# **SENAI ROBERTO MANGE**

**DESENVOLVIMENTO DE SISTEMAS**

LUIZA SANTOS GONÇALVES

**PROGRAMAÇÃO ORIENTEDA A OBJETO**

Campinas, SP

2025

# **Programação Orientada a Objetos: Herança e Polimorfismo**

## **Introdução**

Você já percebeu como muitas coisas no mundo real compartilham características em comum, um carro e uma moto, por exemplo, são veículos, ambos se movem, tem rodas, e precisam de combustível ou eletricidade. Mas cada um tem seu jeitinho particular de funcionar.

É exatamente essa ideia que a **Programação Orientada a Objetos (POO)** tenta capturar no mundo da programação. A POO nos ajuda a organizar nosso código como se estivéssemos modelando objetos do mundo real, tornando o desenvolvimento mais claro, reutilizável e fácil de manter.

Entre os quatro pilares da POO, dois se destacam por sua importância e versatilidade: herança e polimorfismo.

## **O que é Herança?**

Imagine que você está programando um jogo com diferentes tipos de animais. Todos eles têm algo em comum — um nome, a capacidade de emitir um som, talvez se movimentar. Em vez de escrever esse comportamento repetidamente para cada animal, você cria uma classe mãe chamada Animal, e as classes Cachorro, Gato, Pássaro herdam dela.

A herança funciona como um legado. A classe filha ou subclasse herda características e comportamentos da classe mãe ou superclasse. Isso evita repetição e deixa o código mais organizado.

**O que é Polimorfismo?**

O nome pode parecer complicado, mas o conceito é simples: polimorfismo vem do grego e significa "muitas formas". Na programação, é quando objetos diferentes respondem de formas distintas ao mesmo comando.

## **🔍 Herança vs. Polimorfismo: Qual a diferença?**

|  |  |  |
| --- | --- | --- |
| **Característica** | **Herança** | **Polimorfismo** |
| O que é? | Reutilização de código entre classes | Um mesmo método com comportamentos diferentes |
| Quando usar? | Quando classes compartilham estrutura/comportamento | Quando queremos generalizar ações sobre objetos diferentes |
| Exige relação direta? | Sim (superclasse e subclasse) | Não necessariamente |
| Benefício principal | Menos repetição, mais organização | Mais flexibilidade e reutilização |

## **Por que usar esses conceitos na arquitetura de sistemas?**

### **Vantagens da Herança**

* **Reuso de código**: Você escreve apenas uma vez e pode usar várias vezes.
* **Organização lógica**: Crie estruturas hierárquicas claras.
* **Facilidade de manutenção**: Se alterar a superclasse acaba afetando todas as subclasses.

### **Vantagens do Polimorfismo**

* **Flexibilidade**: Trate diferentes objetos de forma unificada.
* **Extensibilidade**: Adicione novos comportamentos sem mexer no código existente.
* **Simplicidade**: Escreva código mais limpo e desacoplado.

## **Referências**

1. Lutz, Mark. *Learning Python*. O’Reilly Media.
2. Deitel, Paul & Deitel, Harvey. *Python – Como Programar*. Pearson.
3. Gamma, Erich et al. *Design Patterns: Elements of Reusable Object-Oriented Software*.
4. [Documentação oficial do Python](https://docs.python.org/3/)
5. TutorialsPoint – Python OOP
6. Real Python – OOP