

*introdução à*  
**ORIENTAÇÃO**  
*a* **OBJETOS #3**

BY RAFA

# *introdução à* **ORIENTAÇÃO** *a* **OBJETOS** **#3**

**REVISÃO**

**ENCAPSULAMENTO**

**BIBLIOTECA PARTE #2**

# PQ USAR ENCAPSULAMENTO?



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



A diagram showing four OOP concepts arranged in a 2x2 grid. The top row contains 'ABSTRAÇÃO' and 'ENCAPSULAMENTO', and the bottom row contains 'HERANÇA' and 'POLIMORFISMO'. Each concept is centered in its respective quadrant. Small colored squares (pink, teal, yellow, pink) are placed at the intersections of the grid lines. A hand-drawn pink oval encircles the word 'ENCAPSULAMENTO'.

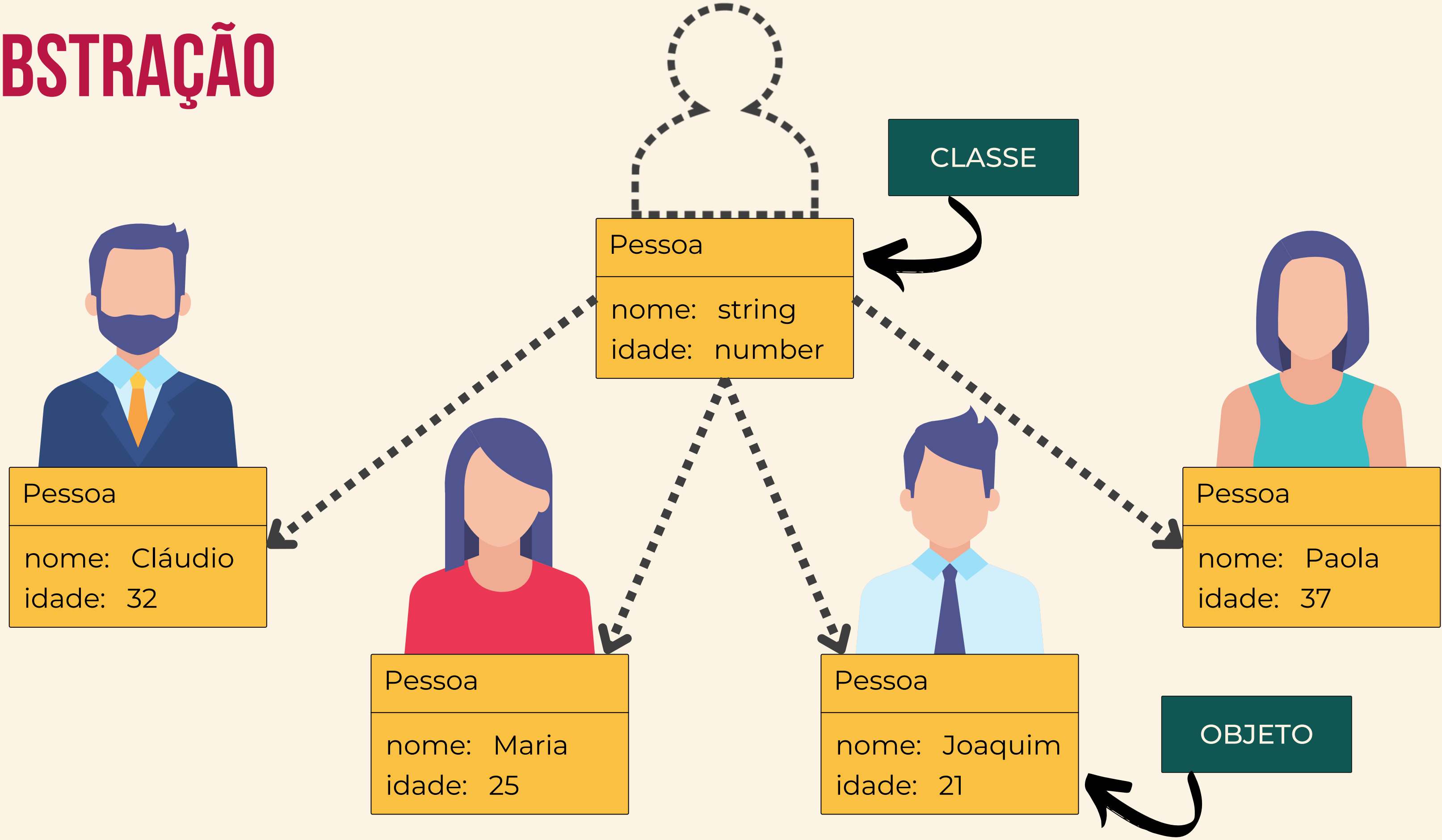
**ABSTRAÇÃO**

**ENCAPSULAMENTO**

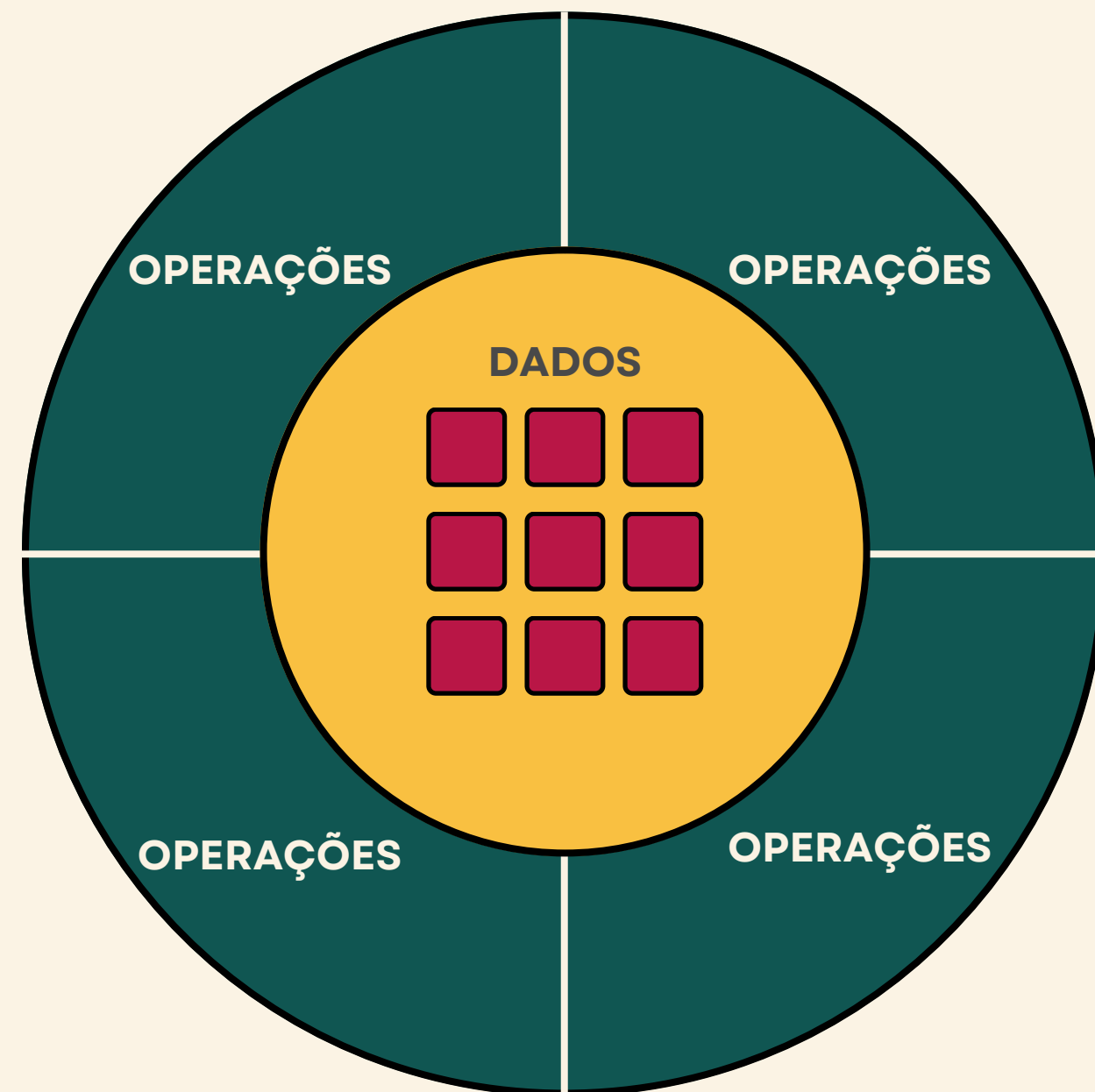
**HERANÇA**

**POLIMORFISMO**

# ABSTRAÇÃO



# ENCAPSULAMENTO



Biblioteca
nome: string livros: Array<Livro> emprestimos: Array<Emprestimo>
getLivros(): Array<Livro> addLivro(livro: Livro): void empresta(livro: Livro, leitor: Leitor): void

# ENCAPSULAMENTO

```
class Biblioteca{
  nome: string;
  livros: Array<Livro>;
  emprestimos: Array<Emprestimo>;

  constructor(nome: string,
               livros: Array<Livro> = [],
               emprestimos: Array<Emprestimo> = []) {
    this.nome = nome;
    this.livros = livros;
    this.emprestimos = emprestimos;
  }

  getLivros(): Array<Livro> {
    return this.livros;
  }
  addLivro(livro: Livro): void {
    this.livros.push(livro);
  }
  empresta(data: string, livro: Livro,
           emprestimo: Emprestimo): void {
    this.emprestimos.push(
      new Emprestimo(data, livro, emprestimo));
  }
}
```

```
let biblioteca: Biblioteca =
  new Biblioteca("Biblioteca Municipal");

let leitor: Leitor = new Leitor("Claudio Silva");

let livro: Livro = new Livro("Enciclopédia Barsa II");

biblioteca.addLivro(livro);

console.log(biblioteca.getLivros());
console.log(biblioteca.livros);

biblioteca.empresta("17/08/2022", livro, leitor);
```

**ABSTRAÇÃO**

# **INFORMAÇÃO**

---

# **EXTRA**



# ABSTRAÇÃO

# GETTER

Utilizado para **criar** uma instância de uma **classe**. Tem como objetivo, **inicializar** um objeto populando os atributos da classe.

Caso **não declarado**, é considerada com um construtor **vazio**, ou seja, **sem argumentos**, inicializando um objeto cujos atributos serão todos **undefined**

# SETTER

Uma referência à **instância** corrente, ou seja, ao **objeto** em questão. Usado para referenciar os **atributos** e **métodos** daquela instância

```
class Autor{  
  nome: string;  
  cpf: string;  
  
  constructor(nome: string,  
              cpf: string) {  
    this.nome = nome;  
    this.cpf= cpf;  
  }  
}
```

# ABSTRAÇÃO

Declaração da classe

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

Definição de atributos

Classes como tipo de dado

```
  constructor(titulo: string, autor: Autor,  
              categoria: string, ano: number) {
```

Definição do construtor

Referência ao escopo  
dos atributos

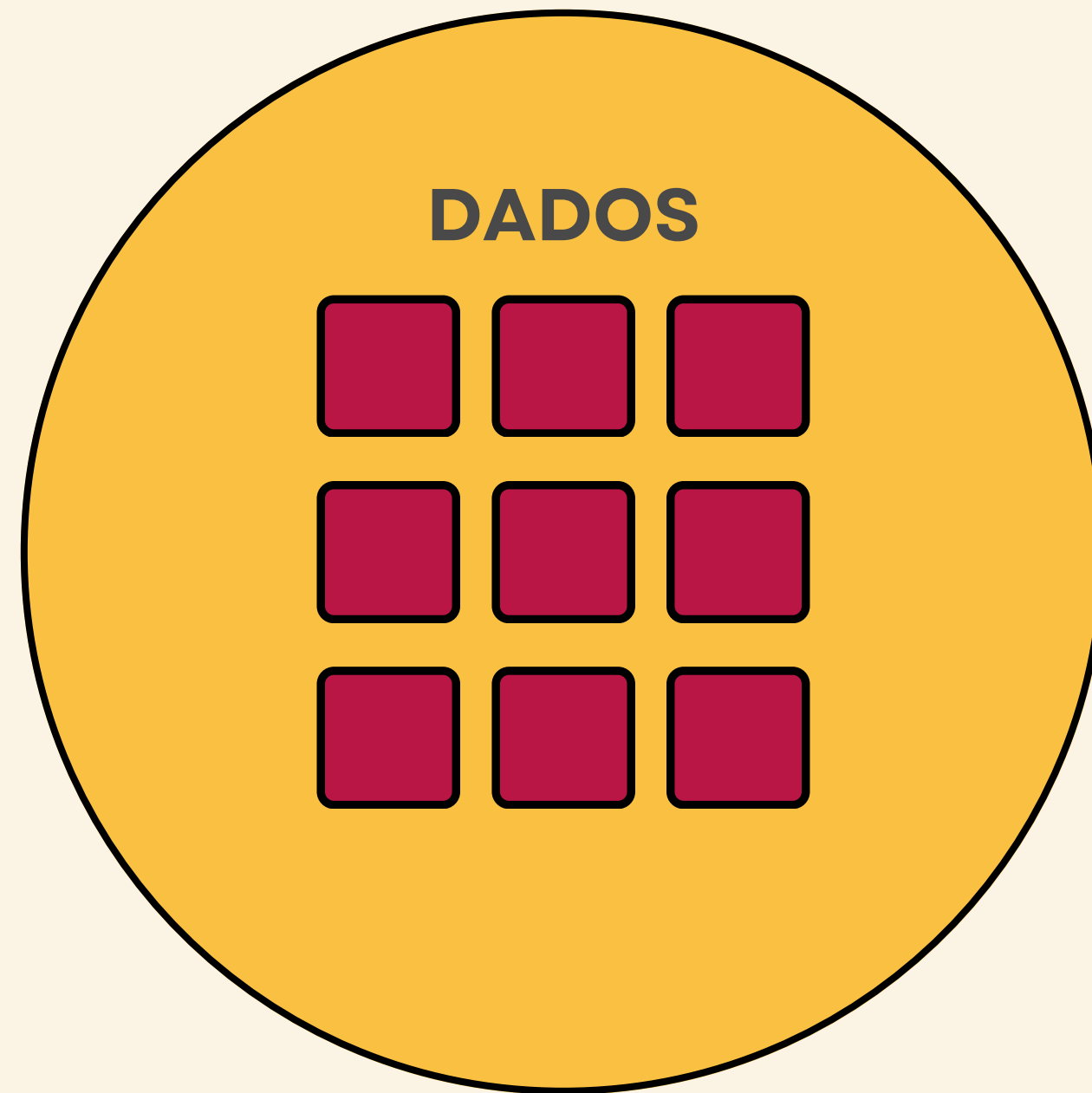
```
    this.titulo = titulo;  
    this.autor = autor;  
    this.categoria = categoria;  
    this.ano = ano;
```

```
}
```

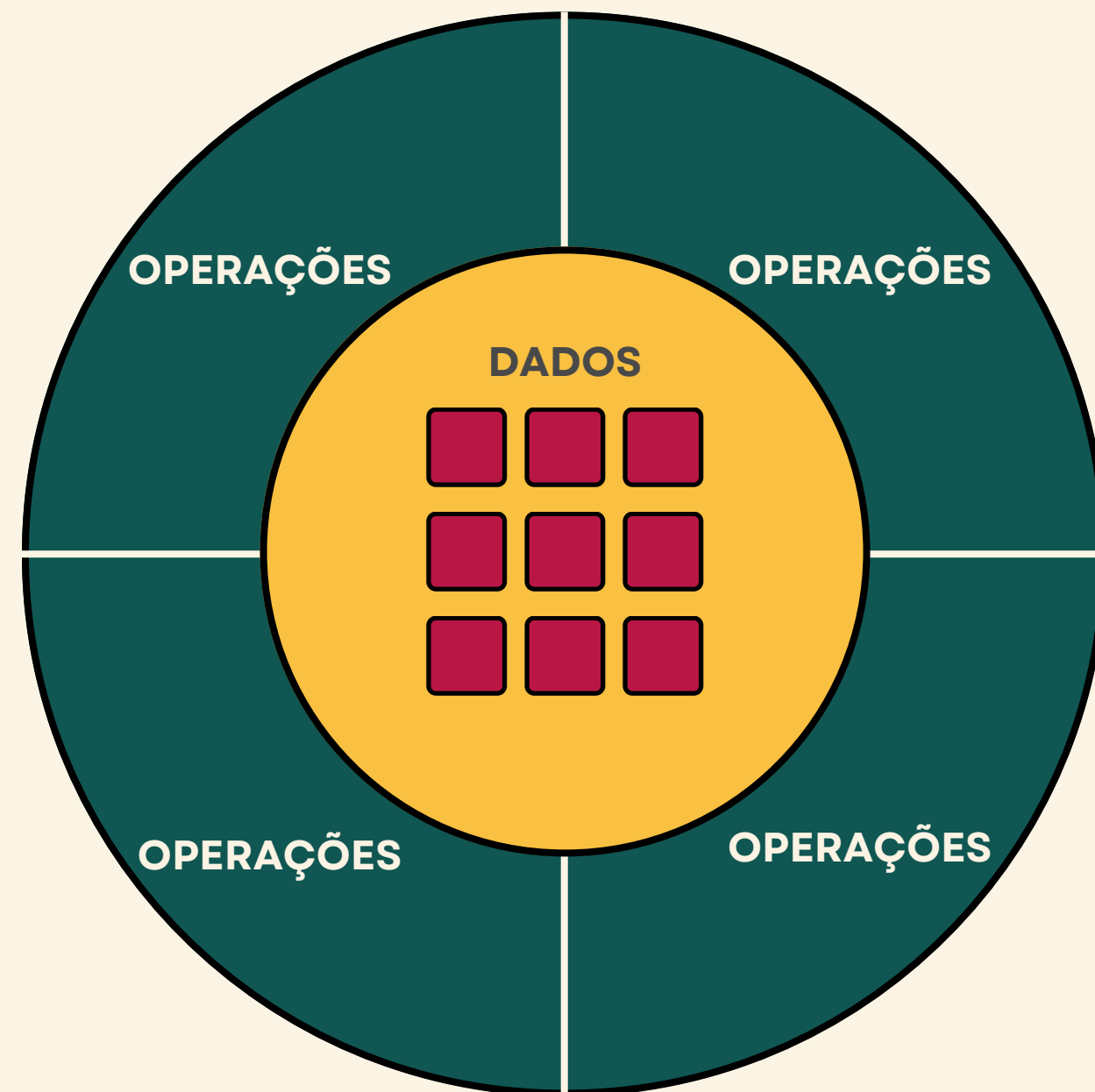
# ABSTRAÇÃO

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

# ENCAPSULAMENTO



# ENCAPSULAMENTO



Biblioteca
nome: string livros: Array<Livro> emprestimos: Array<Emprestimo>
getLivros(): Array<Livro> addLivro(livro: Livro): void empresta(livro: Livro, leitor: Leitor): void

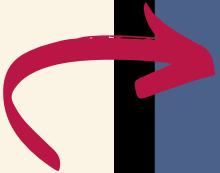
# ENCAPSULAMENTO

```
class Biblioteca{
  nome: string;
  livros: Array<Livro>;
  emprestimos: Array<Emprestimo>;

  constructor(nome: string,
               livros: Array<Livro> = [],
               emprestimos: Array<Emprestimo> = []) {
    this.nome = nome;
    this.livros = livros;
    this.emprestimos = emprestimos;
  }
}
```

# ENCAPSULAMENTO

Definição de método



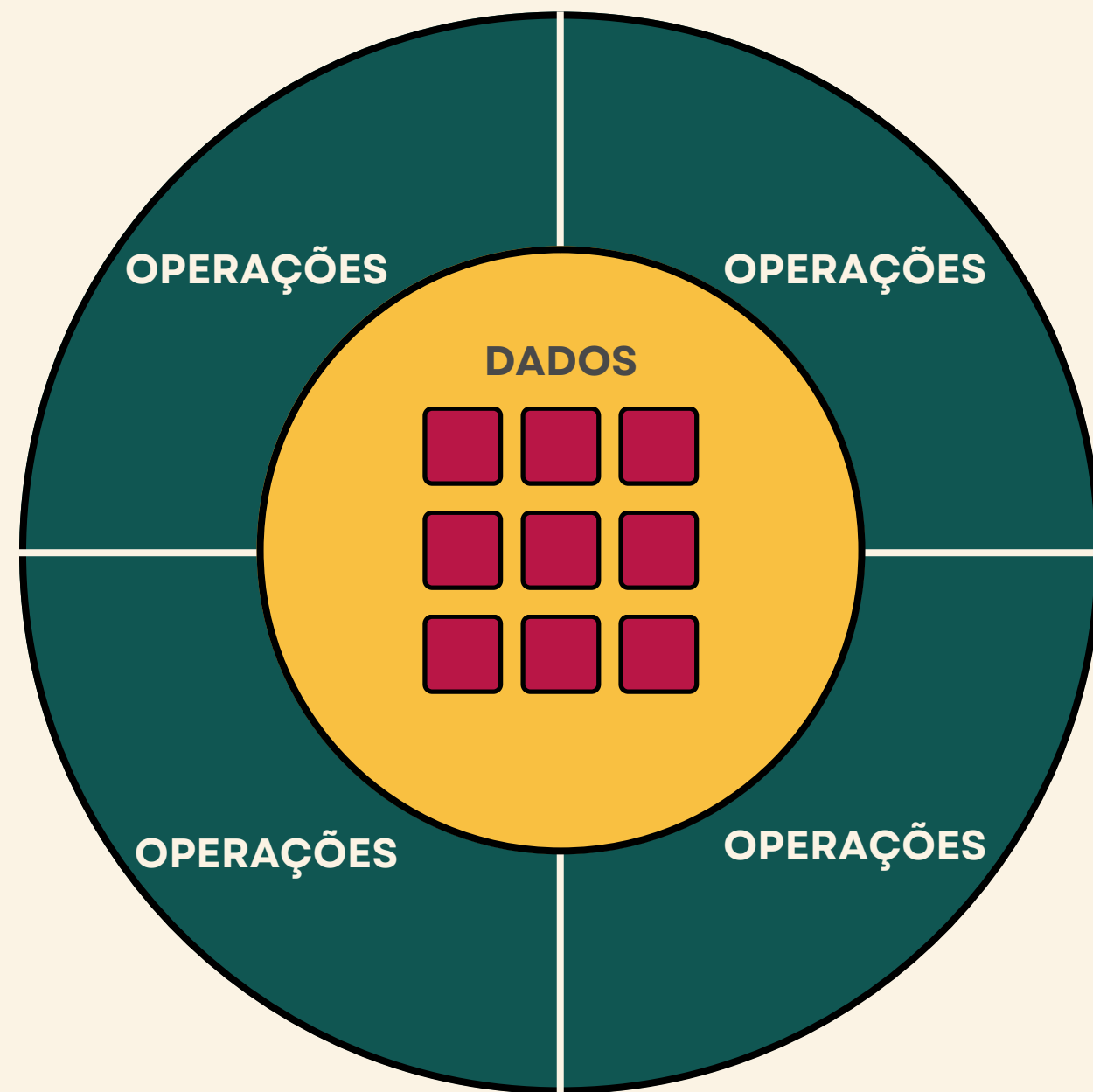
```
class Biblioteca{
  nome: string;
  livros: Array<Livro>;
  emprestimos: Array<Emprestimo>;

  constructor(nome: string,
              livros: Array<Livro> = [],
              emprestimos: Array<Emprestimo> = []) {
    this.nome = nome;
    this.livros = livros;
    this.emprestimos = emprestimos;
  }

  getLivros(): Array<Livro> {
    return this.livros;
  }
  addLivro(livro: Livro): void {
    this.livros.push(livro);
  }
  empresta(data: string, livro: Livro,
           emprestimo: Emprestimo): void {
    this.emprestimos.push(
      new Emprestimo(data, livro, emprestimo));
  }
}
```

# ENCAPSULAMENTO

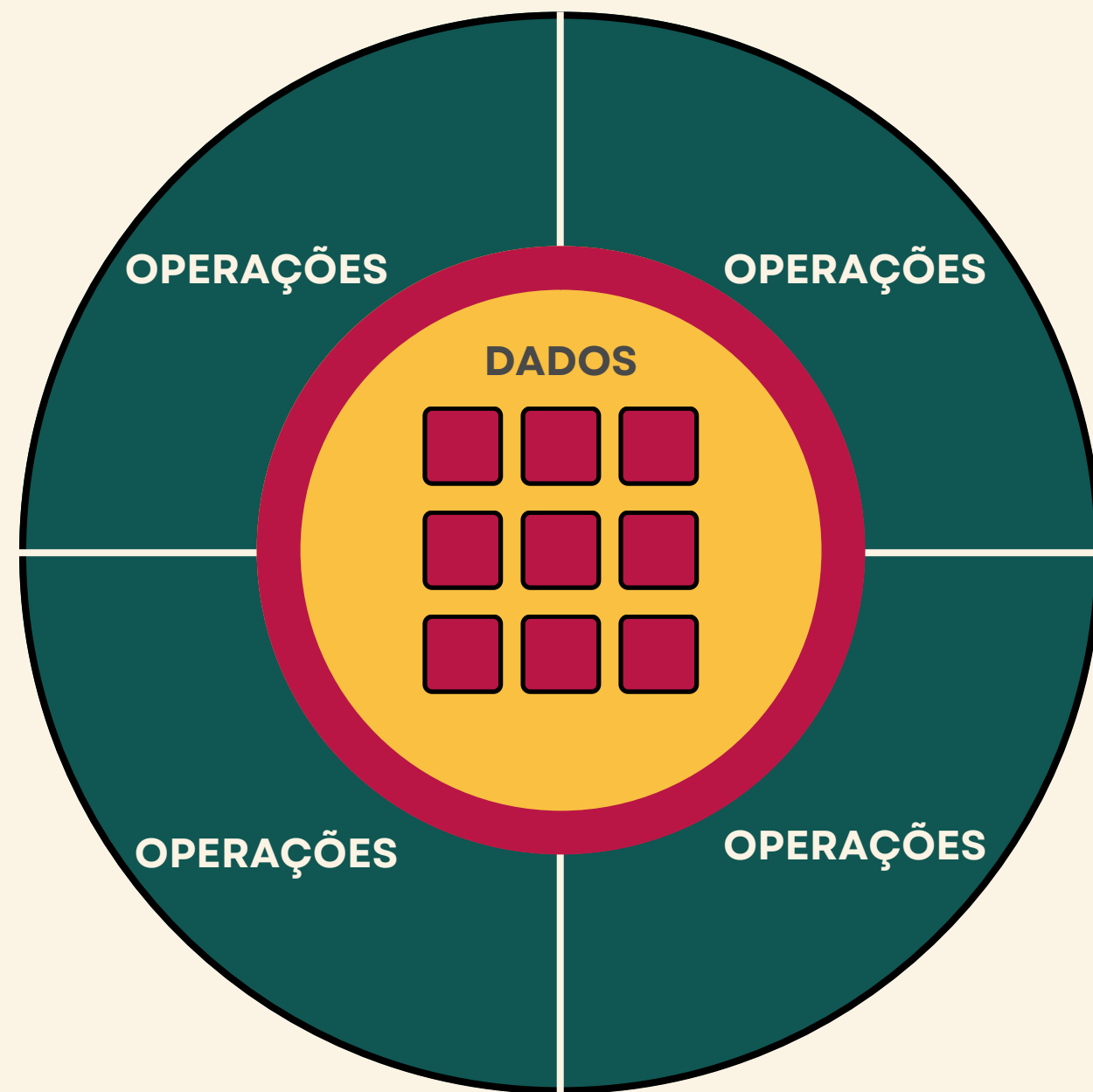
**MODIFICADORES  
DE ACESSO**





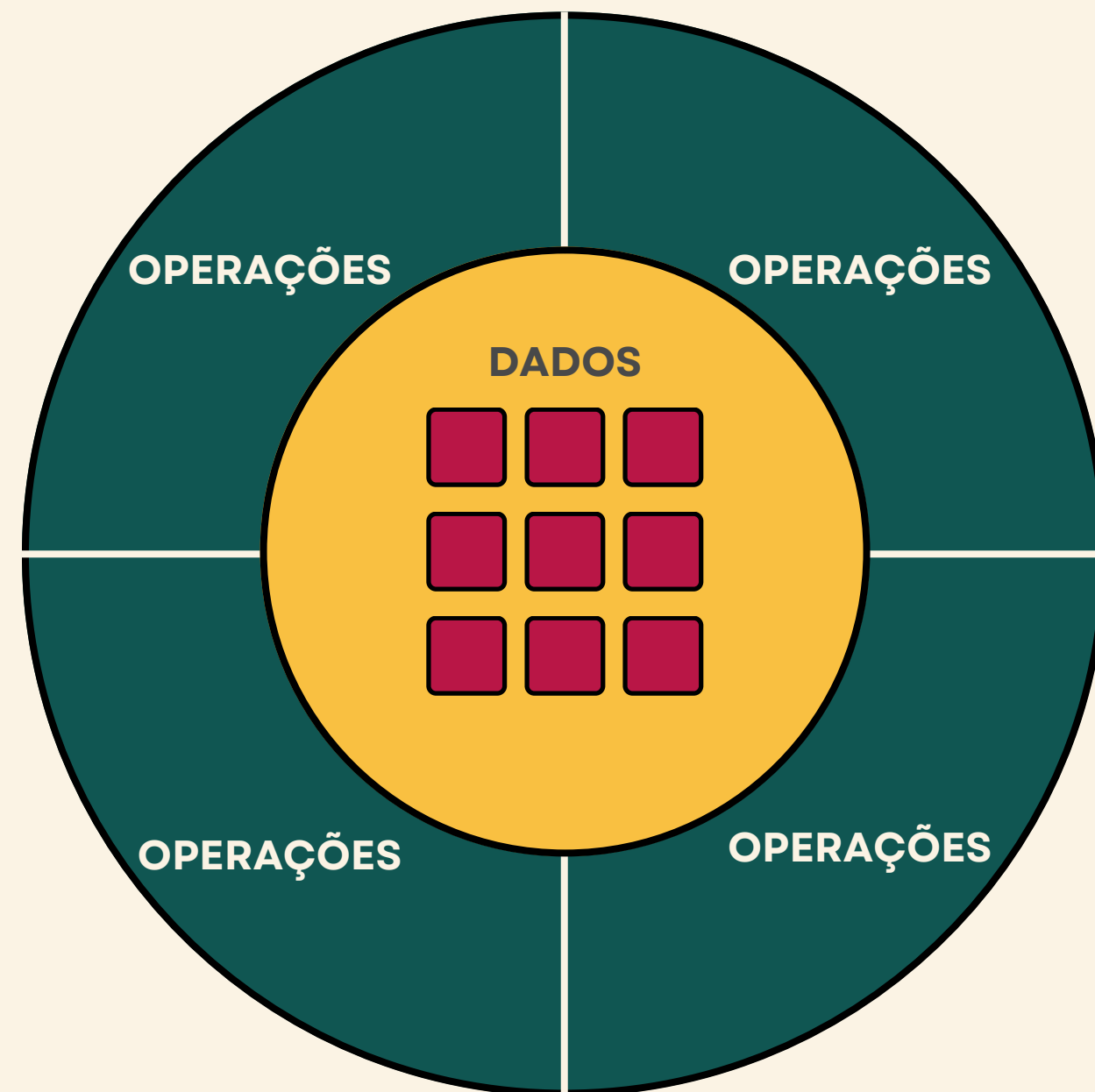
# ENCAPSULAMENTO

**MODIFICADORES  
DE ACESSO**



# ENCAPSULAMENTO

## MODIFICADORES DE ACESSO



### **PUBLIC**

Modificador de acesso PADRÃO. Permite o acesso INDISCRIMINADO.

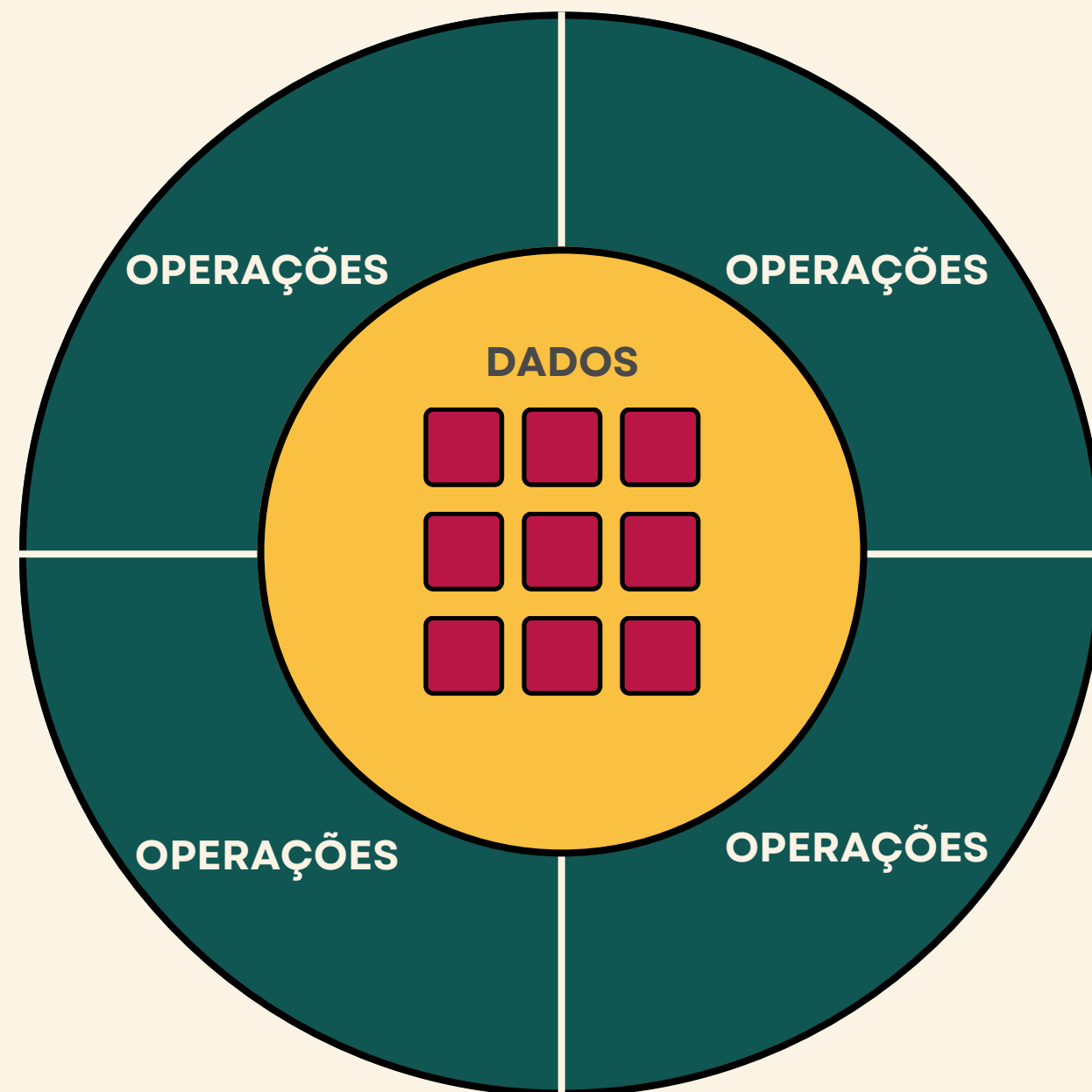


### **PRIVATE**

Define o acesso PRIVADO a VARIÁVEIS e MÉTODOS permitindo o acesso apenas INTERNO pela própria classe.

# ENCAPSULAMENTO

## MODIFICADORES DE ACESSO



### **PUBLIC**

Modificador de acesso PADRÃO. Permite o acesso INDISCRIMINADO.

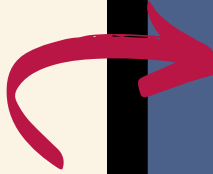


### **PRIVATE**

Define o acesso PRIVADO a VARIÁVEIS e MÉTODOS permitindo o acesso apenas INTERNO pela própria classe.

# ENCAPSULAMENTO

Modificadores de acesso



```
class Biblioteca{
  private nome: string;
  private livros: Array<Livro>;
  private emprestimos: Array<Emprestimo>;

  constructor(nome: string,
               livros: Array<Livro> = [],
               emprestimos: Array<Emprestimo> = []) {
    this.nome = nome;
    this.livros = livros;
    this.emprestimos = emprestimos;
  }

  getLivros(): Array<Livro> {
    return this.livros;
  }
  addLivro(livro: Livro): void {
    this.livros.push(livro);
  }
  empresta(data: string, livro: Livro,
           emprestimo: Emprestimo): void {
    this.emprestimos.push(
      new Emprestimo(data, livro, emprestimo));
  }
}
```

**ABSTRAÇÃO**

**REVISÃO**

---

# ENCAPSULAMENTO

Declaração da classe

Modificadores de acesso

```
class Biblioteca{  
  private nome: string;  
  private livros: Array<Livro>;  
  private emprestimos: Array<Emprestimo>;  
  
  constructor(nome: string,  
              livros: Array<Livro> = [],  
              emprestimos: Array<Emprestimo> = []) {  
    this.nome = nome;  
    this.livros = livros;  
    this.emprestimos = emprestimos;  
  }  
  
  getLivros(): Array<Livro> {  
    return this.livros;  
  }  
  addLivro(livro: Livro): void {  
    this.livros.push(livro);  
  }  
  empresta(data: string, livro: Livro,  
           emprestimo: Emprestimo): void {  
    this.emprestimos.push(  
      new Emprestimo(data, livro, emprestimo));  
  }  
}
```

# ENCAPSULAMENTO

Declaração da classe

Modificadores de acesso

Definição de atributos

```
class Biblioteca{  
  private nome: string;  
  private livros: Array<Livro>;  
  private emprestimos: Array<Emprestimo>;  
  
  constructor(nome: string,  
              livros: Array<Livro> = [],  
              emprestimos: Array<Emprestimo> = []) {  
    this.nome = nome;  
    this.livros = livros;  
    this.emprestimos = emprestimos;  
  }  
  
  getLivros(): Array<Livro> {  
    return this.livros;  
  }  
  addLivro(livro: Livro): void {  
    this.livros.push(livro);  
  }  
  empresta(data: string, livro: Livro,  
           emprestimo: Emprestimo): void {  
    this.emprestimos.push(  
      new Emprestimo(data, livro, emprestimo));  
  }  
}
```

# ENCAPSULAMENTO

Declaração da classe

Modificadores de acesso

Definição de atributos

Definição do construtor

```
class Biblioteca{  
  private nome: string;  
  private livros: Array<Livro>;  
  private emprestimos: Array<Emprestimo>;  
  
  constructor(nome: string,  
    livros: Array<Livro> = [],  
    emprestimos: Array<Emprestimo> = []) {  
    this.nome = nome;  
    this.livros = livros;  
    this.emprestimos = emprestimos;  
  }  
  
  getLivros(): Array<Livro> {  
    return this.livros;  
  }  
  addLivro(livro: Livro): void {  
    this.livros.push(livro);  
  }  
  empresta(data: string, livro: Livro,  
    emprestimo: Emprestimo): void {  
    this.emprestimos.push(  
      new Emprestimo(data, livro, emprestimo));  
    }  
}
```



# ENCAPSULAMENTO

Declaração da classe

```
class Biblioteca{  
  private nome: string;  
  private livros: Array<Livro>;  
  private emprestimos: Array<Emprestimo>;  
  
  constructor(nome: string,  
    livros: Array<Livro> = [],  
    emprestimos: Array<Emprestimo> = []) {  
    this.nome = nome;  
    this.livros = livros;  
    this.emprestimos = emprestimos;  
  }  
  
  getLivros(): Array<Livro> {  
    return this.livros;  
  }  
  addLivro(livro: Livro): void {  
    this.livros.push(livro);  
  }  
  empresta(data: string, livro: Livro,  
    emprestimo: Emprestimo): void {  
    this.emprestimos.push(  
      new Emprestimo(data, livro, emprestimo));  
    }  
}
```

Definição de atributos

Definição do construtor

Classes como tipo de dado

Modificadores de acesso

# ENCAPSULAMENTO

Declaração da classe

```
class Biblioteca{  
  private nome: string;  
  private livros: Array<Livro>;  
  private emprestimos: Array<Emprestimo>;  
  
  constructor(nome: string,  
    livros: Array<Livro> = [],  
    emprestimos: Array<Emprestimo> = []) {  
    this.nome = nome;  
    this.livros = livros;  
    this.emprestimos = emprestimos;  
  }  
  
  getLivros(): Array<Livro> {  
    return this.livros;  
  }  
  addLivro(livro: Livro): void {  
    this.livros.push(livro);  
  }  
  empresta(data: string, livro: Livro,  
    emprestimo: Emprestimo): void {  
    this.emprestimos.push(  
      new Emprestimo(data, livro, emprestimo));  
    }  
}
```

Definição de atributos

Definição do construtor

Classes como tipo de dado

Modificadores de acesso

Referência ao escopo  
dos atributos

# ENCAPSULAMENTO

Declaração da classe

```
class Biblioteca{  
  private nome: string;  
  private livros: Array<Livro>;  
  private emprestimos: Array<Emprestimo>;  
  
  constructor(nome: string,  
    livros: Array<Livro> = [],  
    emprestimos: Array<Emprestimo> = []) {  
    this.nome = nome;  
    this.livros = livros;  
    this.emprestimos = emprestimos;  
  }  
  
  getLivros(): Array<Livro> {  
    return this.livros;  
  }  
  addLivro(livro: Livro): void {  
    this.livros.push(livro);  
  }  
  empresta(data: string, livro: Livro,  
    emprestimo: Emprestimo): void {  
    this.emprestimos.push(  
      new Emprestimo(data, livro, emprestimo));  
    }  
}
```

Definição de atributos

Definição do construtor

Classes como tipo de dado

Modificadores de acesso

Referência ao escopo  
dos atributos

Definição de métodos

**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

# DESAFIO

Dados os elementos criados para definir a biblioteca, implemente métodos para que os objetos interajam entre si:

- **empresta** - adiciona empréstimo à biblioteca
- **devolve** - remove o empréstimo da biblioteca
- **venda** - adiciona uma venda à loja (informar o preço)
- **cadastraLeitor** - adiciona Leitor à biblioteca

Obs.: garanta que os atributos sejam privados;

# BIBLIOTECA



**OBRIGADO!**

# FEEDBACK

**ACESSE**

**WWW.MENTI.COM**

**INSIRA O CÓDIGO**

**2311 1468**



ou use o QR code