Ejercicio Evaluación Continua MF0492_3 - UF1844

Sistema Avanzado de Gestión de Biblioteca

Instrucciones:

Se debe crear un sistema para gestionar una Biblioteca. El sistema debe gestionar libros, autores, usuarios y miembros premium. Cada libro tiene un título, un autor y un indicador de disponibilidad, además hay una lista genérica de libros al que se añade cada libro en cuanto se crea. Los autores tienen un nombre y una lista de libros que han escrito, y son los únicos que pueden publicar los libros (cuando el autor publica el libro lo está creando). Los usuarios tienen un nombre y una lista de libros en préstamo. Los miembros premium, que heredan de usuarios, tienen la capacidad adicional de reservar libros que están actualmente prestados.

Especificaciones:

1. Clase Libro

Atributos:

- título (string)
- autor (referencia a Autor)
- disponible (booleano)
- lista_libros (lista de Libro)

Métodos:

- marcar_prestado(): Cambia el estado de `disponible` a `False`.
- marcar_devuelto(): Cambia el estado de `disponible` a `True`.

2. Clase Autor

Atributos:
- nombre (string)
- libros (lista de Libro)

Métodos:
- publicar_libro(libro): Añade un libro a la lista de libros del autor.

3. Clase Usuario

Atributos:
- nombre (string)
- libros_prestados (lista de Libro)

Métodos:
- tomar_prestado(nombre_libro): Si el libro está disponible, lo añade a `libros_prestados` y cambia su disponibilidad.
- devolver_libro(nombre_libro): Remueve un libro de `libros_prestados` y cambia su disponibilidad.

4. Clase MiembroPremium (hereda de Usuario)

Atributos:
- libros_reservados (lista de Libro)

Métodos adicionales:
- reservar_libro(libro): Si el libro no está disponible, lo añade a una lista de reservas. Si está disponible, lo añade a su lista de libros prestados y cambia su disponibilidad.

***Recuerda añadir los métodos necesarios de get y set para cada clase.**

Análisis de Clases y Relaciones

Relaciones Libro
- Pertenece a un Autor: Cardinalidad 1 (cada Libro tiene exactamente un Autor).
- Un Libro puede estar o no en préstamo a un Usuario o MiembroPremium.

Relacions Autor

- Tiene múltiples Libros: Cardinalidad 1 a muchos (un Autor puede escribir y publicar varios libros).

Relaciones Usuario

- Puede tener préstamos de múltiples Libros: Cardinalidad 0 a muchos (un Usuario puede tener varios libros en préstamo al mismo tiempo).

Relaciones MiembroPremium

- Hereda las propiedades y métodos de Usuario.
- Puede reservar libros que están en préstamo: Cardinalidad 0 a muchos (puede tener reservas de libros no disponibles).

Indicaciones para el Main / Salida por Terminal

1. Instancias a Crear:

- Al menos 2 Autores.
- Al menos 3 Libros publicados por los autores.
- Al menos 1 Usuario y 1 MiembroPremium.

2. Relaciones/Acciones a Demostrar:

- Autor publica libros.
- Usuario toma prestado y devuelve al menos un libro.
- MiembroPremium reserva al menos un libro no disponible.
- MiembroPremium va a reservar un libro que se encuentra disponible y por lo tanto lo toma prestado.
- Mostrar cada vez que un libro está disponible o deja de estarlo.

*** Estas son las relaciones mínimas a demostrar. Tienen que aparecer los mensajes en la terminal que demuestran estas relaciones/acciones. Puedes crear tantas instancias de las clases y muestras de relaciones como quieras.**

**** Puedes crear más atributos, métodos y listas si lo crees conveniente; pero no es necesario.**

***** Recuerda usar las mejoras prácticas siempre. El código debe estar comentado correctamente.**

Rúbrica de Evaluación:

Criterios de Evaluación	Excelente (10 puntos)	Bueno (7-9 puntos)	Satisfactorio (4-6 puntos)	Insuficiente (1-3 puntos)	No realizado (0 puntos)
Creación de Clases y Métodos **Ponderación: 30%**	Todas las clases y métodos están correctamente implementados y son totalmente funcionales. Los métodos y clases adicionales propuestos son adecuados y mejoran la funcionalidad.	Las clases y métodos principales están implementados correctamente con mínimos errores.	Las clases y métodos están implementados, pero con errores significativos o falta de algunos métodos clave.	Las clases y métodos están implementados de manera incompleta o incorrecta.	No se crearon clases ni métodos.
Estructuras de Datos y Control de Flujo (String, boolean, int, if, for, while..) **Ponderación: 30%**	Las estructuras de datos son óptimas y el control de flujo es completamente eficiente y efectivo.	Las estructuras de datos y el control de flujo son adecuados con pequeños errores o ineficiencias.	Las estructuras de datos y el control de flujo son implementados pero no de manera óptima, presentan errores notables.	Estructuras de datos y control de flujo inadecuados para la funcionalidad requerida.	No se implementaron estructuras de datos ni control de flujo.

Documentación del Código **Ponderación: 10%**	Documentación completa y detallada de todas las clases, métodos y estructuras de datos utilizadas.	Documentación buena pero incompleta en algunas partes del código.	Documentación básica que omite detalles importantes para la comprensión del código.	Mínima o ninguna documentación, dificultando la comprensión del código.	No hay documentación del código.
Depuración y Verificación de Componentes (Main/Salida por Terminal) **Ponderación: 30%**	Todos los componentes son depurados y verificados con casos de prueba exhaustivos, demostrando total funcionalidad sin errores.	Los componentes son mayoritariamente correctos, con depuración adecuada y pocos errores residuales.	Depuración básica realizada, con varios errores funcionales no corregidos.	Escasa depuración y verificación, con errores graves que afectan la funcionalidad.	No se realizó ninguna depuración ni verificación.