Nombre: Nicolk Anelca Diaz Hernandez

Ficha:

**Investigación teórica**

1. ¿Qué es la Programación Orientada a Objetos (POO) y cuáles son sus ventajas frente a la programación estructurada?

Es un modelo o paradigma de programación, que nos permite organizar el código pensando en la problemática como una relación quiere “cosas” denominadas objetos. Los objetos se trabajan utilizando “las clases”.

**Ventajas**

* Facilita su mantenimiento y su comprensión es orientada al mundo real.
* Énfasis en la abstracción de datos.
* Las funciones y los datos son encapsulados en una entidad.

1. ¿Qué es una clase? ¿Qué es un objeto? Explica con tus palabras y da un ejemplo.

**Clase:** Es una plantilla o molde que define las características y comportamientos que tendrán los objetos. Ej:Persona

**Objeto**: Son características específicas que tendrá esta clase asignada y el describirla es un comportamiento. Ej: Persona1=persona (“Juan”,25).

1. Define los conceptos de atributo, método y constructor en POO.

**Atributo**: Son las propiedades especificas que un objeto (como una persona) puede tener.

Método: Son acciones que un objeto puede realizar o un comportamiento que tengan.

**Constructor:** Método especial que inicializa el objeto cuando se crea. En Python es: \_ \_init\_ \_()

1. Explica qué significa encapsulación, herencia, polimorfismo y abstracción.

**Encapsulación:** Permite ocultar los detalles internos de un objeto y controlar el acceso a sus datos. Es decir, poner los datos dentro de una cápsula y decidir quién puede verlos o modificarlos.

**Herencia**: La herencia es un proceso técnico que categoriza clases de objetos, o que forma nuevas clases a partir de otras ya existentes.

**Polimorfismo:** Se trata de un proceso técnico en el que las entidades o funciones pueden operar de la misma manera, pero pasan por un proceso de implementación diferente entre sí y en distintos momentos del desarrollo

**Abstracción:** La abstracción visualiza el comportamiento del objeto y su dependencia de otro elemento del diagrama.

1. ¿Qué es UML y para qué sirven los diagramas de clases, de objetos, de secuencia y de actividad?

El Lenguaje de Modelado Unificado (UML) es un estándar utilizado en el mundo del desarrollo de software para visualizar, especificar, construir y documentar los artefactos de un sistema. La importancia del UML radica en su capacidad para facilitar la comunicación entre los diversos actores involucrados en el ciclo de vida de un proyecto, como analistas, diseñadores, programadores y clientes.

1. **Diagrama de Clases:** Este diagrama muestra la estructura estática de un sistema, representando las clases, atributos y relaciones entre ellas. Es útil para visualizar la arquitectura general de un sitio web.
2. **Diagrama de objeto:** Muestra un vista completa o parcial de los objetos de un sistema en un instante de ejecución determinado. Especifica los detalles del modelo, ilustra las estructuras de datos/objetos del sistema. Además, este podría cambiar a medida que se ejecute el programa.
3. **Diagrama de Secuencia:** Este diagrama ilustra la interacción entre los distintos componentes o actores en el sistema a lo largo del tiempo. Es útil para entender el flujo de trabajo en una página web o la comunicación entre el cliente y el servidor.
4. **Diagrama de Actividad:** Este tipo de diagrama muestra el flujo de control en un proceso o actividad dentro del sistema. Puede ser útil para representar la navegación en un sitio web o los pasos que sigue un usuario para completar una tarea.
5. ¿Cuáles son las diferencias principales al implementar POO en Python, Java y C++?

En **Python**, no necesitas declarar el tipo de datos (como si algo es un número o una cadena), porque Python lo detecta automáticamente. Además, Python permite herencia múltiple, o sea que una clase puede heredar de varias al mismo tiempo. No existe palabras claves para proteger un atributo sino \_ \_ o \_ que no se puede tocar desde fuera.

En **Java**, todo es más estructurado, se debe declarar el tipo de variable de cada una, y usar modificadores como **private, public o protected** para controlar quien puede acceder a que parte de tu clase. Java no permite herencia múltiple directamente pero la compensa con un llamado interfaces, que son como contratos que las clases pueden seguir. En Java toca escribir clases mas detalladas y largas que en Python.

En **C++,** en este lenguaje de programación debes declarar tipos, igual que en Java, y puedes usar modificadores de acceso. C++ sí permite herencia múltiple, como Python, pero también te obliga a manejar la memoria tú misma: si creas un objeto con **new**, tienes que destruirlo con **delete**. Además, C++ permite sobrecargar operadores (como +, ==, etc), lo que te deja personalizar como se comportan tus objetos en operaciones.

1. Elige un mini-proyecto (ejemplo: biblioteca, gestión de cursos o tienda online) y responde:

Mi mini proyecto es una papeleria llamada “Norma”

o ¿Qué clases identificas en el problema?

Cliente

Producto

Vendedor

Detalleventa

Venta

o ¿Qué atributos y métodos tendría cada clase?

Clase:Cliente

Atributos: - id\_cliente:int

- nombre:string

-apellido:string

- Correo electrónico:string

Métodos: +consultarVenta()

Clase: Venta

Atributos:

-idVenta:int

-Costo\_venta:float

-Fecha\_venta:date

-Vendedor:Vendedor

-Detalles:Detalleventa

Métodos:

+agregar\_prpoductos()

+calcular\_total()

+mostrar\_detalle()

Detallesventa:

Atributos:

-iddetallesVenta:int

-cantidad:int

Producto:Producto

Métodos: +calcular\_subtotal()

Producto:

Atributo:

+idProducto:int

+descripción:string

+precio:double

+marca:string

Métodos: +mostrar\_informacion()

o ¿Qué relaciones existen entre ellas?

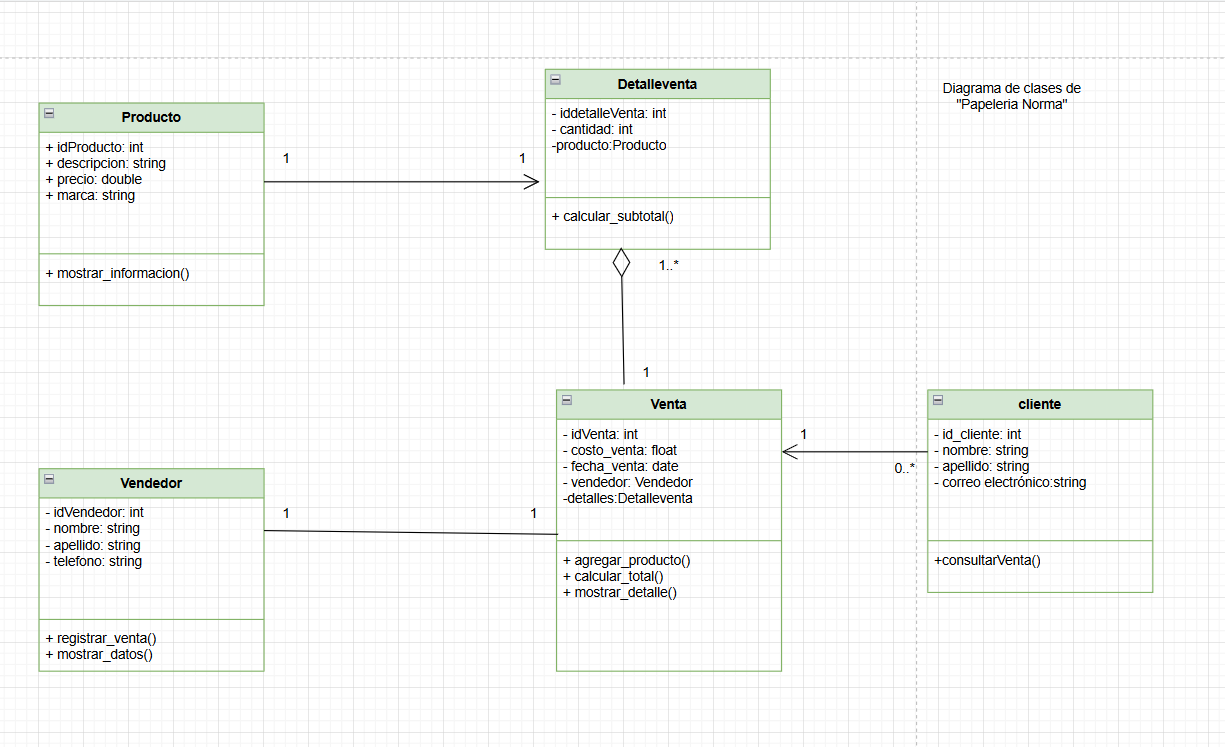
Producto 1-1 asociacion Detalleventa

Vendedor 1-1 asociacion Venta

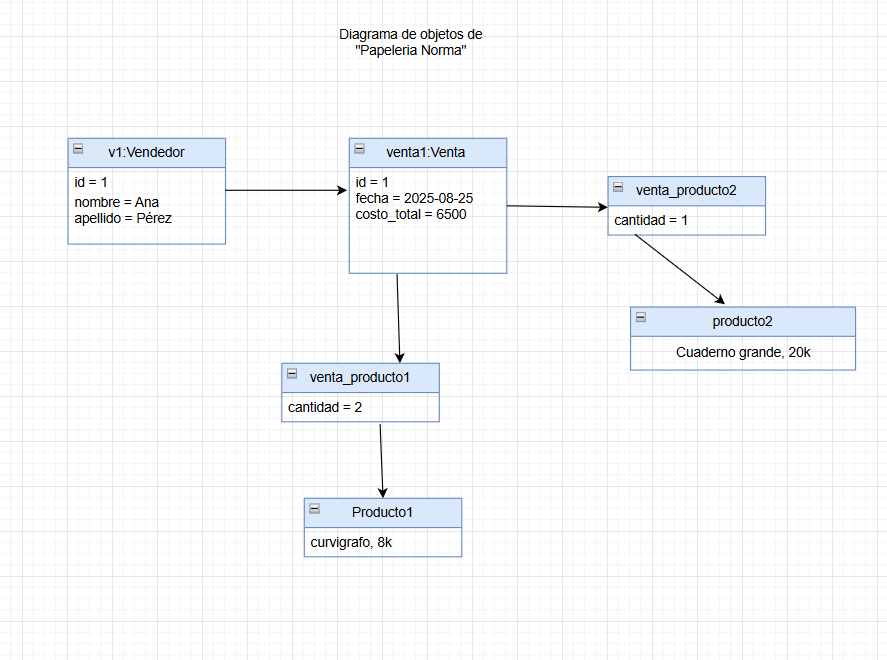
Venta 1-0…\* asociación Cliente

Detalleventa 1…\*-1 agregacion Venta

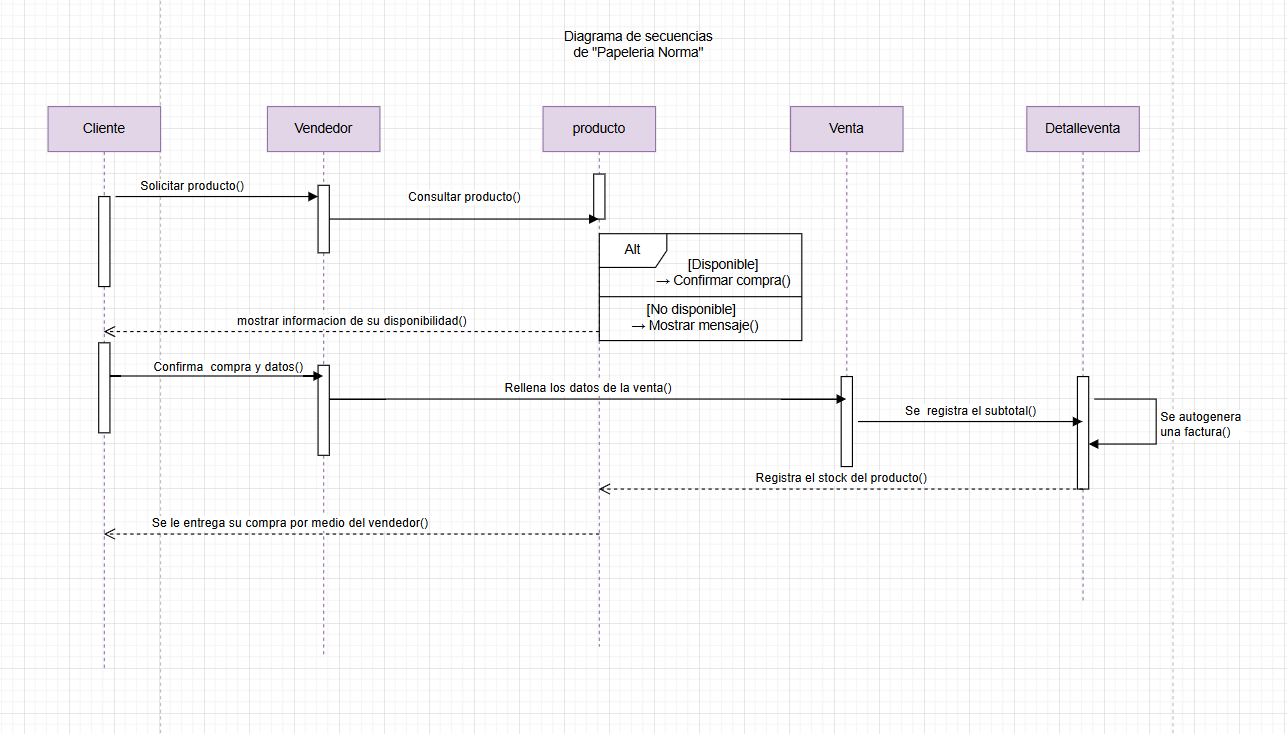
**Diagrama de clase:**

****

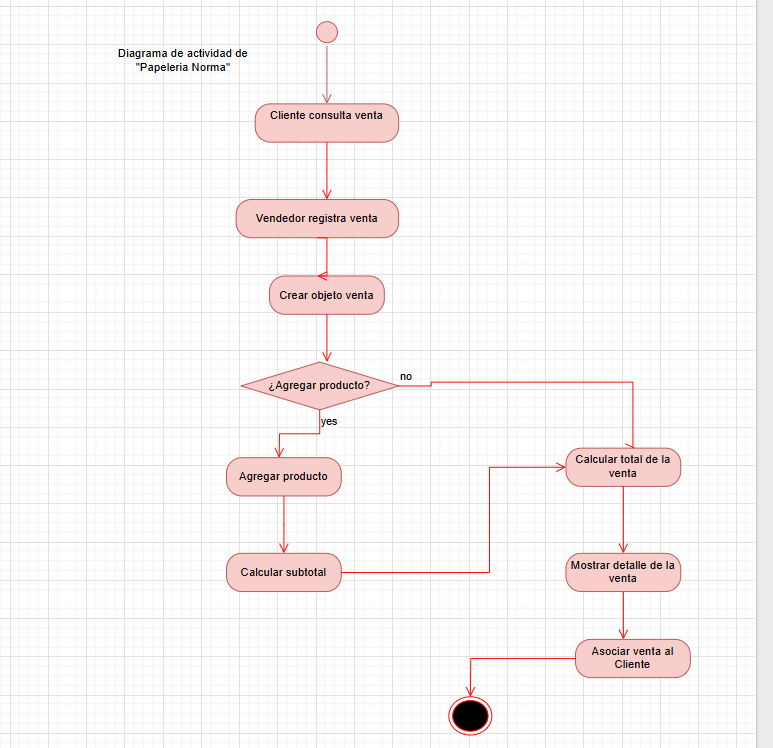
**Diagrama de objeto:**

****

**Diagrama de secuencia:**

****

**Diagrama de actividad:**

****

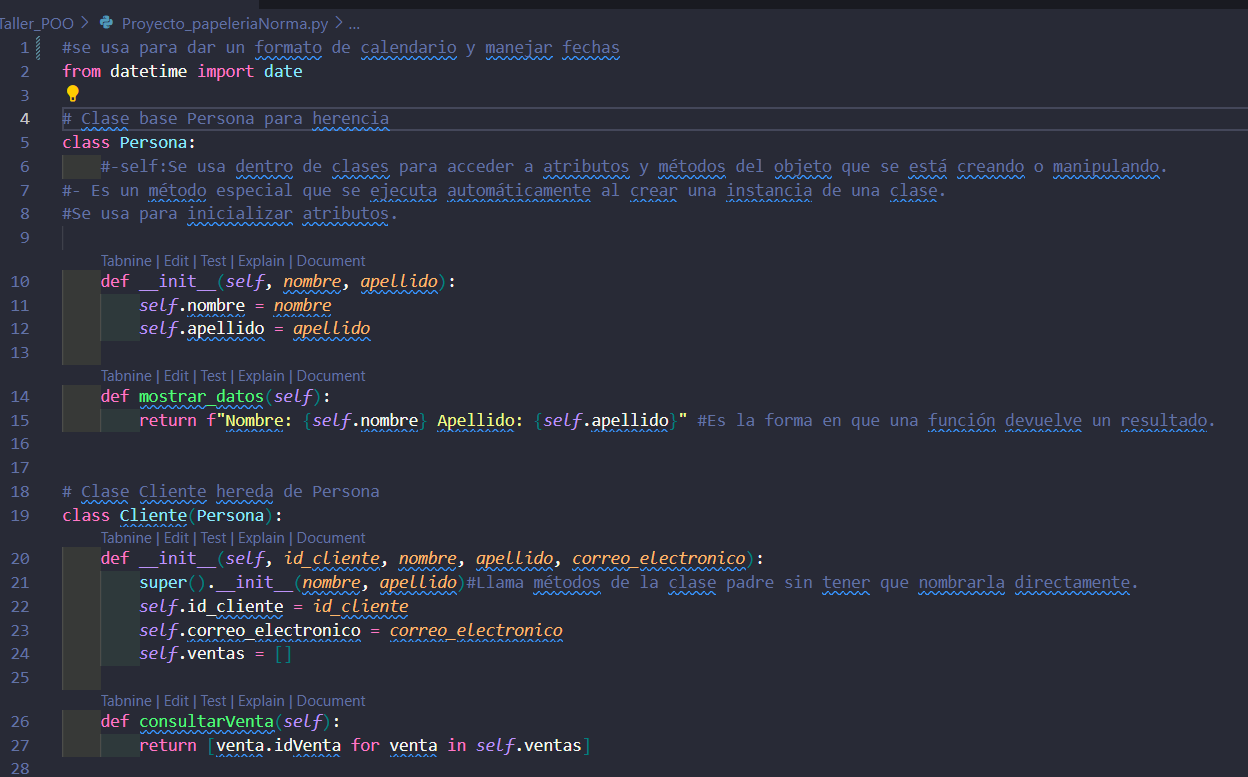
**Parte 3 — Implementación en Python**

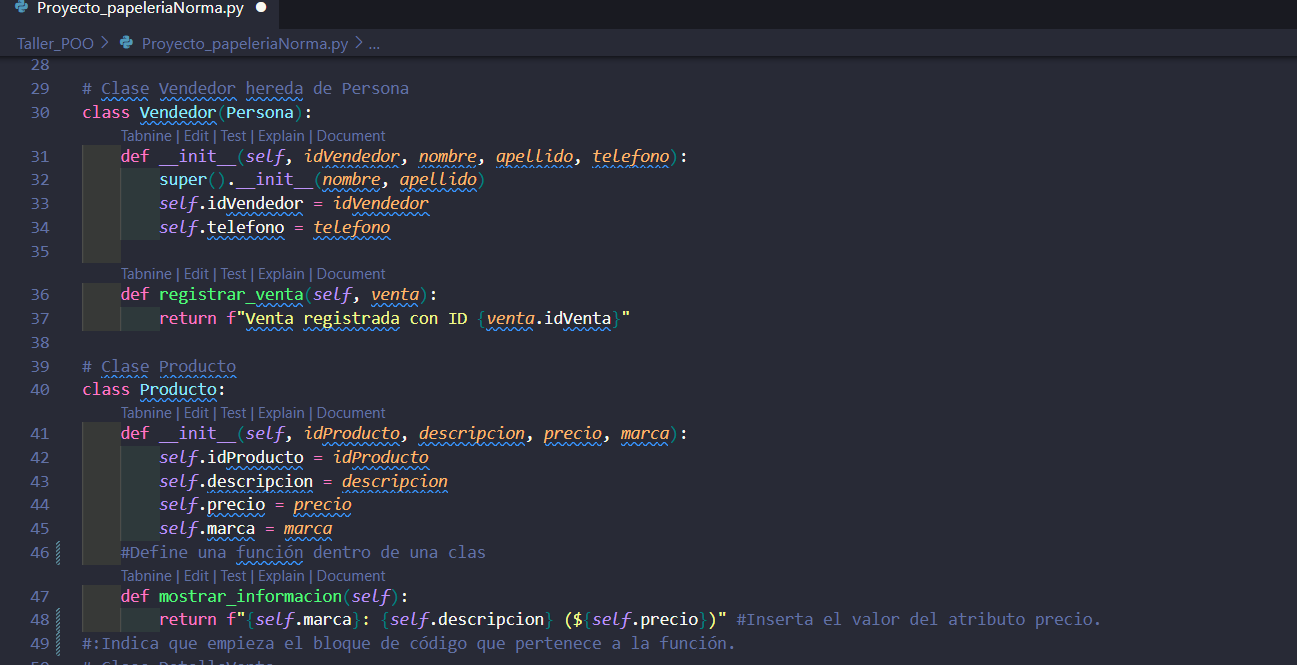
**12.** Implementa en Python las clases de tu diagrama**.**

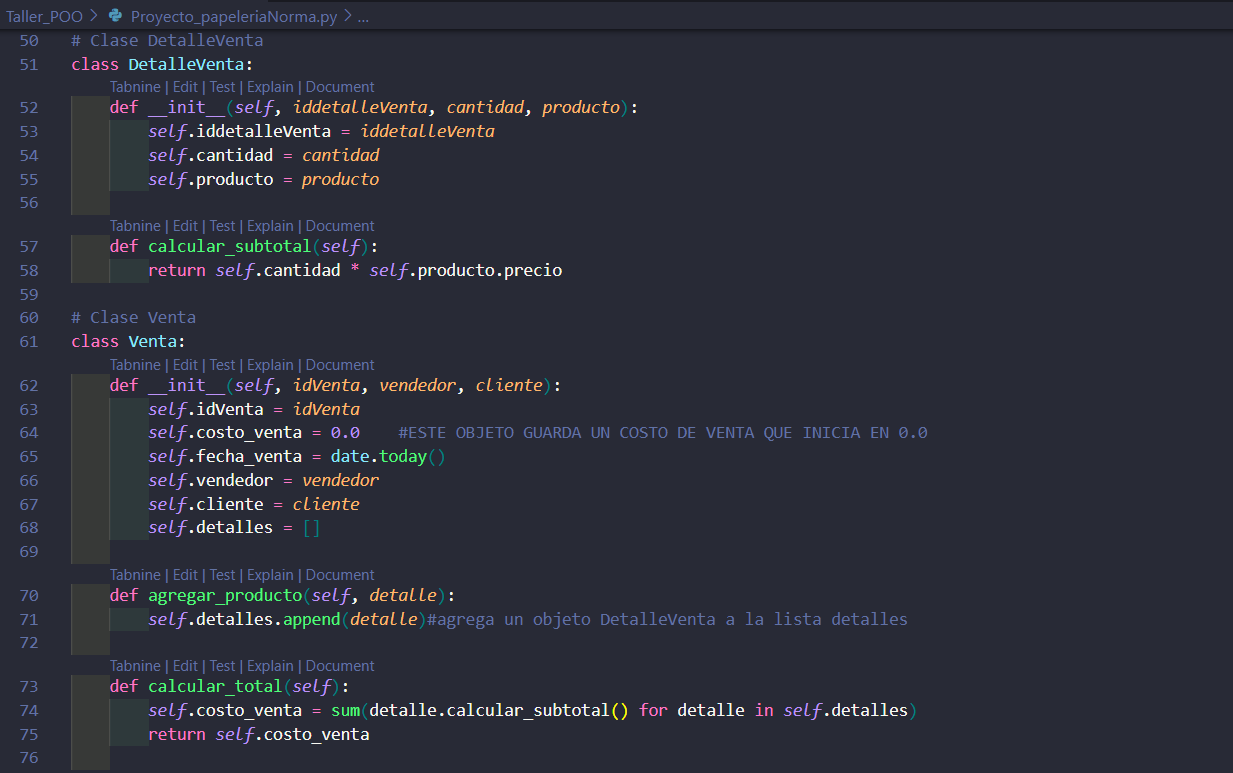
**13.** Agrega atributos, métodos y una relación de herencia**.**

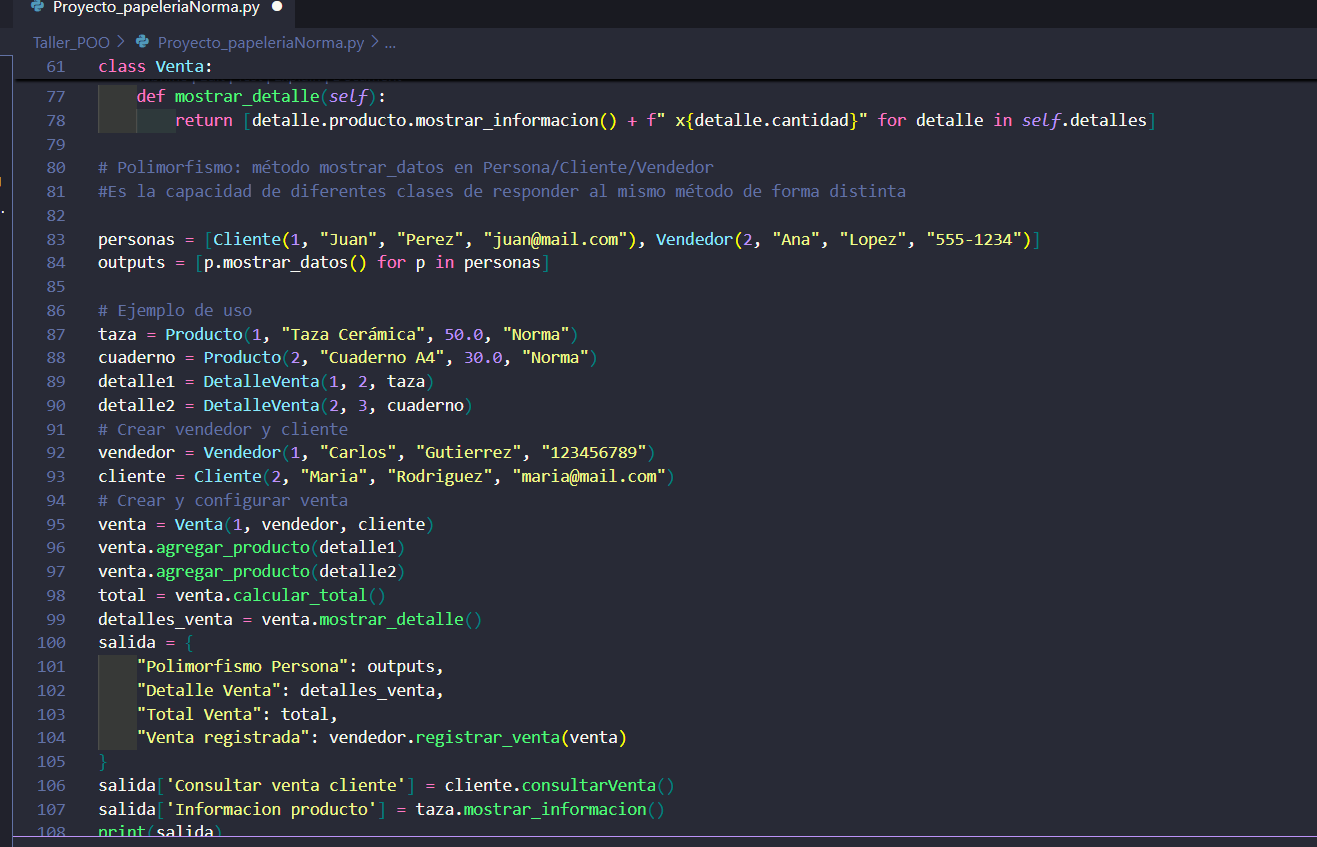
**14.** Muestra un ejemplo de polimorfismo en tu código.

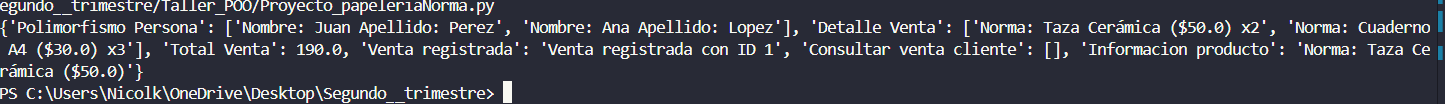
**15.** Ejecuta tu programa y muestra la salida.

****

****

****

****

****

**Parte 5 — Reflexión Final**

**20.** ¿Qué similitudes y diferencias encontraste entre Python, Java y C++ al trabajar con POO?

En los tres lenguajes se pueden usar los mismos conceptos de POO como clases, objetos, herencia y polimorfismo. Sin embargo, sentí que en Python todo es más simple, porque la sintaxis es corta y no me obliga a declarar los tipos. En Java todo es más formal, se escribe más código para lo mismo, pero a la vez se siente más organizado. Y en C++ lo más complicado es que además de la POO toca manejar cosas como la memoria y los punteros, lo cual lo hace más difícil que los otros dos.

**21.** ¿Qué dificultades tuviste al diseñar los diagramas UML?

La mayor dificultad fue representar de forma clara las decisiones y ciclos, ya que en el diagrama de actividad y el de secuencia los caminos podían variar.

**22**. ¿Qué parte del taller consideras más importante para tu formación como técnico en desarrollo de software?

Lo más importante fue aprender a traducir un problema real en diagramas UML y luego implementarlo en código

**23.** Propón una mejora o extensión para tu ejercicio (ej.: agregar persistencia, interfaz gráfica o más clases).

Una mejora interesante sería agregar persistencia en una base de datos, para que las ventas, clientes y productos no se pierdan al cerrar el programa. Además,. También se podría ampliar el sistema con más clases, por ejemplo, una clase *Factura* o una clase *Inventario* para controlar el stock de productos.