Biblioteca Digital - Proyecto Python

Autor: Juan Pablo Gonzalez Trujillo
Fecha: Septiembre 2025
Lenguaje: Python
Tema: Programación Orientada a Objetos (POO)
Interfaz: Consola (CLI)

Descripción del Proyecto

Este proyecto simula el funcionamiento básico de una **biblioteca digital**, aplicando los 4 pilares fundamentales de la **Programación Orientada a Objetos (POO)** en Python.

El sistema permite:

- Crear y listar usuarios
- Crear y listar materiales (libros y revistas)
- Realizar préstamos y devoluciones
- Calcular multas por retraso
- Generar reportes de préstamos activos

Todo esto desde una **interfaz de línea de comandos (CLI)** interactiva.

Aplicación de los Pilares POO

- **Abstracción:**
  Se define una clase abstracta `Item`, que establece un modelo común para los materiales prestables.
- **Herencia:**
  Las clases `Libro` y `Revista` heredan de `Item`, reutilizando atributos y comportamientos.
- **Polimorfismo:**
  Los métodos `prestamo()` y `multa()` se implementan de manera distinta en `Libro` y `Revista`, pero se usan de forma uniforme.
- **Encapsulación:**
  Uso de `@property` y validaciones para proteger atributos como `stock` y `documento`, restringiendo acceso directo.

Capturas del Sistema

Menú principal (CLI)

```
--- MENÚ BIBLIOTECA ---
1. Agregar usuario
2. Agregar material
3. Crear préstamo
4. Devolver material
5. Ver préstamos con multa estimada
6. Listar usuarios y materiales
7. Salir
Elige una opción:
```

Gestión de usuarios

En el código:

```python
class Usuario:
    def __init__(self, nombre, documento):
        self.nombre = nombre
        self._documento = documento

    @property
    def documento(self):
        return self._documento
```

```python
class Biblioteca:
    def __init__(self):
        self.materiales = []
        self.usuarios = []
        self.prestamos = []

    def agregar_material(self, material):
        self.materiales.append(material)

    def agregar_usuario(self, usuario):
        for u in self.usuarios:
            if u.documento == usuario.documento:
                print("Documento ya registrado.")
                return
        self.usuarios.append(usuario)
```

En la ejecución:

```
Usuarios:
Juan Esteban - Documento: 15611
Erick - Documento: 21832
Camila - Documento: 33803
Mateo - Documento: 445534
tomas - Documento: 123456
```

Materiales registrados

En el código:

```python
class Libro(Item):
    def prestamo(self):
        return 14

    def multa(self):
        return 300

class Revista(Item):
    def prestamo(self):
        return 7

    def multa(self):
        return 200
```

```python
class Biblioteca:
    def __init__(self):
        self.materiales = []
        self.usuarios = []
        self.prestamos = []

    def agregar_material(self, material):
        self.materiales.append(material)
```

En la ejecución:

```
Materiales:
Libro: La Metamorfosis - Stock: 0
Revista: Ford - Stock: 1
Libro: Hobbit - Stock: 0
Libro: 1964 - Stock: 3
```

☑ Préstamo exitoso

En el código:

```python
def crear_prestamo(self, usuario, material, fecha_inicio):
    for p in self.prestamos:
        if p.usuario == usuario and p.material == material:
            print("Ese usuario ya tiene ese material prestado.")
            return

    if material.stock <= 0:
        print("No hay stock disponible.")
        return

    nuevo = Prestamo(usuario, material, fecha_inicio)
    self.prestamos.append(nuevo)
    material.stock -= 1
    print("Préstamo creado con éxito.")
```

En la ejecución:

```
--- MENÚ BIBLIOTECA ---
1. Agregar usuario
2. Agregar material
3. Crear préstamo
4. Devolver material
5. Ver préstamos con multa estimada
6. Listar usuarios y materiales
7. Salir
Elige una opción: 3
Documento del usuario: 21832
Título del material: 1964
Fecha de inicio (dd/mm/aaaa): 06/09/2025
Préstamo creado con éxito.
```

🔁 Devolución con multa

En el código:

```python
def devolver_material(self, usuario, material, fecha_devolucion):
    for prestamo in self.prestamos:
        if prestamo.usuario == usuario and prestamo.material == material:
            multa = prestamo.multa_total(fecha_devolucion)
            print(f"Multa a pagar: ${multa}")
            self.prestamos.remove(prestamo)
            material.stock += 1
            print("Material devuelto correctamente.")
            return
    print("No se encontró ese préstamo.")
```

En la ejecución:

```
Usuario: Juan Esteban
Material: La Metamorfosis
Fecha de inicio: 06/08/2025
Días permitidos: 14
¿Cuántos días han pasado desde el préstamo?: 30
¿Retrasado?: Sí
Multa estimada: $4800
```

📊 Reporte de préstamos activos

En el código:

```python
class Biblioteca:
    def __init__(self):
        self.materiales = []
        self.usuarios = []
        self.prestamos = []

    def agregar_material(self, material):
        self.materiales.append(material)

    def agregar_usuario(self, usuario):
        for u in self.usuarios:
            if u.documento == usuario.documento:
                print("Documento ya registrado.")
                return
        self.usuarios.append(usuario)

    def crear_prestamo(self, usuario, material, fecha_inicio):
        for p in self.prestamos:
            if p.usuario == usuario and p.material == material:
                print("Ese usuario ya tiene ese material prestado.")
                return
```

En la ejecución:

```
Usuario: Camila
Material: Ford
Fecha de inicio: 01/09/2025
Días permitidos: 7
¿Cuántos días han pasado desde el préstamo?: 5
¿Retrasado?: No
Multa estimada: $0

Usuario: Erick
Material: Hobbit
Fecha de inicio: 04/09/2025
Días permitidos: 14
¿Cuántos días han pasado desde el préstamo?: 2
¿Retrasado?: No
Multa estimada: $0
```

Casos de prueba incluidos

- **Juan Esteban** pidió el libro "*La Metamorfosis*" hace un mes → genera multa.
- **Camila** pidió la revista "*Ford*" hace 5 días → sin multa.

- **Erick** pidió el libro "*Hobbit*" → préstamo válido.

▶ Cómo ejecutar el sistema

Se recomienda abrir el ejecutable app.py en visual estudio code o Google Colab.

1. Asegúrate de tener Python 3 instalado.
2. Ejecuta el archivo en consola:

```
python biblioteca.py
```