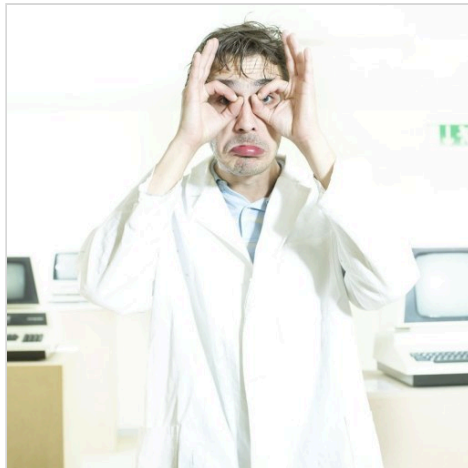


L U M B R E



Herencia y polimorfismo

Grado: 2

Materia: Programación Orientada a Objetos · Programación

11/10/2026

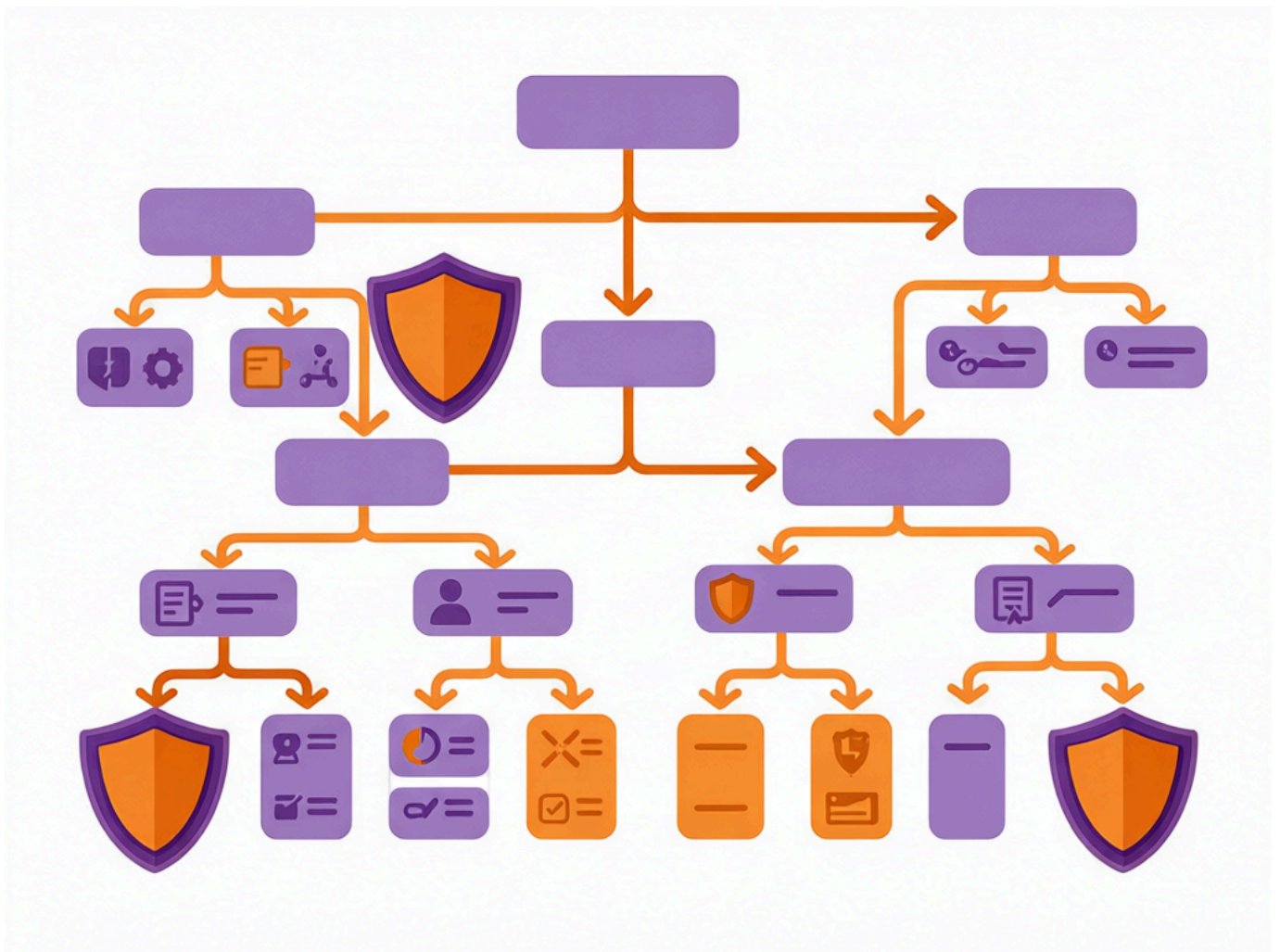
Antes de empezar

La herencia evita repetir: una clase hija hereda de una clase padre. El polimorfismo permite que el mismo mensaje («dibujar()», «calcular()») signifique cosas distintas según el objeto, y eso es lo que hace extensible el código.

QUÉ APRENDERÁS

- Modelar jerarquías con herencia, redefinir comportamiento y aprovechar el polimorfismo para escribir código que trate objetos distintos de forma uniforme.

Herencia y polimorfismo



Principios de la programación orientada a objetos: clases, herencia y polimorfismo.

Mapa conceptual

- Programación OO

- Clases
 - Objetos
 - Atributos
 - Metodos self
- Encapsulamiento
 - Public
 - Private
 - Properties
- Herencia
 - Polimorfismo
 - Abstractas
 - Interfaces
- Composicion
 - Sobre herencia
 - Delegacion
 - Acoplamiento
- Proyecto
 - Paquetes
 - Modulos
 - Convenciones

El mapa mental interactivo es interactivo en la versión online.



Reutilizar y generalizar

Si Perro y Gato ambos comen y duermen, no repitas ese código: define un Animal y haz que hereden. Y cuando algo llame a «hablar()», no te importa si es perro o gato: responderá cada uno a su manera. Eso es polimorfismo.

Herencia

Una clase puede heredar de otra: la subclase (hijo) recibe los atributos y métodos de la superclase (padre) y puede añadir los propios o reescribir los heredados. Se expresa indicando el padre entre paréntesis al definir la clase.

Override y super

- **Override** (redefinir): la subclase ofrece su versión de un método del padre, que gana al invocarse.
- **super()**: permite llamar a la versión del padre desde el override, reutilizando en vez de duplicar.

Polimorfismo

Polimorfismo es que distintos objetos respondan al MISMO mensaje con comportamiento propio. Una lista [perro, gato, pajar] puede recibir un bucle que llame a hablar() en cada elemento sin preguntar de qué tipo es. El código depende de la interfaz, no del tipo concreto.

Sustituibilidad

El principio de sustitución de Liskov: donde se espera un Animal, debería poder colocarse cualquier subtipo sin romper el programa. Si un «Pinguino» no puede volar aunque vuel() esté en el padre, la jerarquía está mal diseñada.

Herencia organiza; polimorfismo desacopla.

Concepto	Qué es	Para qué sirve
Herencia	subclase recibe lo de la superclase	no repetir
Override	redefinir un método heredado	especializar
super()	llamar al método del padre	reutilizar y extender
Polimorfismo	mismo mensaje, respuesta según tipo	código genérico y extensible

1. ¿Qué permite el polimorfismo en un bucle que llama a dibujar() sobre formas distintas?

- ☐ Que el bucle pregunte el tipo con if para cada caso
- ☐ Que cada objeto responda a dibujar() a su manera, sin que el bucle conozca sus tipos
- ☐ Que todas las formas dibujen idéntico
- ☐ Que se herede más rápido

Herencia múltiple

Una clase puede heredar de varias. Da flexibilidad pero introduce el problema del diamante: ¿qué versión de un método se usa? Python lo resuelve con un orden lineal (MRO). Como regla, evita herencia múltiple profunda y prefiere composición.

Cuándo NO heredar

Solo tiene sentido heredar cuando hay una relación es-un real y compartida: un Coche ES-un Vehículo. Heredar solo para reutilizar código, cuando no hay esa relación, produce jerarquías frágiles; mejor composición (tema de la siguiente lección).

2. Gracias al polimorfismo, una función que recibe un Animal puede funcionar con subtipos que se creen en el futuro sin modificarse.

- ☐ Verdadero
- ☐ Falso

Duck typing sin herencia

En Python a menudo no hace falta heredar una base común: basta con que el objeto implemente el método esperado (duck typing, «si camina como pato...»). El polimorfismo no exige una jerarquía formal, solo una interfaz compartida.

Errores frecuentes

- Crear jerarquías enormes heredando solo para no copiar código.
- Violar Liskov: una subclase que rompe lo que el padre promete.
- Abusar de isinstance en vez de delegar el comportamiento en cada tipo.

3. Clasifica cada situación:

Elementos:

- Coche y Moto heredan de Vehículo
- Heredar de Lista solo para reutilizar un método
- Perro y Gato heredan de Animal y redefinen hablar
- Jerarquía de 8 niveles para compartir 2 líneas

Buena herencia	. . .
Mala herencia	. . .

Real Python <https://realpython.com/inheritance-composition-python/> Herencia frente a composición, con ejemplos.

VIDEO

Video complementario

Recurso audiovisual para reforzar los conceptos.



Escanea para verlo en Lumbre

<https://www.youtube.com/watch?v=Zbe5K6uEJYw>

4. Modela una jerarquía de medios de pago (tarjeta, transferencia, efectivo) con un método procesar(). Explica dónde aplica polimorfismo y por qué NO heredarías efectivo de tarjeta.

Python: Classes <https://docs.python.org/3/tutorial/classes.html> Tutorial oficial de clases en Python 3.

Referencia curricular

Grado:	2
Materia:	Programación Orientada a Objetos · Programación

Versión interactiva completa (con actividades, videos y autoevaluación):



<https://univ.lumbre.lat/object/cdia-poo-004-herencia-y-polimorfismo>

Material educativo generado desde Lumbre · lumbre.lat