**Objetivos**

1. Aprender a aplicar un patrón estructural a un contexto de diseño específico.
2. Comprender los detalles del patrón "Decorador" en su versión dinámica.
3. Comprender los detalles del patrón "Decorador" en su versión estática.

**Contexto y planteamiento del problema**

Retomando el diseño del juego del laberinto, el Cazador y los monstruos, se requiere para este laboratorio que bajo determinadas condiciones (irrelevantes para este laboratorio) en el juego los monstruos y manadas puedan adquirir nuevos poderes tales como:

1. INVISIBILIDAD: que les permitiría ser invisibles para el Cazador.
2. REPLICABILIDAD: que les permitiría replicarse a su antojo.
3. MEMORIA: que ya se había incorporado en un laboratorio anterior, pero ahora se parte del hecho de que no todos los monstruos y manadas tienen memoria al comienzo y que la ganan en algún momento durante el juego.

Un monstruo o manada podría tener varios de estos poderes, habiéndolos adquirido uno por uno. Estos tres no son los únicos poderes, en el futuro podrían agregarse otros. Además, deben poder ser adquiridos por todos los tipos de monstruos (Wumpus, Cadejos, y los que vengan en el futuro), así como por las manadas correspondientes.

**Características de la solución:**

Se deben generar dos soluciones, una basada en la versión estática del Decorador y la otra basada en la versión dinámica del Decorador. En ambos casos:

1. El diseño debe respetar los principios SOLID.
2. El diseño debe basarse en la aplicación correcta del patrón Decorador .
3. El código debe funcionar correctamente como "prueba de concepto", tal como se ha mostrado en los ejemplos. El controlador debe demostrar que se pueden crear monstruos y manadas, que estos pueden adquirir uno o más poderes en cualquier combinación y que responden a la activación/desactivación de los poderes una vez adquiridos.
4. Incluir el modelo de clases en formato pdf.

**Evaluación**:

| Rubro | Peso |
|---|---|
| Aplicación correcta de principios SOLID (15% por cada versión) | 30% |
| Aplicación correcta de Composición (20% por cada versión) | 40% |
| Funcionamiento correcto del código (10% por cada versión) | 20% |
| Uso correcto de UML en el modelo de clases (5% por cada versión) | 10% |