# **SENAI ROBERTO MANGE**

**DESENVOLVIMENTO DE SISTEMAS**

LUIZA SANTOS GONÇALVES

**HERANÇA E POLIMORFISMO NA PROGRAMAÇÃO ORIENTADA A OBJETOS**

Campinas,SP

2025

# **Herança e Polimorfismo na Programação Orientada a Objetos (POO)**

**Introdução**

Neste material, abordaremos de forma clara e objetiva dois dos principais pilares da Programação Orientada a Objetos: **herança** e **polimorfismo**. São conceitos fundamentais para o desenvolvimento de sistemas mais organizados, reutilizáveis e escaláveis. A seguir, explicamos suas definições, aplicações práticas com exemplos em Python, as diferenças e interações entre ambos, além das principais vantagens no uso desses recursos.

**Conceitos Teóricos**

1. **Herança**

A herança é um mecanismo da POO que permite que uma classe (chamada de *subclasse* ou *classe filha*) reutilize atributos e métodos de outra classe (conhecida como *superclasse* ou *classe pai*). Isso possibilita que funcionalidades comuns sejam centralizadas em uma classe base, e que as subclasses possam especializar ou estender esse comportamento quando necessário.

**Exemplos:**   
Imagine uma classe Veículo, que possui características como marca e modelo. A partir dela, podemos criar outras classes como Carro e Moto, que herdam essas características e ainda acrescentam comportamentos específicos.

Esse tipo de estrutura reduz a repetição de código e facilita a manutenção do sistema.

**Polimorfismo**

O polimorfismo é a capacidade de diferentes classes responderem de maneiras distintas a uma mesma operação ou método. Em outras palavras, objetos de diferentes tipos podem ser tratados de forma uniforme, desde que implementem uma interface ou comportamento esperado.

Existem dois tipos principais de polimorfismo:

* **Polimorfismo por sobrescrita (override):** quando uma subclasse redefine um método herdado da superclasse para adaptá-lo ao seu contexto.
* **Polimorfismo por sobrecarga (overload):** quando métodos com o mesmo nome aceitam diferentes números ou tipos de parâmetros (não suportado diretamente em Python, mas pode ser simulado).

**Exemplos em Python**

**Exemplo de Herança**

Texto

O conteúdo gerado por IA pode estar incorreto.

Neste exemplo, Carro herda os atributos e métodos da classe Veículo, e ainda adiciona o atributo portas.

**Exemplo de Polimorfismo**

Texto

O conteúdo gerado por IA pode estar incorreto.

Perceba que a função **fazer\_barulho** pode receber qualquer objeto que implemente o método **emitir\_som**. Isso é polimorfismo: o mesmo código funciona com diferentes tipos de objeto.

**Diferenças e Relações entre Herança e Polimorfismo**

| **Aspecto** | **Herança** | **Polimorfismo** |
| --- | --- | --- |
| Definição | Reutilização de código entre classes por meio de uma hierarquia | Capacidade de usar a mesma interface com diferentes comportamentos |
| Relação entre classes | "É um tipo de..." (ex: um Carro é um tipo de Veículo) | "Pode ser tratado como..." (ex: qualquer Animal pode emitir som) |
| Objetivo principal | Reduzir duplicidade e facilitar manutenção | Aumentar a flexibilidade e generalização de código |
| Requer herança direta? | Sim | Não necessariamente (pode usar interfaces ou duck typing) |

**Vantagens na Arquitetura de Sistemas**

**Vantagens da Herança**

* Reutilização de código já testado e funcional.
* Redução de redundância.
* Organização hierárquica das entidades no sistema.
* Facilidade na manutenção e evolução de sistemas complexos.

**Vantagens do Polimorfismo**

* Maior flexibilidade na implementação de novos comportamentos.
* Código mais genérico, que se adapta a diferentes objetos.
* Redução da dependência direta entre partes do código.
* Facilita a extensão do sistema sem modificar estruturas já existentes.

**Referências**

* Rocketseat. *Herança e Polimorfismo em Python: Um Guia Detalhado.* Disponível em: [https://blog.rocketseat.com.br/heranca-e-polimorfismo-em-python-um-guia-detalhado](https://blog.rocketseat.com.br/heranca-e-polimorfismo-em-python-um-guia-detalhado/)
* Python Progressivo. *Polimorfismo em Python.* Disponível em: <https://www.pythonprogressivo.net/2018/11/Polimorfismo-O-que-Como-Usar-Como-fazer.html>
* Wikipedia. *Programação orientada a objetos.* <https://pt.wikipedia.org/wiki/Programa%C3%A7%C3%A3o_orientada_a_objetos>
* Lutz, M. (2013). *Learning Python*. O'Reilly Media.
* Deitel, H. M. & Deitel, P. J. (2016). *Python: Como Programar*. Pearson.