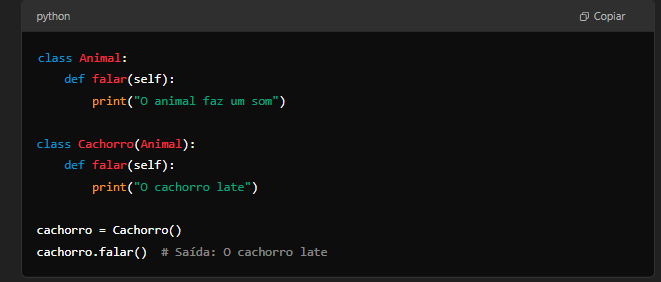
revisao de POO

### **1. Herança**

Herança é um mecanismo que permite que uma classe herde atributos e métodos de outra classe. Isso permite a reutilização de código e a criação de hierarquias. A classe que herda é chamada de **classe derivada** (ou subclasse), e a classe da qual ela herda é chamada de **classe base** (ou superclasse). A herança promove a criação de classes mais especializadas a partir de classes mais gerais.( mecanismo da OO que possibilita a um sudtipo aproveitar propriedades devinidas e um super tipo)

Propriedades ( sao atributos relacionamentos e metodos)

**Exemplo:**



O **polimorfismo** é um conceito fundamental na programação orientada a objetos (POO) que permite que objetos de diferentes classes sejam tratados como objetos de uma classe base comum, mas se comportem de maneira específica de acordo com sua própria implementação. Tecnicamente, o polimorfismo se manifesta quando uma mesma operação (método ou função) pode ser executada de formas diferentes, dependendo do objeto que a invoca.

### **Definição Técnica**

O polimorfismo é a capacidade de um objeto de assumir múltiplas formas, permitindo que métodos com a mesma assinatura (nome e parâmetros) tenham comportamentos diferentes em classes distintas. Isso é alcançado por meio de dois mecanismos principais:

1. **Sobrescrita de Métodos (Override):**
   * Uma subclasse redefine um método já existente em sua superclasse, fornecendo uma implementação específica.
   * O método na subclasse tem a mesma assinatura (nome e parâmetros) que o método na superclasse.
2. **Sobrecarga de Métodos (Overload):**
   * Uma classe define múltiplos métodos com o mesmo nome, mas com parâmetros diferentes.
   * O método específico a ser executado é determinado em tempo de compilação com base nos argumentos passados.

### **Tipos de Polimorfismo**

1. **Polimorfismo de Subtipo (Inclusão):**
   * Ocorre quando uma subclasse é usada no lugar de sua superclasse.
   * Exemplo: Um objeto da classe Cachorro pode ser tratado como um objeto da classe Animal (se Cachorro herdar de Animal).
2. **Polimorfismo Paramétrico (Generics):**
   * Envolve o uso de tipos genéricos, permitindo que uma classe ou método opere sobre qualquer tipo de dado.
   * Exemplo: Listas genéricas em Java (List<String>, List<Integer>).
3. **Polimorfismo de Coerção (Conversão):**
   * Ocorre quando um tipo de dado é implicitamente convertido em outro tipo.
   * Exemplo: Conversão de int para double em operações aritméticas.
4. **Polimorfismo de Sobrecarga (Ad Hoc):**
   * Envolve a definição de múltiplos métodos com o mesmo nome, mas com parâmetros diferentes.
   * Exemplo: Métodos somar(int, int) e somar(double, double).

### **Exemplo Técnico em Java**

java

Copy

// Superclasse

class Animal {

void fazerSom() {

System.out.println("Som genérico de animal");

}

}

// Subclasse

class Cachorro extends Animal {

@Override

void fazerSom() {

System.out.println("Latido");

}

}

// Subclasse

class Gato extends Animal {

@Override

void fazerSom() {

System.out.println("Miado");

}

}

public class Main {

public static void main(String[] args) {

Animal meuAnimal1 = new Cachorro(); // Polimorfismo de subtipo

Animal meuAnimal2 = new Gato(); // Polimorfismo de subtipo

meuAnimal1.fazerSom(); // Saída: Latido

meuAnimal2.fazerSom(); // Saída: Miado

}

}

### **Benefícios do Polimorfismo**

1. **Reutilização de Código:**
   * Permite que métodos genéricos sejam definidos na superclasse e reutilizados nas subclasses.
2. **Extensibilidade:**
   * Facilita a adição de novas classes sem modificar o código existente.
3. **Abstração:**
   * Permite tratar objetos de diferentes classes de maneira uniforme, simplificando o design do sistema.
4. **Manutenção:**
   * Reduz a complexidade do código, tornando-o mais modular e fácil de manter.

### 

### 

### 

### 

### 

### 

### 

### 

### 

### **2. Encapsulamento**

Encapsulamento é o conceito de ocultar os detalhes internos de uma classe e fornecer uma interface pública para interação. Isso é feito usando modificadores de acesso, como public, private ou protected (dependendo da linguagem). O objetivo é proteger os dados de serem modificados diretamente e garantir que eles só possam ser alterados de maneira controlada, por meio de métodos específicos (getters e setters).

**Exemplo:**python

### 

### 

### 

### 

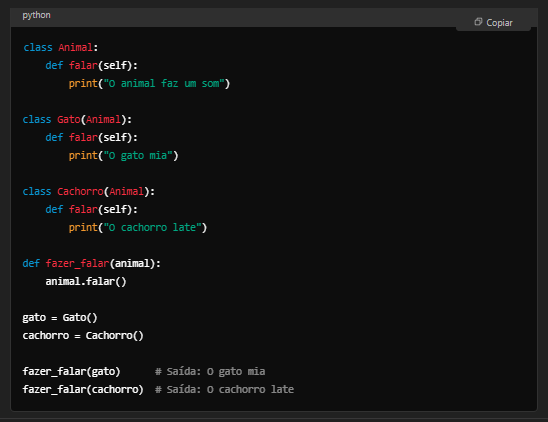
### 

### 

### 

### **3. Polimorfismo**

Polimorfismo é o conceito de que objetos de diferentes classes podem ser tratados como objetos de uma classe comum, mas cada um pode ter comportamentos diferentes. Ele permite que métodos com o mesmo nome se comportem de maneira distinta, dependendo da classe do objeto. O polimorfismo pode ser alcançado por **sobrecarga** (mesmo nome, diferentes parâmetros) ou **sobrescrita** (método na classe derivada substitui o método da classe base).

**Exemplo:**python  
Copiar  


Esses conceitos formam a base da POO e são essenciais para a criação de software mais modular, reutilizável e de fácil manutenção.