Diagnóstico Programación Orientada a Objetos Ene-May 2023

A continuación, desarrollarás una parte de un sistema muy básico para la administración de las habitaciones de un hotel. Es importante respetar las convenciones y muy importante que declares correctamente aquello que debe ser **public** y lo que debe ser **private**. Además, respeta los nombres que se indican para cada atributo y método. No es necesario validar nada que no se pida de manera explícita. Es MUY IMPORTANTE que separes la clase en archivos .h y .cpp.

Crea una clase llamada **Habitacion**, que tiene los siguientes atributos:

- ✓ numero: entero //# de habitación
- ✓ nombre: string //del huesped
- ✓ adultos: entero //# en la habitación
- ✓ infantes: entero //# en la habitación
- ✓ credito: double //voucher abierto
- ✓ cargo: double //$ cargos adicionales
- ✓ disponible: bool //libre?

Además tendrá los siguientes métodos:

- ✓ Constructor default: Inicializa el nombre con un string vacío y los atributos numéricos en 0 y disponible en true.
- ✓ Constructor que recibe como parámetro el número de la habitación: Lo único que hace es inicializa el atributo **numero** y marca la habitación como disponible.
- ✓ Getters para atributos numero y disponible. Muy importante respetar la convención de los nombres para funcionaes getters.
- ✓ checkin: recibe el nombre del huésped a registrar en la habitación, el número de adultos y de niños que se alojarán en la habitación y cuánto crédito dejarán abierto para compras. Inicializa los atributos que se reciben como parámetros y además marca la habitación como no disponible.
- ✓ checkout: Valida que la habitación estuviera ocupada. Si estaba ocupada deja el nombre del huésped como un String vacío y los atributos nombre, adultos, infantes, cargos y crédito los establece en 0 y marca la habitación como disponible y el método regresa true. Si la habitación no estaba ocupada no hace nada más que regresar false.
- ✓ getTarifaBase. Este método regresa cuanto se tiene que cobrar por concepto de alojamiento en la habitación. Para calcular este monto simplemente por cada adulto se cobra $450.00 pesos y $150.00 por cada niño.
- ✓ realizarCargo: Este método se utiliza cuando se quiere hacer un cargo adicional a la habitación. Recibe como parámetro la cantidad que se desea cargar. Agrega el cargo al saldo actual siempre y cuando la cantidad que se pase como parámetro sea positiva y que al hacer el cargo no exceda el crédito disponible. Si se pudo hacer el cargo regresa true sino false.
- ✓ Implementa el método toString de manera que regrese un string con la siguiente representación.
  `[# habitacion],Huesped:[nombre],Tarifa Base:[tarifaBase],Credito:[Credito],Cargos:[cargos]`
  Un ejemplo de un resultado podría ser (Observa que no hay espacios junto a las comas):
  `100,Huesped:Gerardo Salinas,Tarifa Base:800.0,Credito:3000.0,Cargos:2500`

Crea otra clase llamada **Hotel**, la cual representa un hotel que se va a administrar con este sistema. El hotel está compuesto por varias habitaciones las cuales se almacenan en un atributo llamado **habitaciones** y además tiene como atributo **numHabitaciones** que representa la cantidad de habitaciones que tiene el hotel y el atributo **nombre** que representa el nombre del hotel.
Además tendrá los siguientes métodos:

- ✓ Constructor que recibe el nombre del hotel y el número de habitaciones que tendrá el hotel. Inicializa el nombre y además inicializa habitaciones con el número exacto de habitaciones que indican en el parámetro. Cada habitación estará numerada **SIEMPRE** a partir del número 100. Es decir la primera habitación será la #100, la siguiente la #101 y así hasta llegar a la última habitación.
- ✓ checkin: Recibe el nombre del huésped a registrar, # de adultos, # de niños y el crédito que la persona quiere dejar abierto. El método registra al huésped en la primer habitación disponible y regresa el # de habitación donde lo registró. En caso de que el hotel estuviera lleno regresa -1. (En este método por favor sí déjenlo que regrese el # de la habitación donde se está alojando la persona y -1 si no hay disponibilidad)
- ✓ checkout: Recibe el # de habitación donde se realiza el checkout. Este método valida que el número de habitación que se recibe como parámetro sea válido (exista). Si fue válido regresa si fue posible hacer el checkout de la habitación, en caso contrario regresa false.
- ✓ realizarCargosHabitacion: Recibe el # de habitación y el cargo a realizar. Intenta hacer el cargo a la habitación y regresa si se hizo el cargo o no siempre y cuando la habitación estuviera ocupada sino regresa false. No necesitas validar que la habitación exista.
- ✓ getTotalXTarifaBase, No recibe parámetros y regresa el total que se recaudará ese día por concepto de alojamiento en todas las habitaciones ocupadas.
- ✓ imprimeOcupacion: Este método imprime primero un encabezado que dice “Ocupacion en [nombre hotel] y en las siguientes líneas la información de sólo aquellas habitaciones ocupadas. Un ejemplo que podría desplegar esta función es:

```
Ocupación en Hotel Holiday POO
100,Huesped:Carlos Alba,Tarifa Base:800,Crédito:3000,Cargos:2500
101,Huesped:María José Gutiérrez,Tarifa Base:700,Crédito:500,Cargos:2500
102,Huesped:Sandra Reyes,Tarifa Base:650,Crédito:1000,Cargos:0
```

Opcionalmente puedes hacer un main para probar tu programa de manera que a través de un menú llames las funciones de la clase Hotel.