UNIVERSIDAD CATÓLICA DE TEMUCO | INGENIERÍA CIVIL EN INFORMÁTICA FACULTAD DE INGENIERÍA

Informe de POO y control de versiones con LATEX

Rallen Castro

17 de Octubre 2025

Institución: Universidad Católica de Temuco

Ramo: Programación II Sección 2

Profesor a Cargo: Guido Mellado

---

1. Introducción

Para este informe se explicará sobre Herencia, Clases Abstractas, Polimorfismo e Interfaces, pilares fundamentales en el núcleo de la programación orientada a objetos (POO), esto complementado con ejemplos prácticos y esquemas ilustrativos que faciliten su comprensión. Además, se incluirá una sección especializada dedicada al Method Resolution Order (MRO), un concepto importante en la herencia múltiple que determina el orden en que Python busca métodos en la jerarquía de clases.

2. Conceptos POO

2.1. herencia

Es un mecanismo para crear nuevas clases (subclase o hijas), se basan en clases existentes porque dependen de sus elementos (padres o superclase). La herencia facilita la reutilización de codigo, no necesitando copiar y pegar ciertos mecanismos en una clase que, por ejemplo, pueden ya estar en otra clase, asi mismo estableciendo relaciones entre clases, donde la subclase hereda atributos y/o métodos de una superclase.

Ventajas

- Reutilización de código: Evita tener que duplicar al heredar funcionalidades extistentes
- Extensibilidad: Permite añadir nuevas características sin modificar la clase base.
- Organización jerárquica: Establece relaciones naturales entre clases.
- Mantenibilidad: Los cambios en la clase base se propagan automáticamente a las hijas.

2.2. Clases Abstractas

Una clase que existe solo para servir como una plantilla para las demás clases, en esta clase no se pueden crear objetos directamente, su función principal es definir una interfaz común y obligar a sus subclases a implementar métodos específicos.

Ventajas

- Fuerza el Contrato: Garantiza que las subclases implementen métodos importantes, asegurando una buena estructura de codigo.
- Modelo Jerárquico: Permite organizar clases relacionadas de manera lógica, proporcionando una base común para ellas.
- Prevención de Instanciación: Evita que el crear objetos de una clase que no tiene sentido por sí misma.

2.3. Polimorfismo

permite que un mismo nombre de método pueda comportarse de manera diferente según el objeto que lo invoca. Esto significa que una sola interfaz puede usarse para diferentes implementaciones. Esta característica es clave para escribir código más general y flexible, ya que permite tratar objetos de distintas clases de forma uniforme a través de una misma interfaz. El polimorfismo aparece cuando varias clases implementan el mismo método con comportamientos específicos, pero pueden ser accedidos mediante una referencia más general. Esto ayuda a ampliar el código fácilmente y a reducir la dependencia entre partes del programa.

Ventajas

- Flexibilidad: Permite que un programa interactúe con objetos de diferentes tipos a través de una interfaz común, sin necesidad de saber el tipo exacto de cada objeto.
- Desacoplamiento: Reduce las dependencias directas entre las clases, lo que hace el código más modular.
- Claridad: Simplifica el código, ya que se puede usar el mismo nombre de método (ej: `dibujar()`) para diferentes objetos (Círculo, Cuadrado, Triángulo).

Ejemplo

Para ilustrar la sobrescritura, donde cada subclase define su propia implementación del método `Persona`:

```
class Persona:
    def saludar(self):
        return "Hola, soy una persona"

class Niño(Persona):
    def saludar(self):
        return "Hola, soy un niño"
```

2.4. Interfaces

una forma de definir un conjunto de métodos sin tener que escribir cómo funcionan. Las interfaces se centran solo en el comportamien-

to que una clase debe tener (el çontrato"). Esto permite separar la definición del comportamiento de la forma en que se implementa. Así, diferentes clases pueden cumplir con el mismo contrato sin depender unas de otras. Las interfaces hacen que el diseño sea más flexible y fácil de mantener. Las clases que las usan deben ofrecer todas las funciones que la interfaz pide, lo que ayuda a mantener la coherencia en el sistema. Además, permite cambiar componentes sin afectar el código que los usa.

Ventajas

- Multitipos: Permite que una clase implemente múltiples interfaces, modelando comportamientos diversos sin la complejidad de la herencia múltiple de implementación.
- Estándar: Define un estándar claro y obligatorio de comunicación entre componentes del sistema.
- Simplicidad: Fomenta el diseño basado en el principio "programar a una interfaz y no a una implementación".

3. Method Resolution Order (MRO)

En español; Orden de Resolución de Métodos, es un concepto vital y obligatorio en lenguajes de programación que soportan Herencia Múltiple, obviamente en este caso es python

El MRO define la secuencia precisa en la que el intérprete busca un método o un atributo dentro de una jerarquía de clases cuando una subclase hereda de múltiples padres. Si no existiera este orden determinista, el programa podría fallar al encontrar múltiples implementaciones del mismo método.

Criterios de Linealización
Para garantizar que la búsqueda sea consistente y segura, el MRO sigue el Algoritmo de Linealización C3, el cual respeta dos principios fundamentales:

- Clase Precede a la Base: Cualquier clase siempre debe aparecer antes que sus clases base.
- Orden de Definición: El orden en que las clases base son especificadas en la definición de la clase debe ser respetado.

3.1. Fórmula del Algoritmo C3

La MRO de una clase C, denotada como $L[C]$, se define mediante la siguiente fórmula recursiva del Algoritmo C3 para una herencia de clases base $P_1, P_2, \ldots, P_N$:

$$L[C] = C + \text{merge}(L[P_1], L[P_2], \ldots, L[P_N], (P_1, P_2, \ldots, P_N)) \quad (1)$$

La operación merge() combina las linealizaciones de los padres ($L[P_i]$) y las clases padre directas (P_i) bajo las reglas de precedencia, resultando en una lista única y determinista que define el orden de búsqueda del método.

Referencias

[1] Cisco Networking Academy: INFO1123-2 Fundamentos de Python 2
https://www.netacad.com/