Parcial (Parte 2) – POO en Python

Biblioteca Digital: Préstamos y Penalizaciones

Objetivo general

Diseñar en **Python** un prototipo funcional de **Biblioteca Digital** que evidencie los 4 pilares de POO: *encapsulación, abstracción, herencia y polimorfismo*, con una interfaz por consola (CLI).

Escenario

La biblioteca administra **materiales** (Libro y Revista), **usuarios** y el ciclo de **préstamo/devolución**. Cada tipo de material tiene plazo y multa diaria propios.

1) Alcance funcional mínimo

- Materiales Crear/listar materiales con `tipo` (book/magazine), `título` y `stock ≥ 0`.
- Usuarios Crear/listar usuarios con `nombre` y `documento (único)`.
- Préstamos Prestar solo si `stock > 0`; calcular *fecha límite* por tipo.
- Devoluciones Devolver, actualizar stock y calcular *penalización* si hay retraso.
- Reportes Listar *préstamos activos* y *vencidos* (con días de atraso y penalización estimada).

2) Reglas de negocio

- Stock **nunca** negativo.
- Un usuario **no** puede tener **dos préstamos activos del mismo material**.
- Plazos y multas:
  - **Libro** → 14 días, **$300/día** de retraso.
  - **Revista** → 7 días, **$200/día** de retraso.
- Penalización = `max(0, días_atraso) × multa/día`.

- Documento de usuario **único**.

3) Requisitos técnicos

- **Abstracción:** clase abstracta `Item(ABC)` con:
  - `loan_days() -> int`
  - `penalty_per_day() -> int`
- **Herencia:** subclases `Book(Item)` y `Magazine(Item)`.
- **Polimorfismo:** cálculo de fecha límite y penalización usando la misma interfaz de `Item` (sin `if` por tipo en la capa de servicio).
- **Encapsulación:** atributos sensibles (p. ej. `_stock`, `_document_id`) con `@property`/`@setter` y validaciones simples.

Estructura sugerida

```
biblioteca_digital/
  README.md
  app.py
  domain/
    items.py          # ABC Item, Book, Magazine
    user.py           # User
    loan.py           # Loan
  services/
    inventory.py      # crear/listar items, stock
    loans.py          # prestar/devolver (usa polimorfismo)
    reports.py        # activos y vencidos
```

Pruebas mínimas

- Usuario con **documento único** (rechazar duplicados).
- Préstamo con **stock suficiente** vs **insuficiente**.
- Devolución **sin atraso** (penalización 0) y **con atraso** (penalización correcta).

- Reporte de **vencidos** mostrando atraso y penalización estimada.

Entregable

- **Código en repositorio** (GitHub):
  - Cada estudiante debe hacer fork del repositorio base entregado por el docente y trabajar allí.
  - Incluir `README.md` con: cómo ejecutar, comandos de ejemplo, explicación breve de cómo se evidencian los **4 pilares**.
- **Video de sustentación** (5–10 min):
  - Explica diseño OO (4 pilares), muestra ejecución CLI y resultados de pruebas.
  - Sube el video (YouTube/Drive con acceso público o institucional).
- **Documento único** (PDF o MD exportado a PDF) con:
  - Descripción breve, capturas de ejecución y de pruebas.
  - **Enlaces** al repositorio y al video.
- **Entrega en Moodle (LMS)**:
  - Subir el documento consolidado y **pegar los enlaces** al **repositorio** y al **video**.

Checklist de verificación

- [ ] `Item` abstracto + `Book`/`Magazine` (herencia).
- [ ] Polimorfismo en cálculo de plazo/multa (sin `if` por tipo en servicios).
- [ ] Encapsulación con `@property`/`@setter` y validaciones.
- [ ] CLI ejecuta altas, préstamos, devoluciones y reportes.
- [ ] Pruebas mínimas.
- [ ] README con instrucciones.
- [ ] Repositorio fork público/compartido.
- [ ] Video de sustentación (5–10 min).
- [ ] Documento final con capturas + enlaces (repo y video) subido a Moodle.