Programación 2

Examen de teoría (2021 - C3)

4 de junio de 2021

Ejercicio 2 (5 puntos)

Instrucciones

- **Duración: 1 hora y 25 minutos**. La hora límite para le entrega de este ejercicio serán las **12:00**
- Al final del ejercicio tendrás que entregar los ficheros siguientes: `Calendar.h`, `Calendar.cc`, `Event.h`, `Event.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 Calendar.h Calendar.cc Event.h Event.cc main.cc
```

- **Pon tu DNI y tu nombre en un comentario al principio de cada fichero.**
- 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 (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

Hartos de ser esclavos pasivos de Google, decidimos crear nuestra propia aplicación de gestión de eventos y poder organizarlos en distintos calendarios. Como prueba de concepto, decidimos implementar el diagrama de clases que se muestra en la Figura 1.

A continuación se describe la funcionalidad requerida de cada una de las clases implicadas en el diseño.

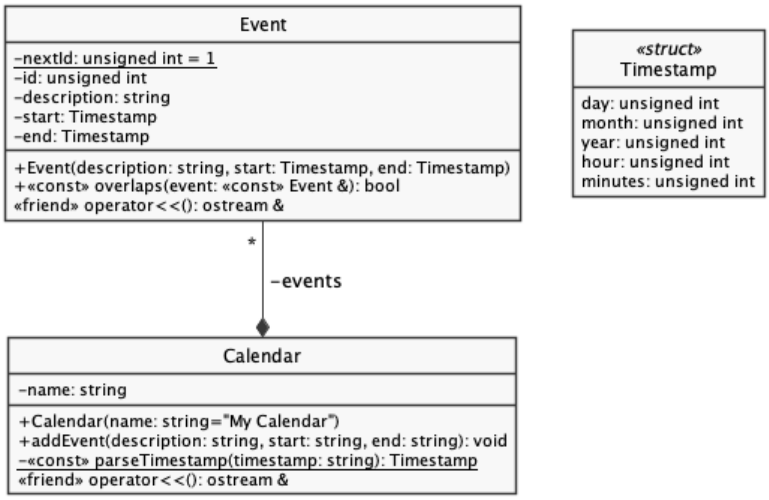

Figura 1: Diagrama de clases UML para el problema 2.

Clase Event

Cada evento está definido por un identificador único (*id*), una descripción (*description*), una fecha de comienzo (*start*) y una fecha de fin (*end*). Estos dos últimos valores se definen, a su vez, mediante un struct *Timestamp* que modela una hora concreta de un día determinado. El struct Timestamp deberá estar declarado en el fichero Event.h.

IMPORTANTE: Por simplicidad, se asume que un evento no puede ocupar días distintos (es decir, el evento finalizaría siempre el mismo día que comienza) y no hace falta comprobarlo en ningún caso. También se asume que todos los eventos comenzarán y terminarán en horas exactas, es decir, los minutos siempre serán cero (p.ej. 9:00). Esta información te puede ayudar a simplificar parte de la lógica de la clase.

- El **constructor** de la clase recibe una cadena con la descripción, junto a dos registro de tipo *Timestamp* que representan el inicio y el fin del evento, respectivamente. **Si la descripción está vacía el constructor lanzará una excepción** para evitar que se cree el evento. No es necesario comprobar que las fechas introducidas contienen valores correctos. El campo *id* debe rellenarse con el valor actual del atributo estático *nextId*. Tras ello, *nextId* debe incrementarse en una unidad.
- El método **overlaps** recibe otro evento como parámetro y debe comprobar si los dos eventos se solapan en el tiempo; es decir, están programados para la misma fecha y uno de ellos comienza antes de que termine el anterior. No se considerará solapamiento si un evento comienza exactamente a la misma hora que termina el anterior. El método devolverá **true** si hay solapamiento y **false** en caso contrario. Por ejemplo, asumiendo que los siguientes eventos suceden en la misma fecha:

  Evento 1: de 13:00 a 15:00, Evento 2: de 14:00 a 16:00 → **true**

  Evento 1: de 14:00 a 16:00, Evento 2: de 13:00 a 15:00 → **true**

  Evento 1: de 13:00 a 14:00, Evento 2: de 14:00 a 16:00 → **false**

- La clase debe sobrescribir el **operador de salida** mediante una función amiga. El formato de salida de un evento es como sigue: "**[id] Dia *dd-mm-yyyy*, de *hh:mm* a *hh:mm* (Duracion: *minutos* minutos): *descripcion.***", en una única línea. Por ejemplo:

  [1] Día 31-5-2021, de 9:00 a 10:00 (Duración: 60 minutos): Estudiar para el examen de P2.

  Nótese en este ejemplo que puede ser necesario añadir un '0' a los minutos para que el formato sea correcto. En el resto de valores no es necesario hacerlo.

Clase Calendar

Esta clase modela un calendario, con un nombre (*name*) y una serie de eventos asociados. Su funcionalidad es la siguiente:

- El **constructor** recibe una cadena que indica el nombre del calendario. Este parámetro tendrá el valor por defecto "My Calendar".
- El método **addEvent** debe crear un objeto de tipo Event y añadirlo a su vector de eventos. Este método recibe, además de la descripción del evento a crear, dos cadenas indicando las fechas de inicio y fin. Las fechas tienen el formato **dd-mm-yyyy hh:mm**. Se asume que la fecha y hora pasada por parámetro siempre es correcta, y no hace falta comprobar que así sea. No obstante, sí es posible que el mes, el día o la hora tengan uno o dos caracteres. Por ejemplo, las siguientes fechas serían igualmente validas: **5-31-2021 09:00** y **05-31-2021 9:00**. Se asume también que la fecha de inicio y fin recibidas serán siempre las mismas, sólo la hora será diferente.

  Si salta una excepción al crear el evento, ésta se deberá capturar y mostrar el mensaje "ERROR: La descripción del evento no puede estar vacía". En este caso no se añadirá el evento al calendario.

  El método **debe comprobar** que este evento no provoque solapamientos en el calendario. En caso de que el nuevo evento solape con otros ya introducidos, éstos deben ser borrados antes de introducir el nuevo.
- El método privado **parseTimestamp** recibe una cadena con el formato descrito para el método addEvent y devuelve un registro de tipo *Timestamp* con los valores correspondientes. Este método es estático.
- La clase debe sobrescribir el **operador de salida** a través de una función amiga. En una primera línea debe indicar el nombre del calendario, seguido por una linea independiente para cada uno de los eventos que contenga que empiece por un guión. Por ejemplo:

  Tareas universidad
  - [1] Día 31-05-2021, de 09:00 a 12:00 (Duración: 180 minutos): Estudiar para el examen de P2.
  - [2] Día 23-06-2021, de 08:00 a 15:00 (Duración: 420 minutos): Estudiar para recuperación de P2.
  - [3] Día 01-09-2021, de 09:00 a 23:00 (Duración: 840 minutos): Matrícula de P2 para el curso 2021-22.

  Los eventos no deben ordenarse de ningún modo sino que se muestran en el mismo orden en el que fueron introducidos en el calendario.

Programa principal y Makefile

Junto con el enunciado has descargado dos ficheros adicionales: `main.cc` y `Makefile`. Puedes usarlos para compilar y probar tu programa.