Práctica 7-8

Objetivos:

Practicar el manejo de ficheros y excepciones personalizadas y estándar, además de seguir utilizando herencia y polimorfismo.

Aplicación:

En una biblioteca digital se desea gestionar de forma eficiente el registro de publicaciones disponibles, permitiendo a los usuarios añadir nuevas publicaciones, consultar el catálogo actual y almacenar o recuperar la información desde un fichero. Este sistema de gestión estará modelado por una clase base `Publicacion` y dos clases derivadas, `Libro` y `Revista`. A continuación, se describen los atributos y métodos de cada una de las clases a implementar.

Clase Base: Publicacion

Atributos (decida cómo deben declararse: privados o protegidos):

- `titulo:` título del libro (cadena de texto).
- `autor:` autor del libro (cadena de texto).
- `anio:` año de la publicación (entero).

Métodos:

- `__init__`: inicializará sus atributos con valores válidos.
- Utilice decoradores para el acceso con el mismo nombre de los atributos.
- `Descripcion():` permitirá visualizar los datos del objeto y se redefinirá en las clases derivadas.

Clase Derivada: Libro

Atributos adicionales (decida cómo deben declararse: privados o protegidos):

- `genero`: género del libro, por ejemplo ciencia ficción, novela, etc (cadena de texto).

Métodos:

- `__init__`: inicializará sus atributos con valores válidos.
- Utilice decoradores para el acceso con el mismo nombre de los atributos.
- `Descripcion()`: muestra los datos del libro.

Clase Derivada: Revista

Atributos adicionales (decida cómo deben declararse: privados o protegidos):

- `num_edicion:` número de la edición (entero).

Métodos:

- `__init__`: inicializará sus atributos con valores válidos.
- Utilice decoradores para el acceso con el mismo nombre de los atributos.
- `Descripcion()`: muestra los datos de la revista.

Para el manejo de errores en la biblioteca debe utilizar excepciones personalizadas. Considere crear una excepción base `ErrorBiblioteca` que herede de `Exception`. Dicha clase contiene un mensaje predeterminado (es decir, en el `__init__`) que puede ser sobrescrito en las clases derivadas. Cree a partir de ella, la clase derivada `ErrorArchivo` para problemas relacionados con ficheros (por ej. archivo no encontrado o error al guardar el fichero). Las dos clases irán en un módulo llamado `excepciones.py`:

El programa principal se escribirá en un fichero `main.py` y contendrá un menú como el siguiente:

```
1. Añadir  publicaciones (libros o revistas).
2. Mostrar publicaciones disponibles.
3. Guardar publicaciones en un fichero.
4. Cargar  publicaciones desde un fichero.
5. Salir.
```

La opción 1 permite al usuario registrar una nueva publicación en el sistema. Si los datos introducidos no son válidos (por ejemplo, el año no es un entero positivo o el título está vacío), el sistema mostrará un mensaje de error.

La opción 2 muestra en pantalla todas las publicaciones registradas en el sistema. Cada publicación se listará con su descripción detallada, indicando si se trata de un libro o una revista, junto con sus atributos.

La opción 3 permite al usuario guardar todas las publicaciones registradas en un fichero. El sistema pedirá al usuario que introduzca el nombre del fichero donde se guardará la información. Si ocurre un error durante el proceso (por ejemplo, no se puede acceder al fichero), se mostrará un mensaje de error.

La opción 4 permite al usuario cargar publicaciones desde un fichero previamente guardado. El sistema solicitará el nombre del fichero a cargar. Si el fichero no existe, está vacío o tiene un formato incorrecto, el sistema mostrará un mensaje de error.
Al cargar los datos, el sistema creará las instancias correspondientes de libros y revistas según la información contenida en el fichero.

La opción 5 terminará el programa.

Extienda las funciones del módulo `Utils.py` de la práctica anterior para el manejo de excepciones cuando los valores estén mal introducidos. Añada a este fichero las funciones para guardar y cargar las publicaciones.