Programación 2

Examen de teoría (2020 - C3 - 119171)

3 de junio de 2020

Ejercicio 2 (4 puntos)

Instrucciones

- **Duración: 1 hora**. La hora límite para le entrega de este ejercicio serán las **11:45h**
- Al final del ejercicio tendrás que entregar los ficheros siguientes: `Book.h`, `Book.cc`, `Author.h`, `Author.cc`, `Library.h`, `Library.cc`, `Reader.h`, `Reader.cc`, `Lending.h`, `Lending.cc` y `main.cc`. Todos ellos se deberán comprimir en un único fichero llamado `ej2.tgz` que se entregará a través del servidor de prácticas de la forma habitual. Para crear el fichero comprimido debes hacerlo de la siguiente manera:

  Terminal

  ```
  $ tar cvfz ej2.tgz Book.h Book.cc Author.h Author.cc Library.h Library.cc
  Reader.h Reader.cc Lending.h Lending.cc main.cc
  ```

- Ninguno de los ficheros anteriores lo vas a editar desde cero, sino que te los descargarás y los modificarás si es necesario. Tienes que entregar todos los ficheros anteriores independientemente de que los hayas modificado o no.
- La entrega se realizará como en las prácticas, esto es, a través del servidor del DLSI (ubicado en la dirección `http://pracdlsi.dlsi.ua.es`), en el enlace `Programación 2 - Ingeniería Informática`. Puedes realizar varias entregas, aunque sólo se corregirá la última.
- Está permitido durante el examen consultar los materiales de la asignatura, libros, apuntes y/o referencias en Internet. Si el código **no compila**, se restará un punto de la nota final del ejercicio
- Para la resolución de los ejercicios, **solo está permitido el uso de las librerías y funciones vistas en clase de teoría y prácticas**. No es válido aportar soluciones copiadas y pegadas de Internet

Código de conducta

- El examen es un trabajo individual. **Cualquier indicio de copia, comunicación con otros alumnos (p.ej. mediante grupos de WhatsApp) o intervención de terceras personas en su realización será sancionada según la legislación vigente**, con medidas que pueden llevar a la **expulsión** del alumno/a de la titulación

Enunciado

Se trata de ampliar un programa de gestión de una biblioteca digital que ha sido escrito en C++ usando programación orientada a objetos; esta ampliación se va a realizar con el objetivo de incorporarle una nueva funcionalidad. La resolución de este problema la harás en dos fases. En la primera, revisarás el código ya existente para entender cada una de sus funciones. En la segunda, escribirás código para ampliar el programa existente y lo entregarás para su evaluación. A modo orientativo, una recomendación general es dedicar aproximadamente un 30 % del tiempo de realización del problema a la primera fase y el 70 % restante a la segunda, pero puedes adaptar estos tiempos a tu conveniencia.

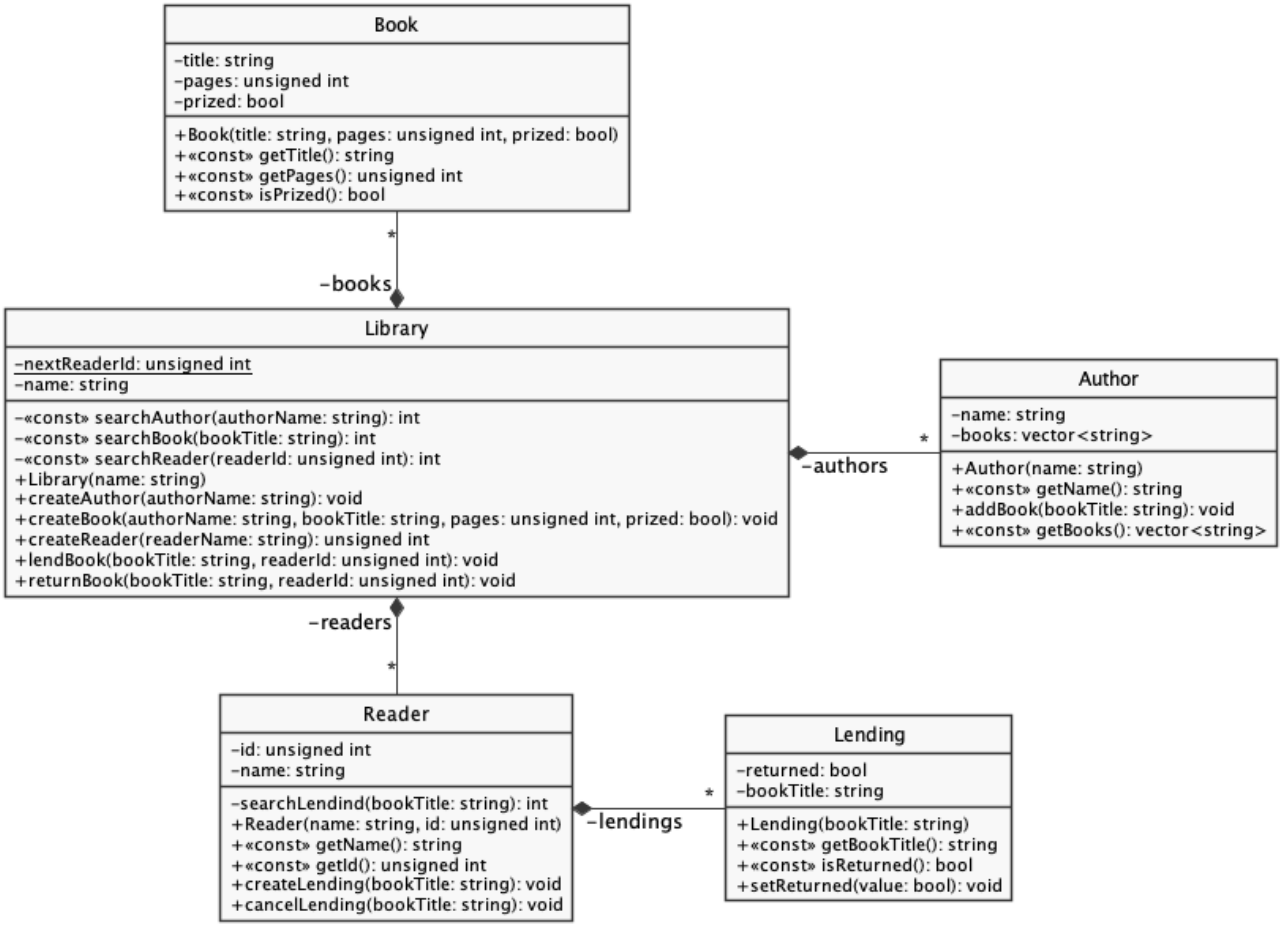

Figura 1: Diagrama de clases UML para el problema 2

Primera fase: entender el código existente

El código del programa, implementado en los ficheros mencionados en las instrucciones, lo puedes encontrar entre los ficheros descargables del examen. Este código implementa el diagrama de clases de la figura 1.

La clase que centraliza la mayor parte de la logística de la aplicación es `Library` (biblioteca). Esta clase tiene métodos para crear un nuevo autor (clase `Author`), un nuevo libro (clase `Book`) y un nuevo lector (clase `Reader`). Cuando se crea un objeto de alguna de estas clases, se añade al vector correspondiente que instrumenta la relación de composición entre `Library` y las otras clases. Cada autor tiene un nombre único que no puede repetirse. Cada libro tiene un título único que no puede tener ningún otro libro (independientemente de quien sea su autor). Un libro puede haber recibido un premio; el método `isPrized()` devuelve `true` si es así, o `false` en caso contrario. Cada lector tiene un identificador único y un nombre que sí puede repetirse en otros lectores.

Un lector puede tomar en préstamo una serie de libros, lo que queda registrado en objetos de la clase `Lending` (préstamo). Asume que un libro con un título concreto puede ser prestado a cualquier cantidad de lectores. Aunque no se refleja en la aplicación actual, a un lector se le cobra por el tiempo que tiene el artículo prestado, y el préstamo puede cancelarse en cualquier momento para que dejen de aplicarse estos gastos. El estado resultante se refleja en el atributo `returned` de la clase `Lending`. Este atributo será `true` si se ha devuelto o `false` en caso contrario.

Junto al código de las clases puedes encontrar un fichero `main.cc` con una función `main` que ejemplifica el uso de las diferentes funcionalidades de la aplicación. Este fichero tiene una función `createData` que crea datos de ejemplo para poder probar el ejercicio.

Segunda fase: programar y entregar una nueva funcionalidad

Añade a la clase `Library` un método público llamado `report` que devuelva el nombre del autor cuyos libros más veces se han sacado en préstamo, independientemente de si han sido devueltos o no; si existe más de un autor que cumpla este criterio, devolverá cualquiera de ellos; si ni un solo libro ha sido sacado en préstamo, devolverá la cadena vacía.

Además de este método puedes añadir los métodos públicos o privados que consideres necesarios en cualquier clase, tanto en la clase `Library` como en las demás.

Añade a la función `main` una llamada a tu método, imprime su resultado y comprueba que es correcto.