Tecnológico Nacional de México campus Tuxtla Gutiérrez

Carrera: Ingeniería en sistemas computacionales

TOPICOS AVANZADOS

Alumna:

Paola Montserrat Cruz Huet

Grado y grupo:

S4C

Sistema de Gestión de Biblioteca: Jerarquía de Clases

Este documento detalla la implementación de un sistema de gestión de biblioteca utilizando una jerarquía de clases en Python. El diseño incluye herencia y polimorfismo para manejar diferentes tipos de elementos bibliográficos y usuarios.

---

1. Definición de Clases

Clase `Libro`

La clase `Libro` es la base para todos los elementos que pueden ser gestionados en la biblioteca.

Python
```
class Libro:
    """
    Representa un libro en la biblioteca.
    """
    def __init__(self, titulo, autor, isbn):
        self.titulo = titulo
        self.autor = autor
        self.isbn = isbn
        self.disponible = True # Atributo para controlar la
disponibilidad del libro

    def mostrar_informacion(self):
        """
        Muestra la información básica del libro.
        """
        estado = "Disponible" if self.disponible else "Prestado"
        return f"Título: {self.titulo}, Autor: {self.autor}, ISBN:
{self.isbn}, Estado: {estado}"
```

Atributos:

- `titulo` (str): El título del libro.
- `autor` (str): El autor del libro.
- `isbn` (str): El ISBN (International Standard Book Number) del libro.
- `disponible` (bool): Indica si el libro está disponible para préstamo (`True`) o si está prestado (`False`).

Métodos:

- `mostrar_informacion()`: Devuelve una cadena de texto con el título, autor, ISBN y el estado de disponibilidad del libro.

Clase `Revista`

La clase `Revista` hereda de `Libro`, lo que significa que una revista es un tipo especializado de libro.

Python
```
class Revista(Libro):
    """
    Representa una revista en la biblioteca, hereda de Libro.
    """
    def __init__(self, titulo, autor, isbn, numero_edicion):
        super().__init__(titulo, autor, isbn) # Llama al constructor de la clase base (Libro)
        self.numero_edicion = numero_edicion # Atributo adicional para revistas

    def mostrar_informacion(self):
        """
        Muestra la información de la revista, incluyendo el número de edición.
        Sobrescribe el método de la clase padre.
        """
        info_libro = super().mostrar_informacion() # Obtiene la información del libro
        return f"{info_libro}, Número de Edición: {self.numero_edicion}"
```

Herencia:

- Hereda de `Libro`.

Atributos adicionales:

- `numero_edicion` (str): El número de edición de la revista.

Sobrescritura de Método:

- `mostrar_informacion()`: Este método se sobrescribe para incluir el `numero_edicion` en la información mostrada, además de la información básica del libro.

Clase `Usuario`

La clase `Usuario` representa a las personas que interactúan con la biblioteca.

Python
```
class Usuario:
    """
    Representa un usuario de la biblioteca.
    """
    def __init__(self, nombre, tipo_usuario):
        self.nombre = nombre
        self.tipo_usuario = tipo_usuario  # Puede ser "Estudiante" o
"Profesor"
        self.libros_prestados = [] # Lista para almacenar los libros que
el usuario tiene prestados

    def prestar_libro(self, libro):
        """
        Método polimórfico para prestar un libro.
        El comportamiento varía según el tipo de usuario.
        """
        if not libro.disponible:
            print(f"Lo siento, el libro '{libro.titulo}' no está
disponible en este momento.")
            return

        if self.tipo_usuario == "Estudiante":
            if len(self.libros_prestados) >= 3:
                print(f"Estudiante {self.nombre}, has alcanzado el límite
de 3 libros prestados.")
            else:
                self._realizar_prestamo(libro) # Llama al método interno
para realizar el préstamo
        elif self.tipo_usuario == "Profesor":
            if len(self.libros_prestados) >= 5:
                print(f"Profesor {self.nombre}, has alcanzado el límite
de 5 libros prestados.")
            else:
                self._realizar_prestamo(libro) # Llama al método interno
para realizar el préstamo
        else:
            print("Tipo de usuario no reconocido.")

    def _realizar_prestamo(self, libro):
        """
        Método interno para realizar el préstamo de un libro.
        """
        libro.disponible = False
        self.libros_prestados.append(libro)
        print(f"{self.nombre} ha prestado el libro: '{libro.titulo}'.")

    def devolver_libro(self, libro):
        """
        Devuelve un libro a la biblioteca.
        """
        if libro in self.libros_prestados:
            libro.disponible = True
            self.libros_prestados.remove(libro)
```

```
            print(f"{self.nombre} ha devuelto el libro:
'{libro.titulo}'.")
        else:
            print(f"{self.nombre} no tiene prestado el libro:
'{libro.titulo}'.")

    def mostrar_libros_prestados(self):
        """
        Muestra los libros que el usuario tiene prestados.
        """
        if not self.libros_prestados:
            print(f"{self.nombre} no tiene libros prestados.")
            return

        print(f"Libros prestados a {self.nombre}:")
        for libro in self.libros_prestados:
            print(f"- {libro.titulo}")
```

Atributos:

- `nombre` (str): El nombre del usuario.
- `tipo_usuario` (str): El tipo de usuario, que puede ser "Estudiante" o "Profesor".
- `libros_prestados` (list): Una lista que almacena los objetos `Libro` que el usuario tiene actualmente en préstamo.

Métodos:

- `prestar_libro(libro)`: Este método exhibe **polimorfismo**. Su comportamiento varía según el `tipo_usuario`. Los estudiantes pueden prestar un máximo de 3 libros, mientras que los profesores pueden prestar un máximo de 5. También verifica si el libro está disponible.
- `_realizar_prestamo(libro)`: Un método auxiliar privado (indicado por el prefijo _) que gestiona la lógica real del préstamo: marca el libro como no disponible y lo agrega a la lista `libros_prestados` del usuario.
- `devolver_libro(libro)`: Permite al usuario devolver un libro, actualizando su estado a disponible y eliminándolo de la lista de préstamos del usuario.
- `mostrar_libros_prestados()`: Imprime una lista de todos los libros que el usuario tiene prestados.

---

2. Ejemplos de Uso

A continuación, se muestran ejemplos de cómo interactuar con las clases definidas, demostrando la creación de objetos, el préstamo y la devolución de libros, y cómo el polimorfismo afecta el comportamiento de `prestar_libro`.

Python

```
# --- Ejemplos de uso ---
print("--- Creación de objetos ---")
libro1 = Libro("Cien años de soledad", "Gabriel García Márquez", "978-
0307474728")
revista1 = Revista("National Geographic", "Varios", "978-1234567890",
"250")

usuario_estudiante = Usuario("Ana", "Estudiante")
usuario_profesor = Usuario("Pedro", "Profesor")

print("\n--- Información de los elementos ---")
print(libro1.mostrar_informacion())
print(revista1.mostrar_informacion())

print("\n--- Préstamos de libros (Polimorfismo) ---")
usuario_estudiante.prestar_libro(libro1)
usuario_estudiante.mostrar_libros_prestados()
print(libro1.mostrar_informacion())

print("\n--- Intentando prestar más libros a un estudiante ---")
libro2 = Libro("Don Quijote de la Mancha", "Miguel de Cervantes", "978-
8420412148")
libro3 = Libro("1984", "George Orwell", "978-0451524935")
libro4 = Libro("El Principito", "Antoine de Saint-Exupéry", "978-
0156013915")

usuario_estudiante.prestar_libro(libro2)
usuario_estudiante.prestar_libro(libro3)
usuario_estudiante.prestar_libro(libro4) # Este préstamo debería fallar
por el límite del estudiante
usuario_estudiante.mostrar_libros_prestados()

print("\n--- Préstamos de libros a un profesor ---")
usuario_profesor.prestar_libro(libro1) # Libro1 ya está prestado por Ana,
por lo que este préstamo fallará
usuario_profesor.prestar_libro(libro4)
usuario_profesor.mostrar_libros_prestados()
print(libro4.mostrar_informacion())

print("\n--- Devolución de libros ---")
usuario_estudiante.devolver_libro(libro1)
usuario_estudiante.mostrar_libros_prestados()
print(libro1.mostrar_informacion())

usuario_profesor.devolver_libro(libro4)
usuario_profesor.mostrar_libros_prestados()
print(libro4.mostrar_informacion())
```