Programación 2

Examen de teoría (2020 - C4 - 23221161)

14 de julio de 2020

Ejercicio 2 (4,5 puntos)

Instrucciones

- **Duración: 1 hora y 15 minutos**. La hora límite para la entrega de este ejercicio será las **12:00h**
- Al final del ejercicio tendrás que entregar los ficheros siguientes: `Doctor.h`, `Doctor.cc`, `Hospital.h`, `Hospital.cc`, `Patient.h`, `Patient.cc`, `Record.h`, `Record.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 Doctor.h Doctor.cc Hospital.h Hospital.cc
  Patient.h Patient.cc Record.h Record.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 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 (por ejemplo, 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 hospital 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.

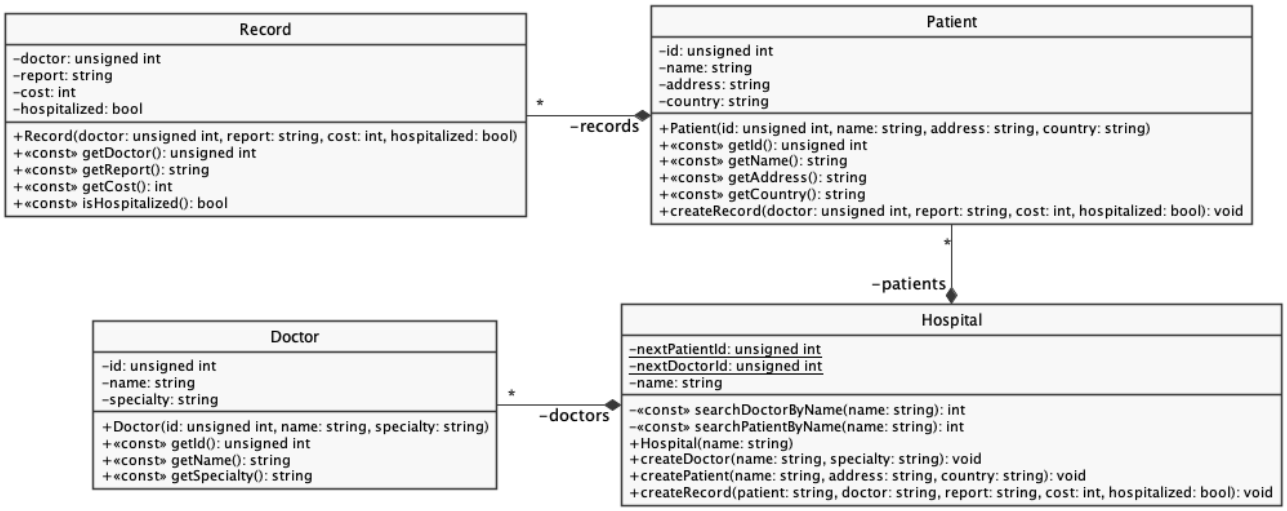

Figura 1: Diagrama de clases UML para el problema 2

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 `Hospital` . Esta clase tiene métodos para crear un nuevo médico (clase `Doctor`), un nuevo paciente (clase `Patient`) y un nuevo registro (clase `Record`). Cuando se crea un objeto de las clases `Doctor` o `Patient`, se añade al vector que instrumenta la relación de composición entre `Hospital` y la clase correspondiente. Cuando a través de `Hospital` se intenta crear un nuevo registro a través del método `createRecord`, este delega la tarea en el método del mismo nombre de la clase `Patient`. Cada médico y cada paciente tienen un identificador único que no puede repetirse en otros objetos de la clase. Los médicos tienen, además, un nombre y una especialidad. Los pacientes tienen, además, un nombre, una dirección y un país de residencia. Por último, los registros contienen una descripción, un coste, una variable booleana que indica si el paciente ha sido hospitalizado y el identificador del doctor implicado.

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 `Hospital` un método público llamado `report` que, dado un país, devuelva un vector (sin duplicados) con el nombre de todos los médicos que han tratado a pacientes que no sean de ese país.

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

Como parte de la solución, deberás añadir a la función `main` una llamada a tu método, imprimiendo su resultado por pantalla para comprobar que es correcto.