AULA 19

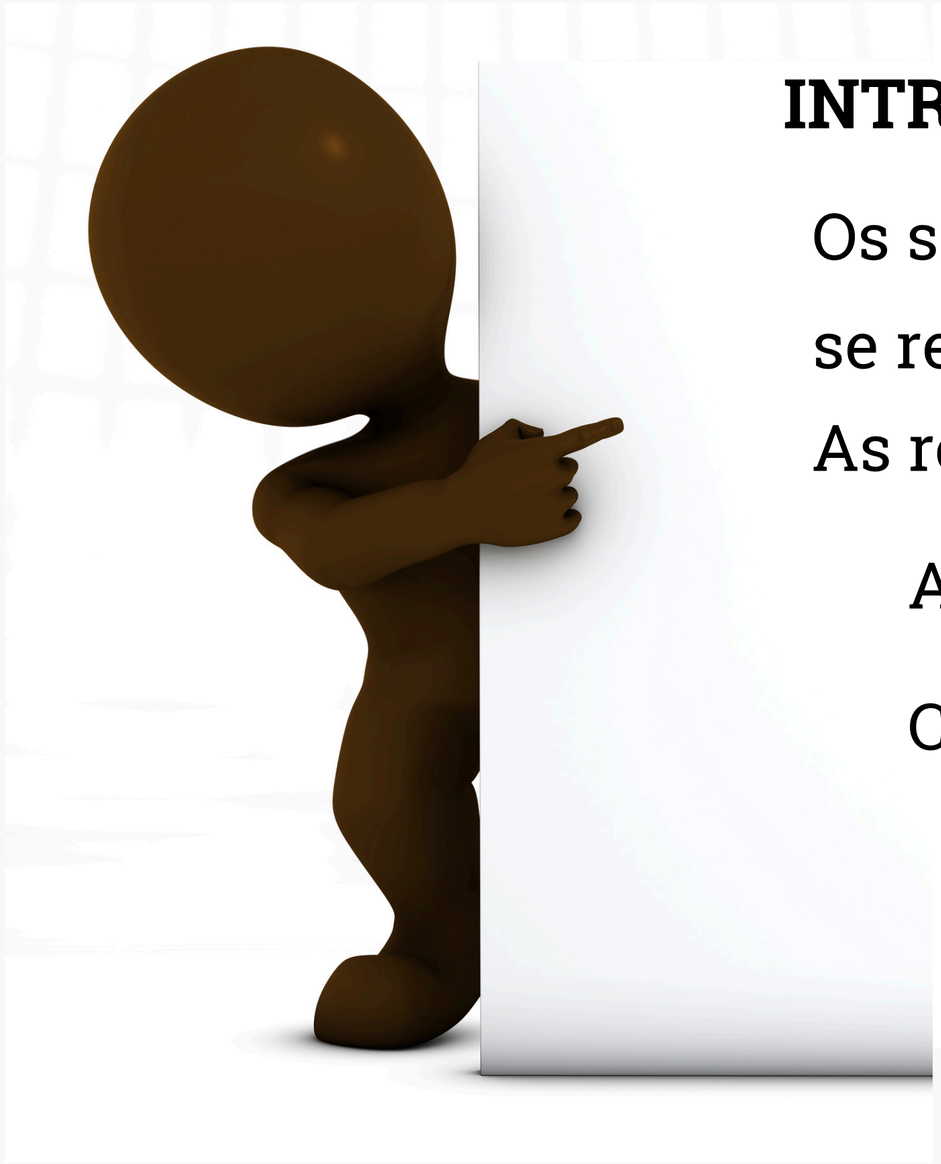
COMPOSIÇÃO E ASSOCIAÇÃO ENTRE OBJETOS

Profª Josiane Lopes

BOLSA FUTURO DIGITAL | PROGRAMA DE FORMAÇÃO ACELERADA
INSTITUTO FEDERAL Sergipe
cepedi
Softex
MINISTÉRIO DA CIÊNCIA, TECNOLOGIA E INOVAÇÃO
GOVERNO FEDERAL BRASIL UNIÃO E RECONSTRUÇÃO

OBJETIVOS DA AULA

- Compreender o que são associação e composição entre objetos
- Aprender a criar relações entre classes para modelar mais sistemas complexos.



INTRODUÇÃO

Os sistemas são formados por objetos que se relacionam entre si

As relações podem ser:

Associação: ***um objeto usa o outro***

Composição ou agregação:

um objeto contém outro como parte essencial de si

ASSOCIAÇÃO:

- **Ocorre quando um objeto conhece outro, mas não depende totalmente dele para existir**
- **É uma relação de colaboração!**

Exemplo: um aluno está associado a um curso, mas o aluno pode existir mesmo sem estar matriculado.

EXEMPLO

```
class Curso {
  constructor(nome) {
    this.nome = nome;
  }
}

class Aluno {
  constructor(nome) {
    this.nome = nome;
    this.curso = null; // associação
  }

  matricular(curso) {
    this.curso = curso;
  }

  exibirInfo() {
    if (this.curso) {
      console.log(`${this.nome} está matriculado em ${this.curso.nome}.`);
    } else {
      console.log(`${this.nome} ainda não está matriculado.`);
    }
  }
}

// Exemplo de uso
const curso1 = new Curso("Programação em JavaScript");
const aluno1 = new Aluno("Mariana");

aluno1.exibirInfo();
aluno1.matricular(curso1);
aluno1.exibirInfo();
```

COMPOSIÇÃO:

- **É um tipo mais forte de relação.**
- **Um objeto é formado por outros objetos (se o objeto principal deixar de existir, os outros também deixam.**

Exemplo: um carro é composto por um motor. Se o carro for destruído, o motor também deixa de existir como parte do sistema desse carro.

EXEMPLO

```
class Motor {
  constructor(potencia) {
    this.potencia = potencia;
  }

  ligar() {
    console.log(`Motor de ${this.potencia}cv ligado.`);
  }
}

class Carro {
  constructor(modelo, potenciaMotor) {
    this.modelo = modelo;
    this.motor = new Motor(potenciaMotor); // composição
  }

  ligar() {
    console.log(`${this.modelo} está sendo ligado...`);
    this.motor.ligar();
  }
}

// Exemplo de uso
const carro1 = new Carro("Honda Civic", 150);
carro1.ligar();
```

Obs.: relação de dependência total

DIFERENÇA:

Característica	Associação	Composição
Relação	Fraca (colaboração)	Forte (parte-todo)
Dependência	Objetos podem existir separadamente	Um depende do outro
Exemplo comum	Aluno ↔ Curso	Carro → Motor
Implementação	A classe recebe outro objeto pronto	A classe cria o objeto internamente

Exemplo prático: Sistema Biblioteca (JS)

```
class Livro {
  constructor(titulo, autor) {
    this.titulo = titulo;
    this.autor = autor;
  }
}

class Biblioteca {
  constructor(nome) {
    this.nome = nome;
    this.livros = []; // composição
  }

  adicionarLivro(titulo, autor) {
    const livro = new Livro(titulo, autor);
    this.livros.push(livro);
  }

  listarLivros() {
    console.log(`Livros disponíveis na ${this.nome}:`);
    this.livros.forEach(livro => {
      console.log(`- ${livro.titulo}, de ${livro.autor}`);
    });
  }
}

// Exemplo de uso
const biblioteca = new Biblioteca("Biblioteca Central");
biblioteca.adicionarLivro("1984", "George Orwell");
biblioteca.adicionarLivro("Dom Casmurro", "Machado de Assis");
biblioteca.listarLivros();
```

EXERCÍCIOS

Crie 2 códigos em JavaScript, com as seguintes características:

ASSOCIAÇÃO

- Crie classe professor com nome
- Classe Disciplina com nome e professor associado
- Método: exibirInfo( ) mostra o nome da disciplina e do professor

COMPOSIÇÃO

- Crie classe computador que contem objetos das classes Processador, memoria e HD
- cada componente deve ter um método info( ) que exibe suas informações
- O método exibirConfig( ) da classe computador deve mostrar todos os detalhes

ATIVIDADE ORIENTADA

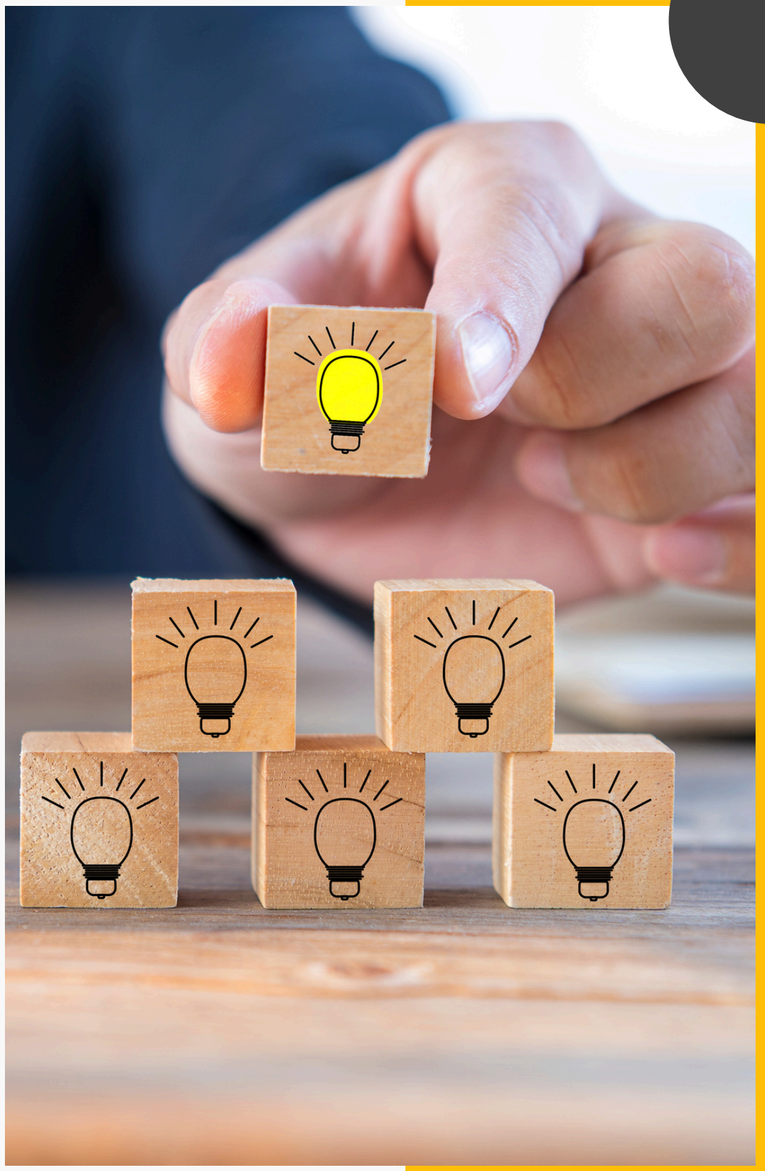
Crie uma classe Empresa que tem vários funcionários. Implemente métodos para:

- Adicionar funcionários (composição).
- Exibir todos os funcionários.
- Mostrar o total de pessoas na empresa.

RESUMINDO...

- **ASSOCIAÇÃO** → um objeto usa o outro, mas ambos são independentes.
- **COMPOSIÇÃO** → um objeto contém outro, criando dependência total.

- Permitem modelar sistemas complexos com múltiplas entidades e relações
- Fundamentais → abstração e modularidade em POO.



OBRIGADA

Profª Josiane Lopes