Programación 2

Examen de teoría (2021 - C4)

8 de julio de 2021

Ejercicio 2 (5 puntos)

Instrucciones

- **Duración: 1 hora y 30 minutos**. La hora límite para la entrega de este ejercicio serán las **15:00**
- Al final del ejercicio tendrás que entregar los ficheros siguientes: `Booking.h`, `Booking.cc`, `Calendar.h`, `Calendar.cc`, `Hotel.h`, `Hotel.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 Booking.h Booking.cc Calendar.h Calendar.cc Hotel.h Hotel.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 `https://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.
- Si lo necesitas, puedes crear más métodos además de los especificados en el diagrama. En este caso deberá tratarse siempre de **métodos privados**.

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

Tras muchos años anotando las reservas en papel y teniendo que estudiar el volumen de clientes a mano cada año, una cadena hotelera nos ha pedido que desarrollemos una aplicación de gestión de reservas de todos sus hoteles. Después de una fase de especificación del problema, diseñamos un diagrama de clases como el que se muestra en la Figura 1.

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

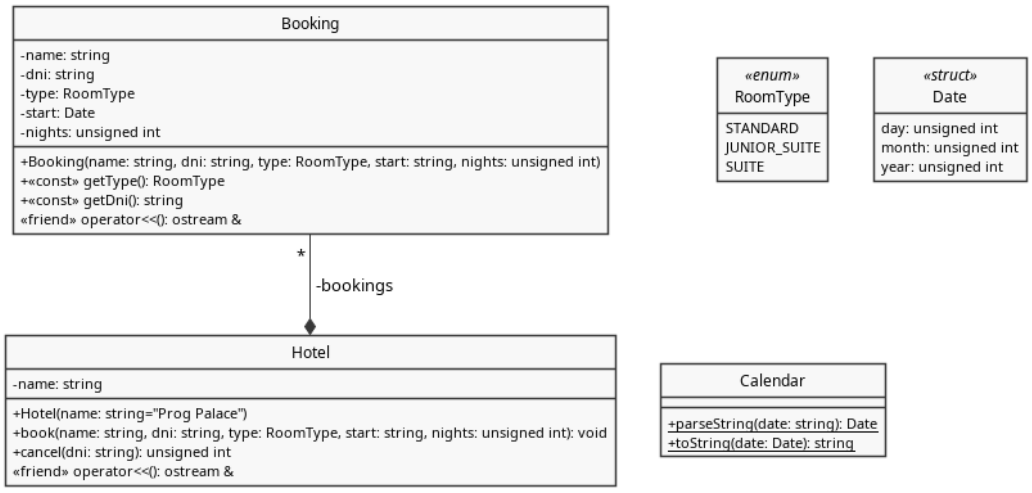

Figura 1: Diagrama de clases UML para el ejercicio 2.

Clase Calendar

Ya que es importante que la aplicación disponga de la fecha de llegada del huésped para cada reserva, tendremos la clase `Calendar`, que tendrá varios métodos que nos facilitarán la gestión de las fechas. Las fechas se representarán a su vez usando el struct `Date`, que se tendrá que declarar en el fichero `Calendar.h`.

La clase `Calendar` contendrá los siguientes métodos estáticos:

- **parseString**: este método recibirá un parámetro de tipo `string` que contendrá una fecha en el formato *DD/MM/AAAA*. No será necesario comprobar que el formato o los datos sean correctos. El objetivo de este método es crear un registro de tipo Date utilizando los datos extraídos del parámetro recibido de tipo `string`. El método deberá aceptar valores de día y mes tanto de uno como de dos dígitos. Por ejemplo, todos los valores siguientes son válidos:

  3/5/2022, 03/5/2022, 3/05/2022, 03/05/2022

  3/12/2022, 03/12/2022, 17/5/2022, 17/05/2022

  17/12/2022

- **toString**: este método recibirá un registro de tipo Date y lo convertirá a un `string` con el formato *DD/MM/AAAA*. En este caso el formato de salida tendrá siempre **dos dígitos** para los campos día y mes. Por ejemplo, si tenemos un registro de tipo `Date` con valores `day` = 1, `month` = 3, `year` = 2021, el `string` generado debería ser "01/03/2021".

Clase Booking

Esta clase representa una reserva y estará definida por un nombre de cliente (`name`), su dni (`dni`), el tipo de habitación reservada (`type`), la fecha de inicio de la reserva (`start`) y el número de noches que el cliente estará alojado en el hotel (`nights`). El tipo de habitación será uno de los tres definidos en el tipo enumerado `RoomType`, que estará declarado en el fichero `Booking.h`.

La clase `Booking` contendrá los siguientes métodos:

- **Constructor**: recibe toda la información de la reserva. La fecha de inicio estará representada mediante un `string` con el mismo formato descrito para el método `parseString` de la clase `Calendar`. Deberá lanzar una excepción si el nombre o dni son la cadena vacía.
- **getType**: devuelve el tipo de habitación.

- **getDni**: devuelve el dni del cliente que realizó la reserva.
- **Operador de salida**: función amiga que mostrará la información de la reserva con el siguiente formato: "DD/MM/AAAA (N nights): Nombre cliente (DNI)", en una única línea, donde la fecha es la fecha de entrada y N es el número de noches. Las fechas usan el formato descrito en el método `toString` de la clase `Calendar`. Por ejemplo:

```
15/07/2021 (7 nights): Hideo Kojima (11111111A)
29/07/2021 (5 nights): Phil Spencer (22222222B)
```

Clase Hotel

La clase `Hotel` representa a un hotel de la cadena hotelera que nos ha pedido desarrollar este proyecto. Esta clase almacenará en un `string` el nombre del hotel (`name`) y contendrá los siguientes métodos:

- **Constructor**: recibirá un parámetro con el nombre del hotel. Dicho parámetro debe configurarse con un valor por defecto, indicado en el diagrama de la Figura 1.
- **book**: recibirá los datos de la reserva (nombre y DNI del huésped, tipo de habitación, fecha de entrada y número de noches). Este método creará una nueva reserva mediante un objeto de tipo `Booking` y la añadirá al vector de reservas. Si al crear la reserva se produce una excepción, deberá capturarse e indicar el error mediante el siguiente mensaje: "ERROR: Los datos de la reserva no son correctos".
- **cancel**: este método recibirá un `string` con un DNI y se encargará de borrar todas aquellas reservas que estén asociadas a ese DNI. Devolverá el número de reservas canceladas o 0 si no se ha encontrado ninguna.
- **Operador de salida**: mostrará el nombre del hotel y, a continuación, todas las reservas agrupadas por tipo de habitación (en el orden `STANDARD`, `JUNIOR_SUITE`, `SUITE`). Si no hay ninguna reserva para alguno de los tipos de habitación **se omitirá su cabecera**. Por ejemplo:

```
Prog Palace
STANDARD
29/07/2021 (5 nights): Phil Spencer (22222222B)
SUITE
15/07/2021 (7 nights): Hideo Kojima (11111111A)
23/08/2021 (3 nights): Shigeru Miyamoto (33333333C)
```

  No es necesario ordenar las reservas por fecha ni ningún otro campo (a excepción del agrupamiento por tipo de habitación) sino que se mostrarán en el mismo orden en que fueron introducidas en el vector.

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.