Metodología de Diseño - Problema 2

Parte 1: Resolviendo problemas de diseño con Patrones

# Objetivo

Aplicar patrones para resolver problemas de diseño

1. Nuestro emprendimiento: un juego de pelea

Cree un nuevo proyecto Eclipse y agregue como fuente el directorio que encontrará en Problema 2/fuentes/ExtremeFighter

## El juego

Extreme Fighter es el prototipo de un juego de pelea (tipo Street Fighter, Mortal Kombat).

***Es un prototipo modesto, no tiene interfaz de usuario y sólo permite probar las acciones de cada luchador.***

Como puede apreciar, el juego permite seleccionar un luchador y elegir qué acciones realizar. El juego, por el momento, sólo tiene sólo 2 luchadores. Las acciones de cada luchador están definidas por los métodos que tiene cada luchador.

Dibujemos un diagrama UML que represente el código, en la pizarra.

¿Qué opina del código? ¿Feo ah? ¿Qué tiene de feo? Comentemos en clases.

Resolvámoslo usando **orientación a objetos.**

## Buscando un inversionista - Round 1

El fabuloso prototipo ha sido mostrado a un potencial inversionista que tiene interés en financiar el juego. Luego de revisarlo, el inversionista dice:

***“Me gustaría que el juego tuviera muchos más personajes, y que la mayoría de ellos tuviera movimientos similares”***

¿Cómo modificaría el diseño actual para asegurarse de satisfacer el requerimiento del cliente?

Conversémoslo en clases y modifiquemos el código, agregando dos luchadores más.

Dibujemos de nuevo el diagrama UML de clases correspondiente.

## Buscando un inversionista - Round 2

Nuestro inversionista está satisfecho con la solución al requerimiento anterior, sin embargo, se pone más exquisito:

***“Me gustaría que no todos los luchadores lanzaran poderes. Algunos buenos luchadores, en vez de lanzar poderes, realizan poderosos agarres, o tienen movimientos rápidos de ataque”.***

¿Cómo abordaría el requerimiento del usuario?

Conversémoslo en clases y codifiquemos de nuevo. Generemos 2 luchadores con cada característica: lanzadores de poderes, agarradores, y ataque rápido.

Dibujemos de nuevo el diagrama UML de clases correspondiente.

## Buscando un inversionista - Round 3

Una vez más, el inversionista está satisfecho con la solución. Sin embargo, para comprometer aún más dinero nos pide el siguiente requerimiento:

***“Me gustaría poder vender DLC’s del juego, en los que se agreguen nuevos comportamientos de luchadores: ya no sólo lanza poderes, agarres y ataque rápidos, quiero que en el futuro se puedan agregar luchadores elásticos (atacan a distancia con su cuerpo), voladores (pueden atacar aire-tierra y aire-aire), y otros que se me puedan ocurrir. Además, me gustaría que en el juego existieran Power-Ups, que en medio del round permitieran que un luchador de Agarres pudiera lanzar poderes, o un Luchador de Ataque Rápido realizara agarres”***

¿Qué pasa con la solución actual desde el punto de vista del diseño? Pensemos en tres criterios:

* Código a modificar: cuántas clases y métodos que hoy funcionan bien tengo que afectar para agregar los nuevos comportamientos de los luchadores.
* Reuso y duplicación de código: ¿estoy reusando el código de forma eficiente?

# 3. Actividad: Aplicando un patrón de diseño

## Definiendo el problema

Podemos conceptualizar el problema con la siguiente frase:

*“Tenemos una clase abstracta que* ***define el comportamiento*** *de sus subclases, pero* ***no todos los comportamientos de la superclase aplican para las subclases*** *de la misma forma, y los* ***comportamientos pueden ser modificados en tiempo de ejecución****”*

¿Este problema es exclusivo de este juego? ¿De los juegos de pelea? ¿?

## Reconociendo un Patrón de Diseño

Cuando tenemos un problema repetitivo, encontramos soluciones genéricas, que nacen de **buenas prácticas de diseño y programación.**

En este caso, aplicaremos el Patrón de Diseño Estrategia (Strategy). Búsquelo en internet como Patrón de DIseño GOF Estrategia. Busque la definición y ejemplos en internet.

Actividad

1.- Aplique el patrón de diseño para generar tres familias de luchadores: LanzaPoderes, Agarradores, AtaqueRapido. Reestructure el código para poder implementar el patrón

2.- Dibuje el diagrama UML que representa la solución. ¿Es similar al del patrón de diseño?

3.- Comente: ¿Se resuelve el problema de modificar código de clases existentes?¿Se resuelve el problema de reuso de código?

***RNL-EGL/20191S***