# Practica Unidad 1
# Programación Avanzada

Nombre del alumno:
Roberto Velez Flores

Materia: Programación avanzada

Maestro: Jimmy

**Justificación del programa cine_proy.ipynb**

El propósito de este código es simplemente el de gestionar y tener en un orden los servicios que puede ofrecer un cine como promociones, reservar asientos, la consulta de películas, sirve para un cliente como para un empleado donde cada uno tiene un rol y solo pueden hacer ciertas opciones como el empleado quien puede modificar que películas se presentaran en un horario que establezca, poder modificar las promociones de un producto, mientras que el usuario puede reservar y cancelar y cuantos cupos puede reservar.

**Clases a ocupar para que el código funcione:**

***Clase Persona:**
**Atributos:**
nombre: Nombre de la persona.
correo: Correo electrónico de la persona.

**Métodos:**
registrar(): Registra a la persona y la agrega a la lista de personas.
actualizar_datos(nombre, correo): Actualiza los datos de la persona.
personas_registradas(): Muestra la lista de personas registradas.

***Clase Usuario (hereda de Persona):**
**Atributos:**
historial_reservas: Lista de reservas realizadas por el usuario.

**Métodos:**
reservar(funcion, asientos, promocion=None): Reserva asientos para una función y aplica una promoción (si la hay).
cancelar_reserva(funcion): Cancela una reserva previamente hecha.

***Clase Empleado (hereda de Persona):**
**Atributos:**
rol: Rol del empleado.

**Métodos:**
agregar_funcion(funcion): Agrega una nueva función de cine.
modificar_promocion(promocion, nuevo_descuento, nuevas_condiciones): Modifica los detalles de una promoción.

***Clase Espacio:**
**Atributos:**
capacidad: Capacidad del espacio.
identificador: Identificador único del espacio.

**Métodos:**
descripcion(): Muestra la descripción del espacio (capacidad y ID).

***Clase Sala (hereda de Espacio):**
**Atributos:**
tipo: Tipo de la sala (ej. "3DX", "Tradicional").
disponibilidad: Si la sala está disponible o no.

**Métodos:**
Consultardisponibilidad(): Consulta si la sala está disponible.

***Clase ZonaComida (hereda de Espacio):**

**Atributos:**
menu: Diccionario con los productos disponibles.
insumos: Ingredientes necesarios para los productos.
precios: Precio de cada producto en el menú.
ventas: Lista de ventas realizadas.

**Métodos:**
agregar_producto(producto, cantidad, precio): Agrega un producto al menú o aumenta su cantidad si ya existe.
vender_producto(producto, cantidad): Vende una cantidad de un producto y registra la venta.
mostrar_ventas(): Muestra todas las ventas realizadas.
mostrar_menu(): Muestra el menú de productos disponibles.

***Clase Pelicula:**
**Atributos:**
titulo: Título de la película.
genero: Género de la película.
duracion: Duración de la película en minutos.

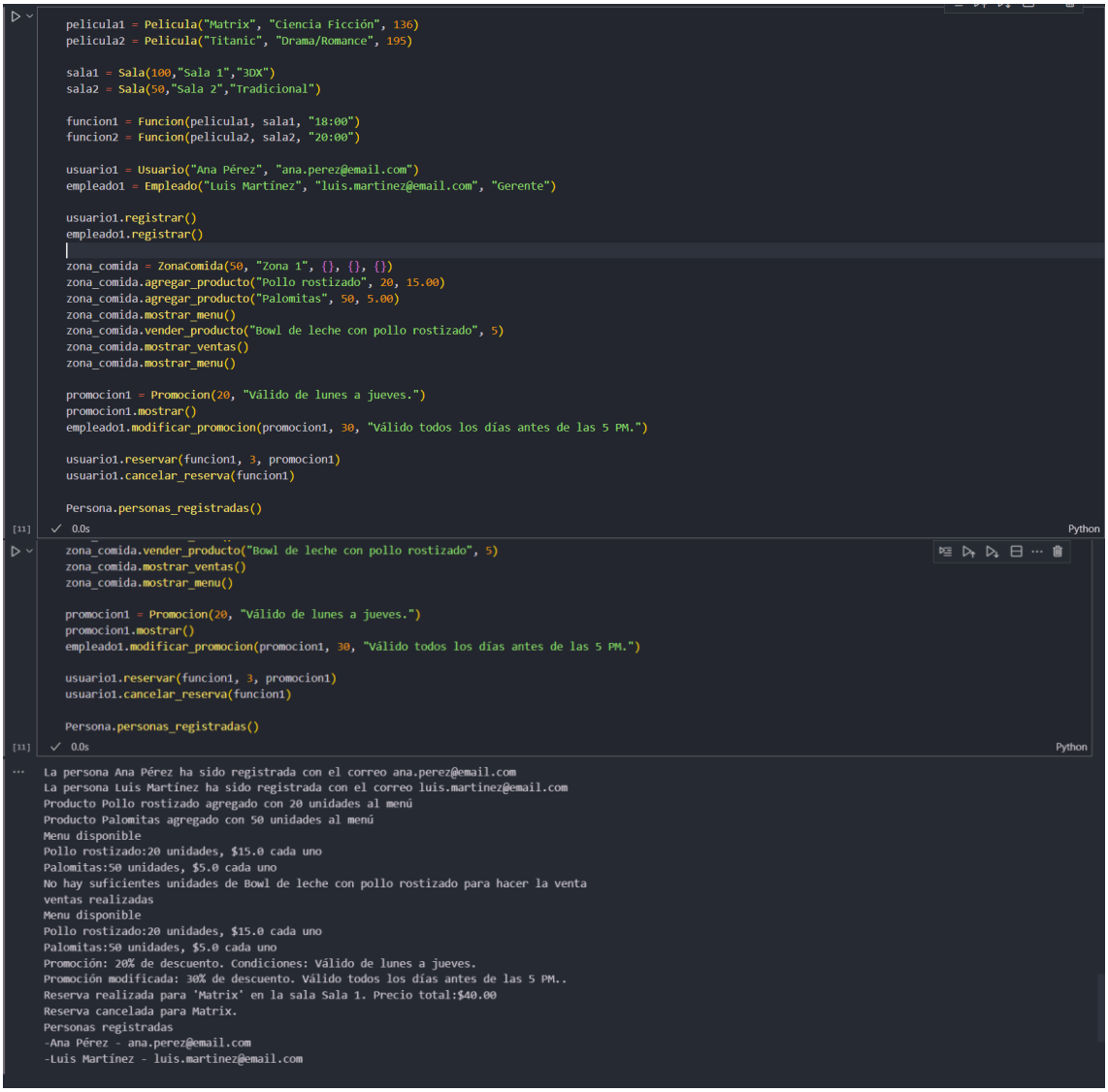
***Clase Funcion:**
**Atributos:**
pelicula: Instancia de la clase Pelicula que representa la película que se proyecta.
sala: Instancia de la clase Sala donde se proyecta la película.
hora: Hora en la que se proyecta la película.
asientos_disponibles: Cantidad de asientos disponibles para la función (por defecto igual a la capacidad de la sala).

***Clase Promocion:**
**Atributos:**
descuento: Porcentaje de descuento de la promoción.
condiciones: Condiciones de la promoción.

**Métodos:**
mostrar(): Muestra los detalles de la promoción.

***Clase Reserva (hereda de Usuario y Funcion):**
**Atributos:**
confirmacion: Si la reserva fue confirmada o no.
precio_total: Precio total de la reserva después de aplicar descuentos (si los hay).

**Métodos:**
calcular_finprecio(): Calcula el precio total de la reserva, aplicando un descuento si es necesario.
realizar_reserva(): Realiza la reserva si hay asientos disponibles.
cancelar_reserva(): Cancela la reserva si está confirmada.



```
pelicula1 = Pelicula("Matrix", "Ciencia Ficción", 136)
pelicula2 = Pelicula("Titanic", "Drama/Romance", 195)

sala1 = Sala(100,"Sala 1","3DX")
sala2 = Sala(50,"Sala 2","Tradicional")

funcion1 = Funcion(pelicula1, sala1, "18:00")
funcion2 = Funcion(pelicula2, sala2, "20:00")

usuario1 = Usuario("Ana Pérez", "ana.perez@email.com")
empleado1 = Empleado("Luis Martínez", "luis.martinez@email.com", "Gerente")

usuario1.registrar()
empleado1.registrar()
zona_comida = ZonaComida(50, "Zona 1", {}, {}, {})
zona_comida.agregar_producto("Pollo rostizado", 20, 15.00)
zona_comida.agregar_producto("Palomitas", 50, 5.00)
zona_comida.mostrar_menu()
zona_comida.vender_producto("Bowl de leche con pollo rostizado", 5)
zona_comida.mostrar_ventas()
zona_comida.mostrar_menu()

promocion1 = Promocion(20, "Válido de lunes a jueves.")
promocion1.mostrar()
empleado1.modificar_promocion(promocion1, 30, "Válido todos los días antes de las 5 PM.")

usuario1.reservar(funcion1, 3, promocion1)
usuario1.cancelar_reserva(funcion1)

Persona.personas_registradas()
```

[11] ✓ 0.0s Python

```
zona_comida.vender_producto("Bowl de leche con pollo rostizado", 5)
zona_comida.mostrar_ventas()
zona_comida.mostrar_menu()

promocion1 = Promocion(20, "Válido de lunes a jueves.")
promocion1.mostrar()
empleado1.modificar_promocion(promocion1, 30, "Válido todos los días antes de las 5 PM.")

usuario1.reservar(funcion1, 3, promocion1)
usuario1.cancelar_reserva(funcion1)

Persona.personas_registradas()
```

[11] ✓ 0.0s Python

```
La persona Ana Pérez ha sido registrada con el correo ana.perez@email.com
La persona Luis Martínez ha sido registrada con el correo luis.martinez@email.com
Producto Pollo rostizado agregado con 20 unidades al menú
Producto Palomitas agregado con 50 unidades al menú
Menu disponible
Pollo rostizado:20 unidades, $15.0 cada uno
Palomitas:50 unidades, $5.0 cada uno
No hay suficientes unidades de Bowl de leche con pollo rostizado para hacer la venta
ventas realizadas
Menu disponible
Pollo rostizado:20 unidades, $15.0 cada uno
Palomitas:50 unidades, $5.0 cada uno
Promoción: 20% de descuento. Condiciones: Válido de lunes a jueves.
Promoción modificada: 30% de descuento. Válido todos los días antes de las 5 PM..
Reserva realizada para 'Matrix' en la sala Sala 1. Precio total:$40.00
Reserva cancelada para Matrix.
Personas registradas
-Ana Pérez - ana.perez@email.com
-Luis Martínez - luis.martinez@email.com
```

Link desde google colab:
https://colab.research.google.com/github/RVF2325/PrograADVRV/blob/main/cine_proy.ipynb
Link desde Git hub:
https://github.com/RVF2325/PrograADVRV/blob/main/cine_proy.ipynb

Justificación del programa Pedidos_cafeteria.ipynb
Este programa es de gran utilidad como gestionar el que vas a comer, bebidas, postres, promociones, conocer el pago total funciona como las apps actuales de comida que hay hoy en dia

Clases a ocupar para que el código funcione:

***Clase Persona:**
**Atributos:**
nombre: Nombre de la persona.
correo: Correo electrónico de la persona.

**Métodos:**
registrar(): Registra a la persona en la lista global.
actualizar_datos(nombre, correo): Actualiza los datos de la persona.
personas_registradas(): Muestra la lista de personas registradas.

***Clase Cliente (hereda de Persona):**
**Atributos:**
consultar_historial: Lista de pedidos realizados por el cliente.

**Métodos:**
hacer_pedido(producto, cantidad, inventario): Realiza un pedido si hay suficiente stock del producto.

***Clase Empleado (hereda de Persona):**
**Atributos:**
rol: Rol del empleado (ej. "Cajero", "Gerente").

**Métodos:**
agregar_comida(platillo, inventario): Agrega comida al inventario.
agregar_bebida(bebida, inventario): Agrega bebida al inventario.

***Clase Producto_Base:**
**Atributos:**
nombre: Nombre del producto.
precio: Precio del producto.

***Clase Bebida (hereda de Producto_Base):**
**Atributos:**
tamaño: Tamaño de la bebida (ej. "Pequeño", "Grande")
tipo: Tipo de bebida (ej. "Refresco", "Jugo").
personalizado: Ingredientes adicionales que el cliente puede añadir.

**Métodos:**
agregar_ingrediente(ingrediente): Agrega un ingrediente adicional a la bebida.

mostrar_informacion(): Muestra la información de la bebida, incluyendo los ingredientes adicionales.

***Clase Postre (hereda de Producto_Base):**
**Atributos:**
es_vegano: Si el postre es vegano o no.
sin_gluten: Si el postre es sin gluten o no.

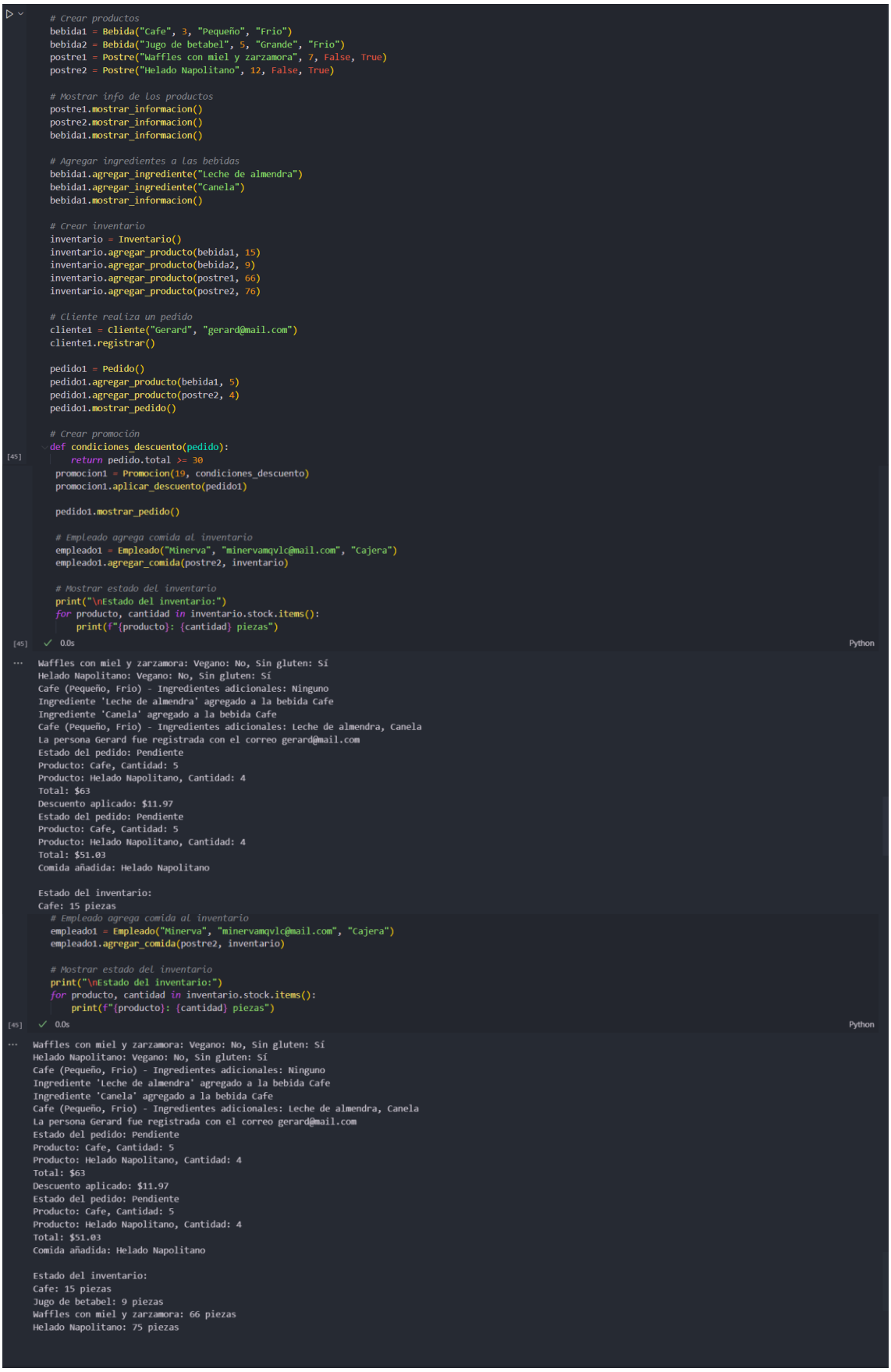
**Métodos:**
mostrar_informacion(): Muestra la información del postre, incluyendo si es vegano o sin gluten.

***Clase Inventario:**
**Atributos:**
stock: Diccionario con los productos y sus cantidades disponibles.

**Métodos:**
agregar_producto(producto, cantidad): Agrega un producto al inventario.
chequear_stock(producto, cantidad): Verifica si hay suficiente stock de un producto.
actualizar_stock(producto, cantidad): Actualiza el stock después de un pedido.

***Clase Pedido:**
**Atributos:**
productos: Lista de productos en el pedido.
estado: Estado del pedido (ej. "Pendiente").
total: Total a pagar por el pedido.
**Métodos:**
agregar_producto(producto, cantidad): Agrega un producto al pedido.
mostrar_pedido(): Muestra los detalles del pedido, incluidos los productos y el total.
***Clase Promocion:**
**Atributos:**
descuento: Porcentaje de descuento de la promoción.
condiciones: Función que define las condiciones para aplicar el descuento.

**Métodos:**
aplicar_descuento(pedido): Aplica el descuento al pedido si se cumplen las condiciones.



```python
# Crear productos
bebida1 = Bebida("Cafe", 3, "Pequeño", "Frio")
bebida2 = Bebida("Jugo de betabel", 5, "Grande", "Frio")
postre1 = Postre("Waffles con miel y zarzamora", 7, False, True)
postre2 = Postre("Helado Napolitano", 12, False, True)

# Mostrar info de los productos
postre1.mostrar_informacion()
postre2.mostrar_informacion()
bebida1.mostrar_informacion()

# Agregar ingredientes a las bebidas
bebida1.agregar_ingrediente("Leche de almendra")
bebida1.agregar_ingrediente("Canela")
bebida1.mostrar_informacion()

# Crear inventario
inventario = Inventario()
inventario.agregar_producto(bebida1, 15)
inventario.agregar_producto(bebida2, 9)
inventario.agregar_producto(postre1, 66)
inventario.agregar_producto(postre2, 76)

# Cliente realiza un pedido
cliente1 = Cliente("Gerard", "gerard@mail.com")
cliente1.registrar()

pedido1 = Pedido()
pedido1.agregar_producto(bebida1, 5)
pedido1.agregar_producto(postre2, 4)
pedido1.mostrar_pedido()

# Crear promoción
def condiciones_descuento(pedido):
    return pedido.total >= 30
promocion1 = Promocion(19, condiciones_descuento)
promocion1.aplicar_descuento(pedido1)

pedido1.mostrar_pedido()

# Empleado agrega comida al inventario
empleado1 = Empleado("Minerva", "minervamqvlc@mail.com", "Cajera")
empleado1.agregar_comida(postre2, inventario)

# Mostrar estado del inventario
print("\nEstado del inventario:")
for producto, cantidad in inventario.stock.items():
    print(f"{producto}: {cantidad} piezas")
```

[45] ✓ 0.0s Python

```
Waffles con miel y zarzamora: Vegano: No, Sin gluten: Sí
Helado Napolitano: Vegano: No, Sin gluten: Sí
Cafe (Pequeño, Frio) - Ingredientes adicionales: Ninguno
Ingrediente 'Leche de almendra' agregado a la bebida Cafe
Ingrediente 'Canela' agregado a la bebida Cafe
Cafe (Pequeño, Frio) - Ingredientes adicionales: Leche de almendra, Canela
La persona Gerard fue registrada con el correo gerard@mail.com
Estado del pedido: Pendiente
Producto: Cafe, Cantidad: 5
Producto: Helado Napolitano, Cantidad: 4
Total: $63
Descuento aplicado: $11.97
Estado del pedido: Pendiente
Producto: Cafe, Cantidad: 5
Producto: Helado Napolitano, Cantidad: 4
Total: $51.03
Comida añadida: Helado Napolitano

Estado del inventario:
Cafe: 15 piezas
```

```python
# Empleado agrega comida al inventario
empleado1 = Empleado("Minerva", "minervamqvlc@mail.com", "Cajera")
empleado1.agregar_comida(postre2, inventario)

# Mostrar estado del inventario
print("\nEstado del inventario:")
for producto, cantidad in inventario.stock.items():
    print(f"{producto}: {cantidad} piezas")
```

[45] ✓ 0.0s Python

```
Waffles con miel y zarzamora: Vegano: No, Sin gluten: Sí
Helado Napolitano: Vegano: No, Sin gluten: Sí
Cafe (Pequeño, Frio) - Ingredientes adicionales: Ninguno
Ingrediente 'Leche de almendra' agregado a la bebida Cafe
Ingrediente 'Canela' agregado a la bebida Cafe
Cafe (Pequeño, Frio) - Ingredientes adicionales: Leche de almendra, Canela
La persona Gerard fue registrada con el correo gerard@mail.com
Estado del pedido: Pendiente
Producto: Cafe, Cantidad: 5
Producto: Helado Napolitano, Cantidad: 4
Total: $63
Descuento aplicado: $11.97
Estado del pedido: Pendiente
Producto: Cafe, Cantidad: 5
Producto: Helado Napolitano, Cantidad: 4
Total: $51.03
Comida añadida: Helado Napolitano

Estado del inventario:
Cafe: 15 piezas
Jugo de betabel: 9 piezas
Waffles con miel y zarzamora: 66 piezas
Helado Napolitano: 75 piezas
```

link desde Google colab

https://colab.research.google.com/github/RVF2325/PrograADVRV/blob/main/Pedidos_cafeteria.ipynb

Link desde Github
https://github.com/RVF2325/PrograADVRV/blob/main/Pedidos_cafeteria.ipynb

Justificacion del programa Biblioteca digital.ipynb

Un programa para la gestión de libros donde son prestados y si te llevas de mas tiempos seras sancionado, también es posible llevar libros entre sucursales

```python
libro1 = Libro(autor="Thomas Harris", genero="Thriller psicológico")
libro2 = Libro(autor="Dante Aligheri", genero="Ficcion Universal")
revista1 = Revista(edicion=8, periodicidad="Mensual")
material_digital1 = MaterialDigital(tipo_archivo="PDF", enlace_descarga="www.Hogarlibro.com")

sucursal1 = Sucursal(nombre="Surcusal Central")
sucursal2 = Sucursal(nombre="Sucursal Norte")
# Crear el catálogo
sucursal1.agregar_material(libro1)
sucursal1.agregar_material(libro2)
sucursal1.agregar_material(revista1)
sucursal1.agregar_material(material_digital1)

# Crear un usuario
usuario1 = Usuario(nombre="Carlos")

# Mostrar todos los materiales disponibles en el catálogo
print("Consulta general del catálogo:")
for material in sucursal1.consultar_catalogo():
    if isinstance(material, Libro):
        print(f"Libro: Autor: {material.autor}, Género: {material.genero}")
    elif isinstance(material, Revista):
        print(f"Revista: Edición: {material.edicion}, Periodicidad: {material.periodicidad}")
    elif isinstance(material, MaterialDigital):
        print(f"Material Digital: Tipo de archivo: {material.tipo_archivo}, Enlace: {material.enlace_descarga}")
# Usuario presta material
fecha_devolucion = datetime.now() + timedelta(days=7)
usuario1.prestar_material(libro1, fecha_devolucion)

# Crear préstamo
prestamo1 = Prestamo(usuario1, libro1, datetime.now() - timedelta(days=10), fecha_devolucion=fecha_devolucion)

# Crear penalización
penalizacion = Penalizacion(usuario1)
penalizacion.agregar_penalizacion(prestamo1.calcular_penalizacion())
    penalizacion.agregar_penalizacion(prestamo1.calcular_penalizacion())
    penalizacion.mostrar_penalizacion()

    # Transferir material entre sucursales
    sucursal1.transferir_material(libro1, sucursal2)

    # Mostrar el catálogo después de la transferencia
    print("\nCatálogo después de la transferencia:")
    print("\nSucursal Central:")
    for material in sucursal1.consultar_catalogo():
        if isinstance(material, Libro):
            print(f"Libro: Autor: {material.autor}, Género: {material.genero}")
        elif isinstance(material, Revista):
            print(f"Revista: Edición: {material.edicion}, Periodicidad: {material.periodicidad}")
        elif isinstance(material, MaterialDigital):
            print(f"Material Digital: Tipo de archivo: {material.tipo_archivo}, Enlace: {material.enlace_descarga}")
    print("\nSucursal Norte:")
    for material in sucursal2.consultar_catalogo():
        if isinstance(material, Libro):
            print(f"Libro: Autor: {material.autor}, Género: {material.genero}")
        elif isinstance(material, Revista):
            print(f"Revista: Edición: {material.edicion}, Periodicidad: {material.periodicidad}")
        elif isinstance(material, MaterialDigital):
            print(f"Material Digital: Tipo de archivo: {material.tipo_archivo}, Enlace: {material.enlace_descarga}")
```

[109] ✓ 0.0s Python

```
Consulta general del catálogo:
Libro: Autor: Thomas Harris, Género: Thriller psicológico
Libro: Autor: Dante Aligheri, Género: Ficcion Universal
Revista: Edición: 8, Periodicidad: Mensual
Material Digital: Tipo de archivo: PDF, Enlace: www.Hogarlibro.com
El usuario Carlos no tiene penalización
El material 'Libro' ha sido transferido de Surcusal Central a Sucursal Norte.

Catálogo después de la transferencia:
```

```python
        print(f"Libro: Autor: {material.autor}, Género: {material.genero}")
    elif isinstance(material, Revista):
        print(f"Revista: Edición: {material.edicion}, Periodicidad: {material.periodicidad}")
    elif isinstance(material, MaterialDigital):
        print(f"Material Digital: Tipo de archivo: {material.tipo_archivo}, Enlace: {material.enlace_descarga}")
```

[109] ✓ 0.0s Python

```
Consulta general del catálogo:
Libro: Autor: Thomas Harris, Género: Thriller psicológico
Libro: Autor: Dante Aligheri, Género: Ficcion Universal
Revista: Edición: 8, Periodicidad: Mensual
Material Digital: Tipo de archivo: PDF, Enlace: www.Hogarlibro.com
El usuario Carlos no tiene penalización
El material 'Libro' ha sido transferido de Surcusal Central a Sucursal Norte.

Catálogo después de la transferencia:

Sucursal Central:
Libro: Autor: Dante Aligheri, Género: Ficcion Universal
Revista: Edición: 8, Periodicidad: Mensual
Material Digital: Tipo de archivo: PDF, Enlace: www.Hogarlibro.com

Sucursal Norte:
Libro: Autor: Thomas Harris, Género: Thriller psicológico
```

Link desde google colab
https://colab.research.google.com/github/RVF2325/PrograADVRV/blob/main/Biblioteca_digital.ipynb

Link desde Github
https://github.com/RVF2325/PrograADVRV/blob/main/Biblioteca_digital.ipynb