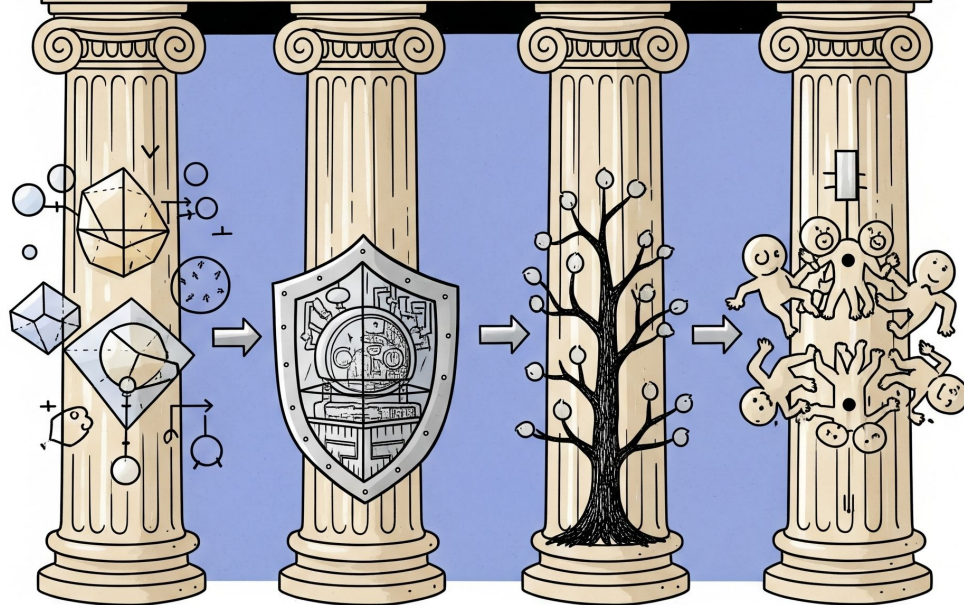


# Polimorfismo

Matheus Barbosa  
matheus.barbosa@dcx.ufpb.br

# OS PILARES DA ORIENTAÇÃO A OBJETOS



**ABSTRAÇÃO**

**ENCAPSULAMENTO**

**HERANÇA**

**POLIMORFISMO**

Polimorfismo significa "muitas formas"

Poli = muitas


Morfo = formas

Permite que um mesmo  
nome represente vários  
comportamentos diferentes

# Assinatura de método

A assinatura de um método em Java é composta por:

- Nome do método
- Lista de parâmetros (quantidade e tipos, na ordem correta)

 Não inclui o tipo de retorno ou modificadores como *public*, *static*, etc.

# Exemplos de métodos com mesma assinatura

```
Class Pessoa {  
    void falar(String mensagem) { ... }  
  
    int falar(String mensagem) { ... } // ✗ Erro: mesma  
assinatura  
}
```

# Exemplos de métodos com assinaturas diferentes

```
Class Pessoa {
```

```
    void falar(String mensagem) { ... }
```

```
    void falar(String mensagem, int vezes) { ... }
```

```
    void falar(int vezes, String mensagem) { ... }
```

```
    void falar() { ... }
```

```
}
```

Assinaturas diferentes  
Mesma classe

## SOBRECARGA

### Classe Animal

```
publico emitirSom(){  
    escrever("Som animal")  
}
```

FimClasse

### Classe Porco estende Animal

```
@Sobrep  
publico emitirSom(){  
    escrever("oinc oinc")  
}
```

FimClasse

### Classe Cachorro estende Animal

```
@Sobrep  
publico emitirSom(){  
    escrever("au au")  
}
```

FimClasse

```
animal = novo Animal();  
porco = novo Porco();  
cachorro = novo Cachorro();
```

```
animal.emitirSom(); // Som animal  
porco.emitirSom();  // oinc oinc  
cachorro.emitirSom(); // au au
```

Mesma assinatura

Classes diferentes

# SOBREPOSIÇÃO



```
// Classe que reproduz o som de qualquer animal
// ~T~ indica tipo genérico.
classe SistemaDeSom<T estende Animal>{
    reproduzir(T animal) {
        // polimorfismo.
        escrever(animal.emitirSom());
    }
}
```

```
sistemaZoologico = novo SistemaDeSom<Animal>();

leao = novo Leao();
pato = novo Pato();
elefante = novo Elefante();

sistemaZoologico.reproduzir(leao); // Roooar!
sistemaZoologico.reproduzir(pato); // Quack quack!
sistemaZoologico.reproduzir(elefante); // Pruuuuuu!
```

Pode ser escrito genericamente para lidar com valores de forma idêntica sem depender do seu tipo

Não é tão comum em Java

# Generics