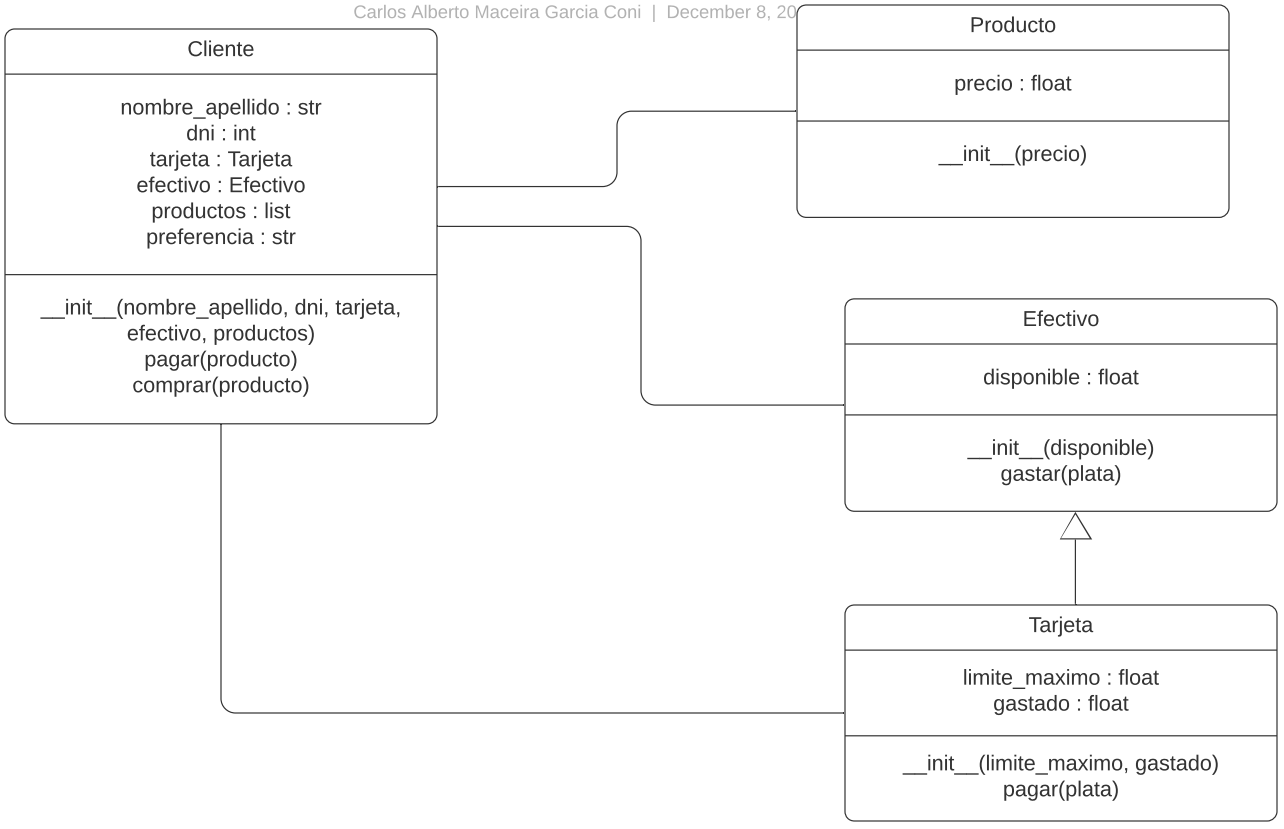
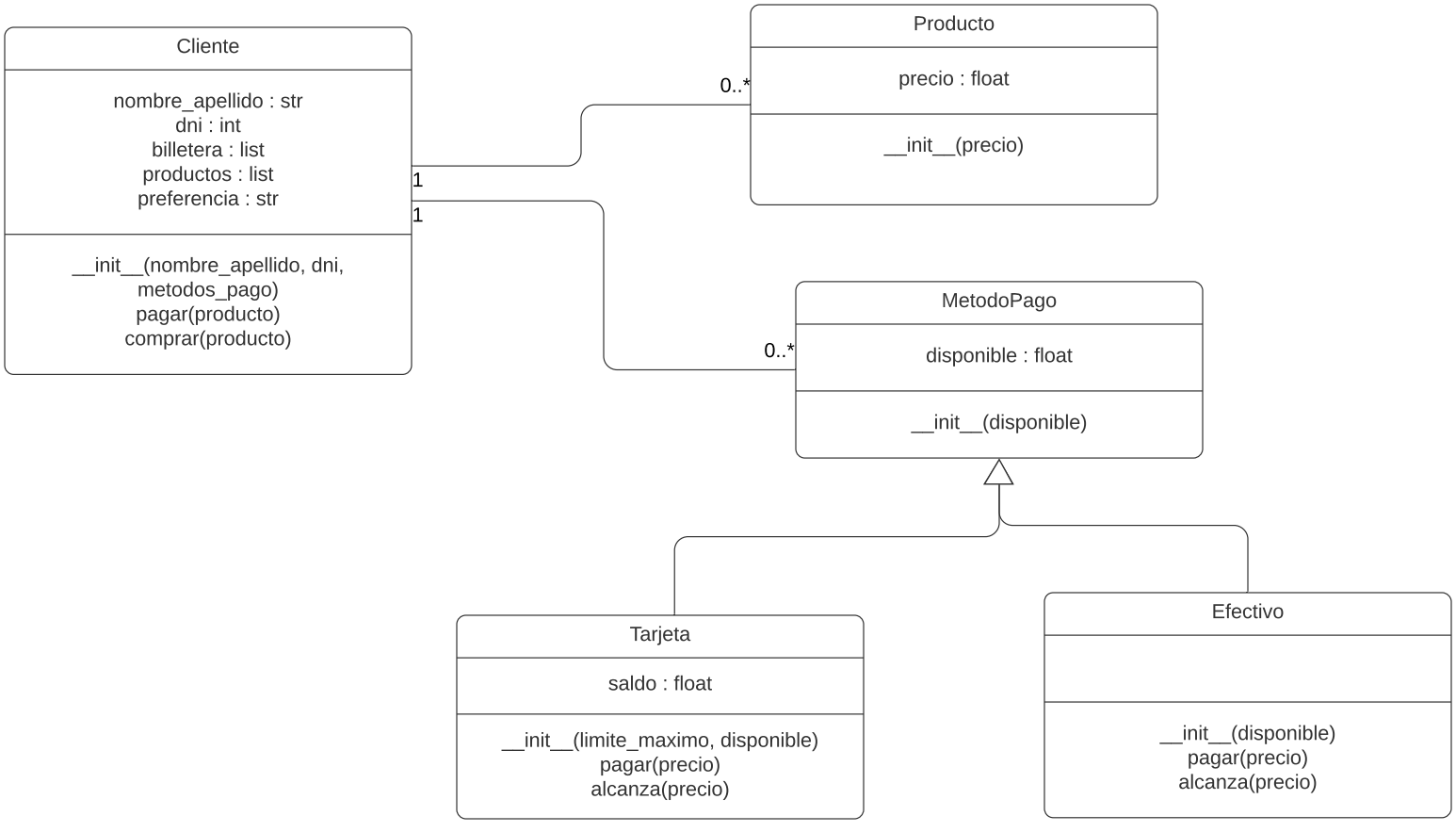
Modelo Final PV 2

Carlos Alberto Maceira Garcia Coni | December 8, 20



La solucion dada presenta las siguientes caracteristicas en cuanto a los conceptos de polimorfistmo, herencia, delcaratividad, expresividad, encapsulamiento y delegación:

- El Polimorfismo se aplica mediante el uso de metodos de identica firma con implementaciones que pueden ser diferentes en clases diferentes. Esta técnica permite una facil extensividad del código y reduce de realizar cambios o correccion de errores. En la solucion el polimorfismo está ausente, pero podria aplicarse en la implementacion de los métodos de pago.
- Herencia: se utiliza herencia en el código, pero de una forma incorrecta o al menos poco util. Se deriva un metodo de pago de otro, en vez de derivar todos de una superclase. Esta diferencia conceptual puede conllevar a la introduccion de errores al introducir modificaciones de comportamiento en una clase que no debería tener impacto en otra: por ejemplo usar el método gastar en la clase tarjeta.
- Declaratividad: La declaratividad es una característica de algunas herramientas que permiten o fuerzan la separación entre el conocimiento del dominio de un programa y la manipulación de dicho conocimiento. Dichas herramientas pueden ser de diversa naturaleza, tanto prácticas como lenguajes, entornos, frameworks, etc o conceptuales como paradigmas o arquitecturas. En esta solucion la declaratividad es baja: por ejemplo Efectivo y Tarjeta implementan métodos diferentes para hacer conceptualmente lo mismo: pagar la compra.
- Expresividad: La expresividad puede definirse informalmente con la heurística “el nivel de lindez del código”. En otras palabras, escribir un código expresivo es poner atención a las cuestiones que hacen que este código fuente sea más fácil de entender por una persona. El uso de la cadena self.preferencia == "tarjeta" como condicion en el método pagar, junto con los if anidados para cada método de pago es una solucion proco expresiva. En esos if anidados se utiliza repedidamente la expresion self.tarjeta.limite_maximo - self.tarjeta.gastado, esto podria convertirse en un comportamiento de la clase Tarjeta, el decir si puede o no afrontar un gasto con su crédito disponible. Por otro lado __init__ de la clase cliente tiene dos parámetros (tarjeta y efectivo) que no se utilizan.
- Encapsulamiento: es el principio mediante el cual se acceden a los atributos de una clase a través de sus métodos. En la solución la clase Cliente "opera" con atributos de la clase Tarjeta y de la clase Efectivo cuando implementa el método pagar. Esto rompe el encapsulamiento de la clase.
- Delegación: La delegación es un mecanismo, usado en la programación orientada a objetos, por medio de la cual una clase delega en otra una determinada funcionalidad. Hay delegación en la solución propuesta ya que el pago lo realiza la clase Tarjeta o la clase Efectivo.


Cliente
nombre_apellido : str
dni : int
billetera : list
productos : list
preferencia : str
__init__(nombre_apellido, dni, metodos_pago)
pagar(producto)
comprar(producto)
Producto
precio : float
__init__(precio)
0..*
1
1
0..*
MetodoPago
disponible : float
__init__(disponible)
Tarjeta
saldo : float
__init__(limite_maximo, disponible)
pagar(precio)
alcanza(precio)
Efectivo
__init__(disponible)
pagar(precio)
alcanza(precio)