Examen Parcial I: *Biblioteca*

Realiza una aplicación demostrativa para validar el diseño de algunas clases que se utilizarán en el desarrollo de una aplicación para la administración de la operación de una biblioteca. La implementación de dicha aplicación demostrativa deberá estar basada en los siguientes puntos de diseño:

- ▷ **(5 pts.)** Todas las clases por implementar estarán localizadas en su propio archivo, bajo el espacio de nombres `MyLibrary`. Se implementarán las clases: `Editorial`, `Book`, `User`, `Borrow`, y `Library`. **Importante:** la implementación de las clases deberá seguir los patrones de diseño planteados en clase.

- ▷ **(5 pts.)** La clase `Editorial` implementará lo siguiente:
  - Propiedades `Id` y `Description`. Se deberán elegir apropiadamente sus tipos así como sus atributos de lectura/escritura, basándose en la información presente en el archivo *editorials.txt* y del uso de la clase en la aplicación.
  - Debe incluir un constructor apropiado.
  - ***Nota:*** *cuando se requiera mostrar la información de la editorial, siempre debe aparecer la descripción y no el* `Id`*.*

- ▷ **(5 pts.)** La clase `Book` implementará lo siguiente:
  - Propiedades `Id`, `Copies`, `Title`, `Author`, `EditorialId`, `Edition` y `Year`. Se deberán elegir correctamente sus tipos así como sus atributos de lectura/escritura, basándose en la información presente en el archivo *books.txt* y del uso de la clase en la aplicación.
  - Debe incluir un constructor apropiado.

- ▷ **(5 pts.)** La clase `User` implementará lo siguiente:
  - Propiedades `Id`, `FirstName` y `LastName`. Se deberán elegir correctamente sus tipos así como sus atributos de lectura/escritura, basándose en la información presente en el archivo *users.txt* y del uso de la clase en la aplicación.
  - Debe incluir un constructor apropiado.

- ▷ **(5 pts.)** La clase `Borrow` implementará lo siguiente:
  - Propiedades `UserId` y `BookId`. Se deberán elegir correctamente sus tipos así como sus atributos de lectura/escritura, basándose en la información presente en el archivo *borrows.txt* y del uso de la clase en la aplicación.
  - Debe incluir un constructor apropiado.

- ▷ **(10 pts.)** La clase `Library` implementará lo siguiente:
  - Un constructor sin parámetros. Debe cargarse en listas la información de las editoriales, libros, usuarios y préstamos, de los archivos correspondientes.
  - **Importante:** Esta clase será la encargada de la administración de las listas, así como de la carga y actualización de los archivos. El diseño y la implementación de esta clase se evaluarán mediante la revisión de la funcionalidad de la aplicación.

▷ Menú principal de la aplicación.

- **Usuarios**. Abre un submenú con las siguientes opciones:
  - **(7.5 pts.) Mostrar**. Mostrar la lista de todos los usuarios ordenados mediante un criterio solicitado. Las opciones ofrecidas serán: (a) ordenar por nombre, (b) ordenar por apellido, o (c) ordenar por `Id`.
  - **(7.5 pts.) Agregar**. Agregar un usuario al sistema.
  - **(12.5 pts.) Eliminar**. Eliminar un usuario del sistema, si y solo si no tiene libros en préstamo. Se solicitará el `Id` del usuario. Si posee libros en préstamo se deberá mostrar una lista con la información de los mismos.
- **Préstamos**. Abre un submenú con las siguientes opciones:
  - **(12.5 pts.) Pendientes**. Se mostrará la información de préstamos pendientes por devolver. La información deberá mostrarse agrupada por usuario, junto con un sublistado de los libros que tiene pendientes por devolver.
  - **(12.5 pts.) Solicitar**. Solicitar préstamo de un libro. Se solicitará el `Id` de usuario y el `Id` del libro. Se debe validar que exista una copia disponible para préstamo.
  - **(12.5 pts.) Devolver**. Realizar la devolución de un libro. Se solicitará el `Id` del libro y se mostrará una lista de los usuarios que actualmente lo tienen en préstamo, para su selección.

Pendientes: Funcionalidad parcialmente implementada.

Prestamos pendientes está parcialemente implementada debido a que al mostrar los prestamos pendientes por usuario, tambien se muestran los usuarios que no tienen prestamos con una lista de libros vacia.
En cuanto al resto de funcionalidades, estas fueran implementadas de manera correcta.

Equipo 8:
Barrera Mendez Jessica Natali
Chan Cortes Yaneli Berenice
Escamilla González Juan Pedro
Gómez Álvarez Pedro

Importante:

1. El programa deberá compilar sin errores; en caso de no hacerlo quedará anulado. El profesor no modificará el código por menor que sea la posible corrección.
2. El programa deberá ejecutarse correctamente sin errores.
3. Se entregará el código completo del proyecto en un archivo comprimido con el nombre completo del alumno.
4. La entrega se hará a través de una memoria USB proporcionada por el profesor. El uso de una memoria USB diferente durante el examen, causará la anulación del mismo.