Materia:	Programación II		
Nivel:	2º Cuatrimestre		
Tipo de Examen:	Modelo de Primer Parcial		
Apellido(1):		Fecha:	
Nombre/s(1):		Docente a cargo(2):	
División(1):		Nota(2):	
DNI(1):		Firma(2):	

(1) Campos a completar solo por el estudiante en caso de imprimir este enunciado en papel.
(2)Campos a completar solo por el docente en caso de imprimir este enunciado en papel.

Sistema Biblioteca

Nos encargan implementar un sistema básico para una biblioteca.
En la biblioteca se administran diferentes tipos de publicaciones como libros, revistas e ilustraciones. Cada publicación cuenta con un titulo y el año en que se publicó. Los libros además tienen un autor y pertenecen a un género (**FICCION**, **NO_FICCION**, **CIENCIA**, **HISTORIA).**
Las revistas cuentan con un número de edición y las ilustraciones con el nombre del ilustrador y las dimensiones ancho y alto.
Tanto los libros como las revistas deben poder leerse llamando a su método leer.

- **agregarPublicacion()** : El empleado de la biblioteca debe poder agregar todo tipo de publicaciones a la misma. Se deberá lanzar una advertencia personalizada si ya existe la publicación. (dos publicaciones son iguales si tienen el mismo titulo y año de publicación)

- **mostrarPublicaciones()** : Muestra todas las publicaciones donde se pueden observar los siguientes atributos.(dedúzcalos de la imagen de ejemplo):

```
Libro [autor=Homero, genero=FICCION]
Ilustracion [ilustrador=Quinquela Martin, ancho=125, alto=105]
Revista [numeroEdicion=345]
```

-leerPublicaciones() : Leer todas las publicaciones "leibles". Informar las que no se puedan leer.
A partir del enunciado anterior se solicita:

- Realizar el diagrama de clases completo en umletino.
- Realizar el código fuente en Java que resuelva las funcionalidades solicitadas.

Objetivos de Aprobación No Directa (Calificación de 4 a 5 puntos):

1. **Diagrama de clases**:
   - El diagrama de clases debe reflejar correctamente las relaciones entre las clases
   - Se deben mostrar atributos como el título y el año en la clase Publicacion, y los atributos específicos de cada subclase.
   - Debe incluir la herencia correctamente.
   - Los métodos mencionados en el enunciado (leer, agregarPublicacion, etc.) deben estar representados.
2. **Clases y Herencia**:
   - El código debe incluir la clase base y las subclases.
   - Cada clase debe tener los atributos solicitados.
   - El uso de herencia debe estar implementado correctamente.
3. **Métodos básicos**:
   - El método agregarPublicacion() debe agregar publicaciones a la biblioteca y lanzar una advertencia si ya existe una con el mismo título y año.
   - El método mostrarPublicaciones() debe mostrar todas las publicaciones con sus atributos.
   - El método leerPublicaciones() debe permitir leer los libros y las revistas, e informar que las ilustraciones no son leíbles.
4. **Funcionamiento general**:
   - El programa debe compilar y ejecutarse sin errores.
   - Las funciones principales deben estar correctamente implementadas y verificables a través de ejemplos simples.

Objetivos de Aprobación Directa (Calificación de G a 10 puntos):

Diagrama de clases detallado:

- El diagrama de clases no solo debe mostrar las relaciones entre clases y métodos, sino que también debe incluir relaciones como composiciones o agregaciones (si aplica).
- Deben incluirse visibilidades correctas para atributos y métodos (privado, público, protegido).
- El diagrama debe demostrar un correcto entendimiento de la arquitectura del sistema.

Implementación completa y estructurada:

- El código debe utilizar correctamente los principios de POO, como encapsulamiento y uso de modificadores de acceso adecuados.
- Deben existir comentarios claros y descriptivos en el código, explicando las partes más complejas.
- El código debe incluir métodos adicionales útiles (como un método `toString()` para cada tipo de publicación).

Métodos y funcionalidades extras:

- Se espera que el método `leerPublicaciones()` no solo lea, sino que maneje de manera elegante qué publicaciones pueden leerse y cuáles no, con reportes claros.

Eficiencia y uso de colecciones adecuadas:

- El programa debe usar colecciones.

Funcionalidad completa y ejemplos:

- El código debe incluir un conjunto de pruebas o un menú interactivo que permita verificar el correcto funcionamiento del sistema.
- Todo debe funcionar de manera fluida sin errores y debe cubrir todos los casos solicitados en el enunciado, incluyendo la lectura y la validación de publicaciones duplicadas.