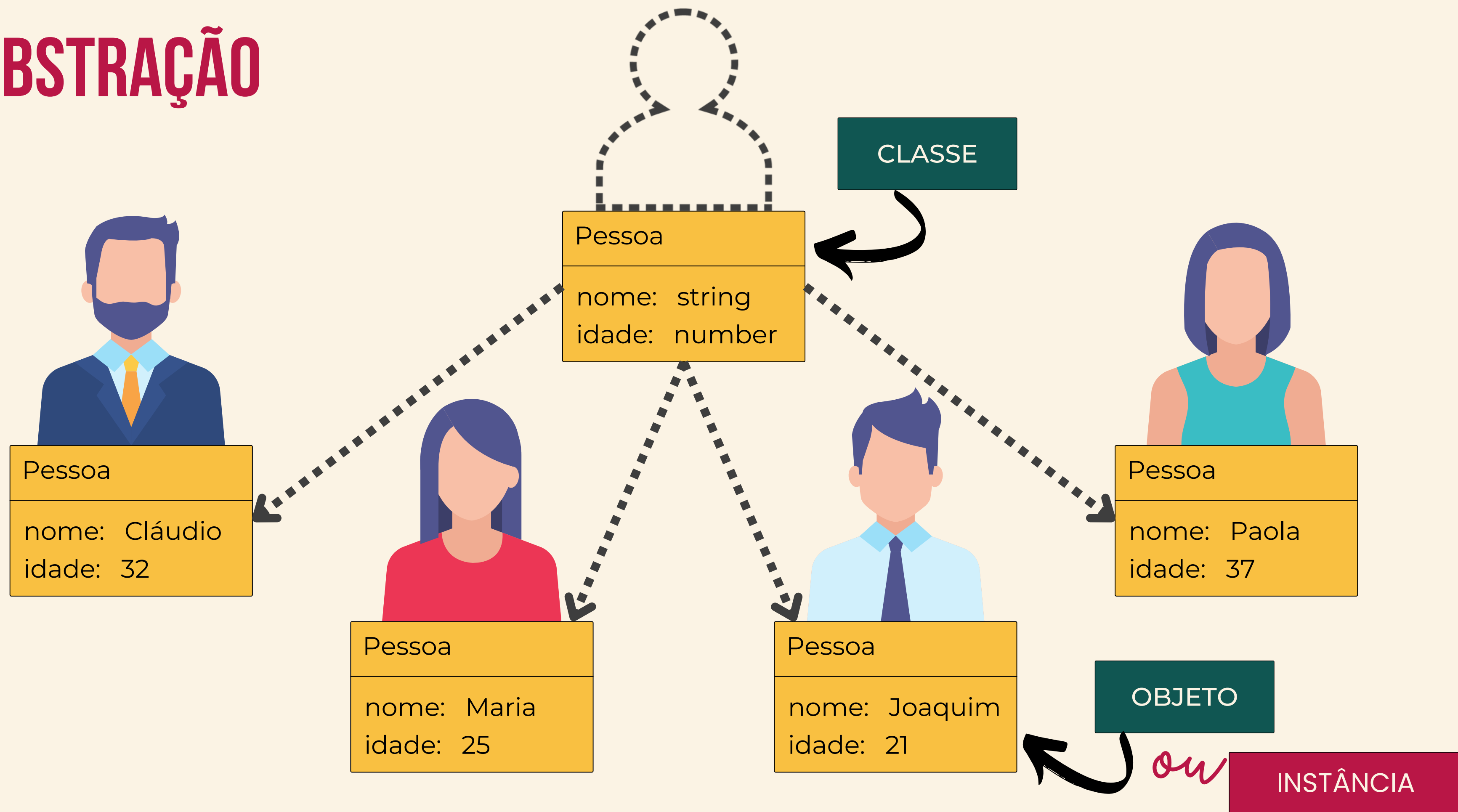
## OBJETOS

São representações dos elementos contidos no processo a ser simulado. São utilizados para INTERAGIREM com outros objetos SIMULANDO o processo real

## CLASSES

São ABSTRAÇÕES aplicadas aos OBJETOS. Um objeto é a forma "CONCRETA" de uma classe

# ABSTRAÇÃO

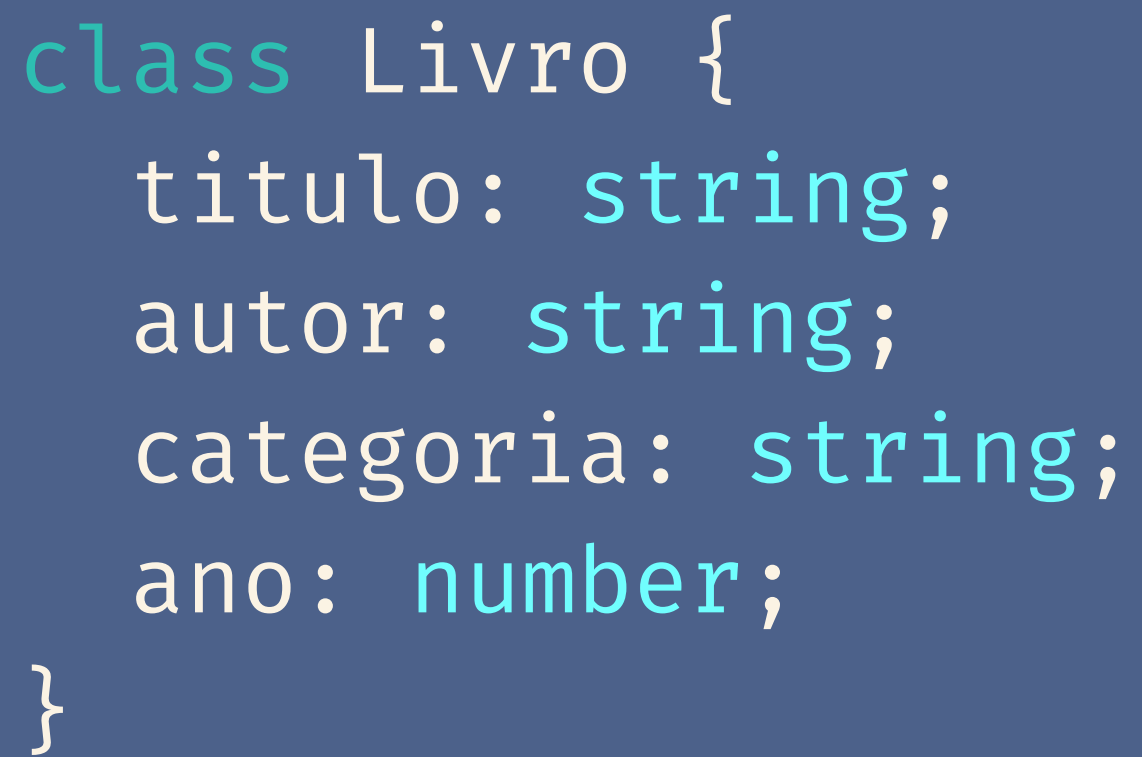


**ABSTRAÇÃO**

**SINTAXE**

---

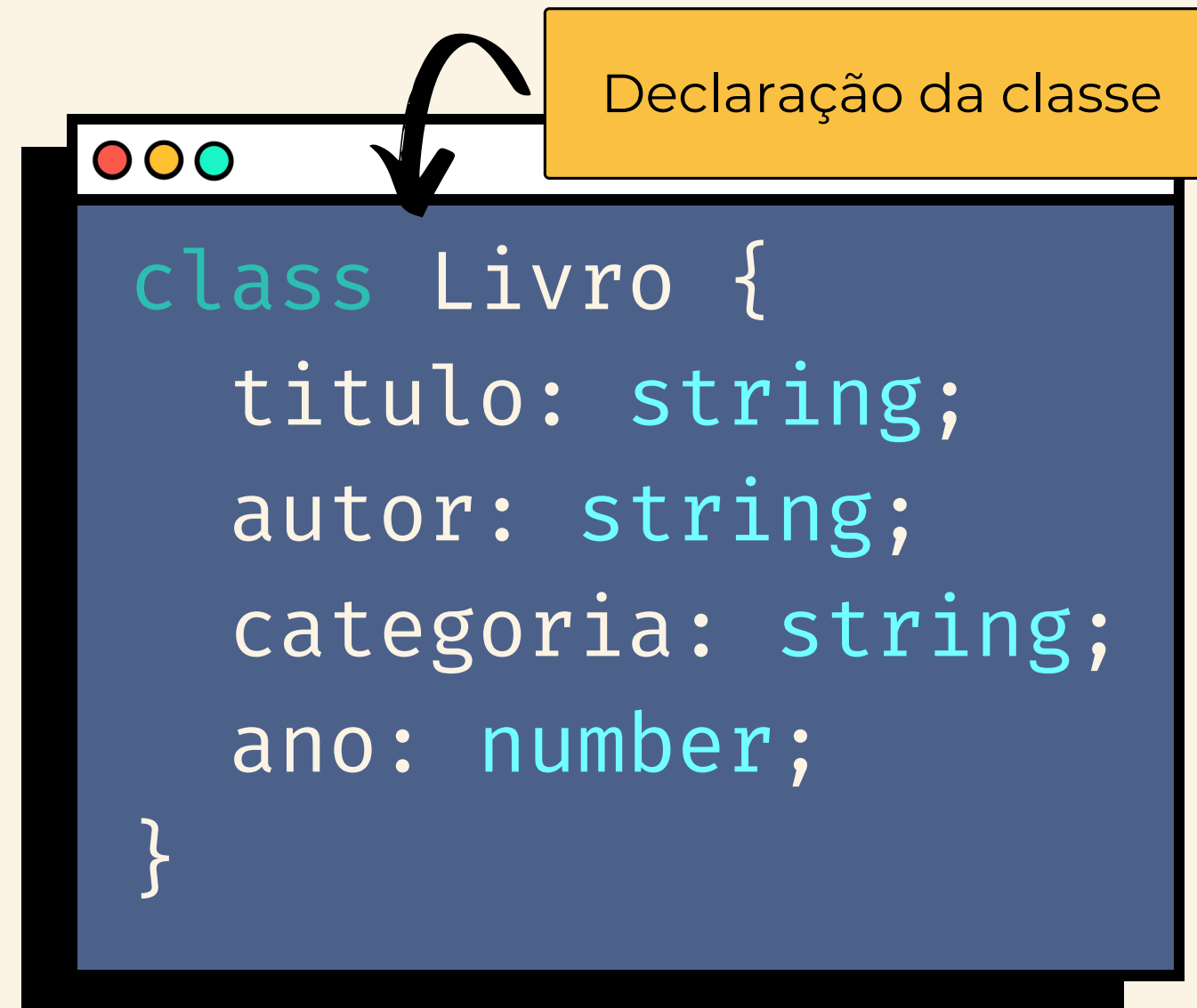
# ABSTRAÇÃO



```
class Livro {  
    titulo: string;  
    autor: string;  
    categoria: string;  
    ano: number;  
}
```

# CLASSE

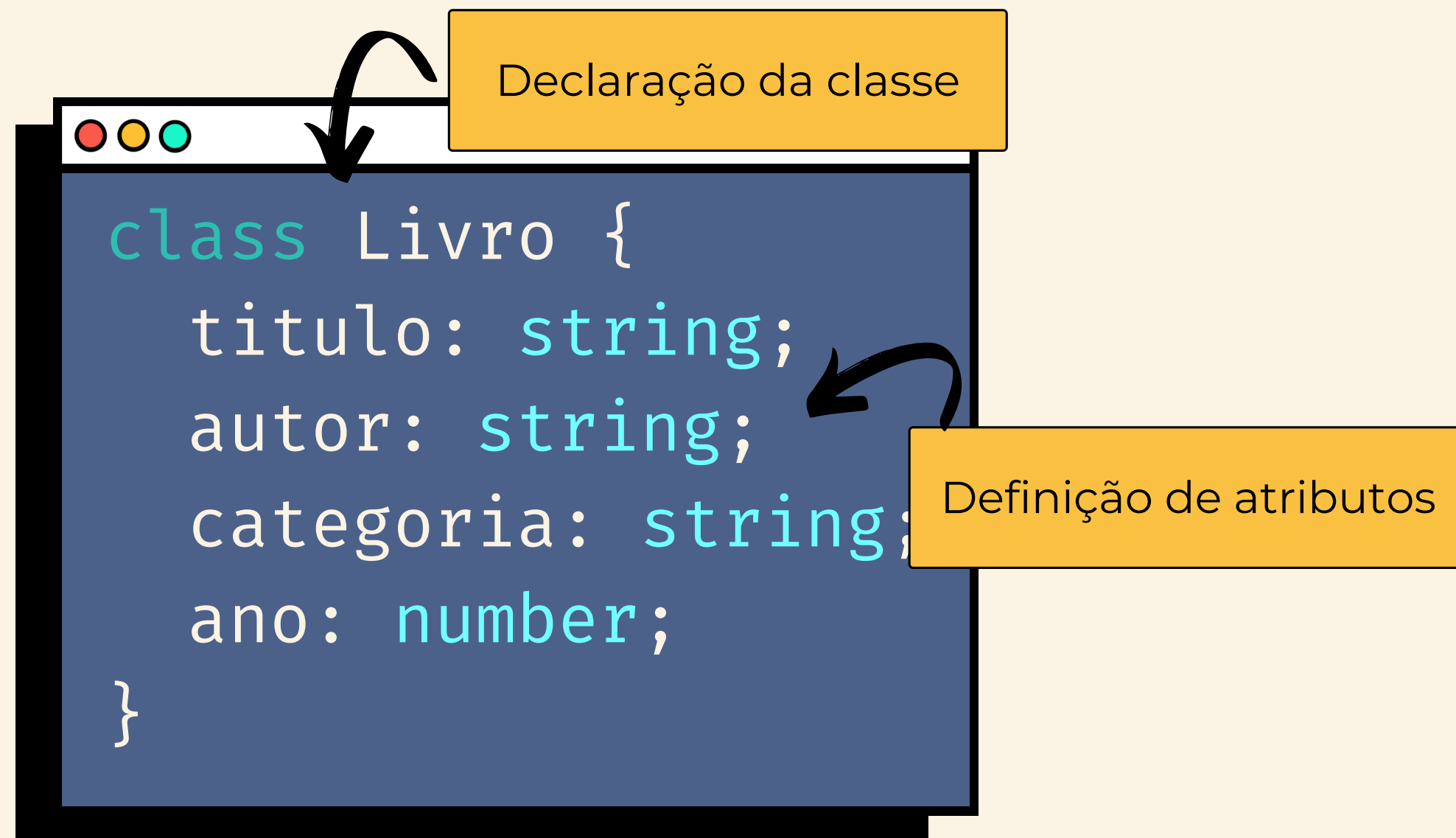
# ABSTRAÇÃO



# CLASSE



# ABSTRAÇÃO



CLASSE

# ABSTRAÇÃO

```
class Livro {  
    titulo: string;  
    autor: string;  
    categoria: string;  
    ano: number;  
}
```

```
class Autor {  
    nome: string;  
    cpf: string;  
}
```

CLASSE

# ABSTRAÇÃO

```
class Livro {  
  titulo: string;  
  autor: Autor;  
  categoria: string;  
  ano: number;  
}
```

Classes como tipo de dado

```
class Autor {  
  nome: string;  
  cpf: string;  
}
```

# CLASSE

# ABSTRAÇÃO

```
let autor1 = new Autor(  
    "Adroaldo da Silva",  
    "013.851.920-03"  
);  
let livro1 = new Livro(  
    "Um Livro qualquer",  
    autor1,  
    "Poesia",  
    2014  
);
```

OBJETO

# *introdução à* **ORIENTAÇÃO** *a* **OBJETOS**

**REVISÃO**

**PARADIGMAS DE PROGRAMAÇÃO**

**ORIENTAÇÃO A OBJETOS**

**ABSTRAÇÃO**

# DESAFIO

## BIBLIOTECA

# DESAFIO

Defina os elementos para uma biblioteca com loja de livros novos inclusa. Alguns elementos são cruciais:

- **Livro** - Utilizado para referenciar os livros disponíveis
- **Autor** - Representando o cadastro de autores
- **Leitor** - pessoa que pega livros emprestados
- **Loja** - precisa ter um conjunto de livros a serem vendidos
- **Biblioteca** - precisa ter um conjunto de livros a serem emprestados
- **Empréstimo** - precisa ter o livro emprestado, o leitor e a data do empréstimo

# BIBLIOTECA



**OBRIGADO!**



# FEEDBACK

**ACESSE**

**WWW.MENTI.COM**

**INSIRA O CÓDIGO**

**2311 1468**



ou use o QR code