



# UNIVERSIDADE DA CORUÑA

## Diseño Software

### Práctica de Diseño (2023-2024)


#### INSTRUCCIONES:

Fecha límite de entrega: 22 de diciembre de 2023 (hasta las 23:59).

- Los problemas se han de resolver aplicando principios y patrones de diseño. No será válida una solución que no los use.
- **Informe:** Para cada problema hay que hacer un informe en el que se incluya:
  - Explicación de los **principios de diseño** usados (en particular los **SOLID**) y dónde en concreto se han usado (nombrar clases específicas de vuestro código).
  - Explicación del **patrón/es de diseño** usados explicando para cada uno:
    - **Breve explicación del patrón elegido** y justificación de su utilización.
    - **Diagrama de clases** en el que se muestren las clases involucradas en el patrón. Es importante señalar el rol que juega cada clase propia en el patrón con anotaciones UML en el propio diagrama.
    - **Diagramas dinámicos** (secuencia, comunicación o estados) que muestren el funcionamiento dinámico de aspectos fundamentales del código. Deberéis decidir qué tipo de diagrama es el más adecuado para cada problema.
- **Código y forma de entrega:**
  - Los ejercicios se entregarán usando *GitHub Classroom*. En concreto en el *assignment* 2324-PD del *classroom* GEI-DS-614G010152324-0P1.
  - Se entregará un proyecto de IntelliJ (ficheros de configuración incluidos) con el nombre del repositorio de *classroom* y dos paquetes (**e1** y **e2**) por cada ejercicio.
  - Deberá incluir pruebas y se deberá comprobar la cobertura de las mismas.
  - La documentación explicando el diseño y los principios y patrones utilizados en ambos problemas se entregará como un fichero PDF dentro de un directorio doc del proyecto IntelliJ IDEA.
- **Evaluación:**
  - Esta práctica corresponde a 1/3 de la nota final de prácticas que consistirá en una evaluación de la memoria y el código según los siguientes criterios.
  - **Calidad de la documentación:** selección del patrón y principios adecuados, explicaciones claras de su uso, calidad y claridad de los diagramas entregados, correspondencia con el código, etc.
  - **Calidad del código:** Aplicación correcta de los patrones y principios, seguimiento correcto de la filosofía orientada a objetos, correspondencia con el diseño, pruebas adecuadas, etc.

## 1. Gestión de hotel

Un pequeño hotel nos encarga un software sencillo para gestionar reservas simples y su servicio de limpieza de habitaciones. Para tal efecto, es necesario implementar los siguientes casos de uso para su personal:

- **Añadir habitación:** registra una nueva habitación en el hotel, siendo necesario especificar el nombre del supervisor encargado de registrarla. El número de habitación (de 1 a N) es asignado automáticamente por el sistema.
- **Reservar habitación:** registra a un cliente en una habitación disponible. Es necesario indicar el número de habitación y el nombre del huésped.
- **Terminar reserva:** indica que el huésped ha cancelado la reserva sin llegar a ocupar la habitación. La habitación queda liberada para volver a ser reservada y no es necesario realizar una limpieza. Es necesario especificar el número de habitación.
- **Limpiar habitación:** indica que una habitación ha sido limpiada. Es necesario especificar el número de habitación y el nombre del personal de limpieza encargado. 
- **Liberar habitación:** indica que una habitación ya no está ocupada y puede procederse con su limpieza. Es necesario especificar el número de habitación.
- **Aprobar limpieza:** indica si el estado de limpieza de una habitación es adecuado o no. Es necesario indicar el número de habitación y el nombre del supervisor.
- **Obtener la lista de habitaciones disponibles:** devuelve la lista las habitaciones disponibles para reserva.
- **Obtener la lista de habitaciones pendientes de limpieza:** devuelve la lista de las habitaciones que todavía no han sido limpiadas.
- **Obtener la lista de habitaciones pendientes de aprobación tras ser limpiadas:** devuelve la lista de las habitaciones limpias pendientes de ser aprobadas por parte de un supervisor.
- **Mostrar la información de las habitaciones del hotel:** muestra el estado completo de todas las habitaciones del hotel (ocupadas, disponibles, limpias, pendientes de limpieza, etc.)

Para cumplir correctamente con el ciclo de trabajo del personal del hotel, el sistema debe cumplir con los siguientes requisitos:

- Una habitación está disponible para reserva siempre y cuando haya sido limpiada y aprobada por un supervisor.
- Añadir una habitación al hotel implica que la misma se encuentra en condiciones de ser reservada.
- En caso de estar ocupada, la habitación no podrá ser limpiada.
- Aunque la limpieza de una habitación haya sido aprobada, el posible revocar dicha aprobación e iniciar de nuevo el procedimiento de limpieza.

A continuación se muestra un ejemplo de la salida que se obtendría al hacer uso de la función encargada de mostrar información del estado de las habitaciones del hotel. Para cada habitación, se muestra el número, si está o no reservada por algún huésped y el estado actual de la misma entre las posibles: ocupada, libre y pendiente de limpieza, limpia y pendiente de aprobación, limpia y aprobada.



```
*****
Hotel UDC-Hills
*****
Room no. 1: Booked by Fulanito. This room was approved by Juanito.
Room no. 2: Booked by Menganita. Room cleaned by Pepe, pending approval.
Room no. 3: Booked by Fulanita. Occupied.
Room no. 4: Free. This room was approved by Juanito.
Room no. 5: Free. Cleaning pending.
*****
```

Se deben contemplar aquellos casos que se consideren más relevantes para no interrumpir la ejecución normal del software, como intentar reservar una habitación no disponible o limpiar una habitación ocupada. Debe tratarse la gestión del hotel como un proceso abierto, es decir, que acepte fácilmente modificaciones en el futuro. Por ejemplo, gestionar habitaciones que estén en mantenimiento/reparación y consultar incidencias. Por tanto, hay que plantear una solución basada en principios y patrones de diseño.

## 2. Incursiones Navales

En un mundo de guerra naval, una flota se embarca en incursiones, enfrentando desafíos en su camino. La flota cuenta con atributos cruciales: ❤️ HP (Puntos de Vida), 🎲 Blindaje, 🚢 Poder de Fuego, 🛩️ Antiaéreos y 🗺️ Línea de Visión.

El paisaje naval (mapa) se representa como un gráfico jerárquico de nodos críticos. A continuación se muestra un mapa de ejemplo:



El sistema abarca tres tipos de nodos:

- **Nodo de Fin:** No tiene hijos y señala el final de una ruta de incursión.
- **Nodos en Ruta Fija:** Guían a la flota en una ruta fija y tienen un solo hijo.
  - **Nodo de Ataque Aéreo:** La flota se encuentra con un ataque aéreo enemigo sorpresa con un Poder Aéreo dado. La flota recibe daño calculado como  $(PoderAereo_{Enemigo} - (2 * Antiaereo_{Flota} + Blindaje_{Flota}))$ .
  - **Nodo de Tormenta Marina:** La flota atraviesa una tormenta con una determinada Fuerza. Si la Línea de Visión de la flota es menor que la Fuerza de la tormenta, pierde 10 HP.
- **Nodos de Bifurcación:** Tienen dos hijos, y el camino de la flota puede variar según el resultado.
  - **Nodo de Batalla:** La flota se enfrenta a una flota enemiga con un cierto HP, Blindaje y Poder de Fuego. Ambas flotas reciben daño calculado  $(PoderdeFuego_{Atacante} - Blindaje_{Defensor})$ . Si los HP enemigos llegan a 0, la flota sigue la ruta izquierda; de lo contrario, sigue la ruta derecha.
  - **Nodo de Avistamiento:** La flota debe avistar un Punto de referencia a una Distancia dada para orientarse correctamente. Si la Línea de Visión de la flota es igual o mayor que la Distancia, la flota sigue la ruta izquierda; de lo contrario, sigue la ruta derecha.

El éxito de la incursión depende de los HP restantes de la flota. Si los HP de la flota llegan a 0 en cualquier nodo dado, la incursión termina en fracaso en dicho nodo. Si la incursión termina en un Nodo Final, la incursión ha sido exitosa.

Tu tarea es implementar una solución que pueda realizar las operaciones requeridas por el almirante:

- **Simular la incursión** de una Flota en un mapa, mostrando el resultado final: éxito o no de la incursión, nodo final de la incursión, y HP final de la flota. Por ejemplo, para unas Flotas A y B con las siguientes características:

Flota A	Flota B
HP 11	HP 1
Blindaje 42	Blindaje 25
Poder de Fuego 47	Poder de Fuego 0
Antiaéreo 0	Antiaéreo 46
Línea de Visión 0	Línea de Visión 28

El resultado en el mapa de ejemplo para la Flota A sería:

```
Sortie Result:
  SUCCESS
  Last Visited Node: H
  Final HP: 1
```

Mientras que para la Flota B sería:

```
Sortie Result:
  FAIL
  Last Visited Node: E
  Final HP: -33
```

- Determinar el **número mínimo de nodos necesarios** para alcanzar un Nodo Final en el mapa. En el caso del mapa de ejemplo, se mostraría lo siguiente:

Smallest Node Count to End: 3

- **Representar en texto** la estructura de la zona de incursión en [Formato Newick](#). En el caso del mapa de ejemplo, se mostraría la siguiente representación:

```
(A WaypointSpotting, (B Battle, (D End), (E AirRaid, (K End))), (C Maelstrom, (F Battle, (H End), (G Maelstrom, (I End))))))
```

Tu solución debe adherirse a patrones y principios de diseño, asegurando la capacidad de incorporar nuevos tipos de nodos o operaciones adicionales en el futuro.