Patrones de Diseño Aplicados a un Prototipo Videojuego de Lucha

Por : Fabián Vidal Torres
Fecha : 09/12/2022

Problema:

Se requiere implementar 3 nuevos requerimientos al juego Extreme Fighter, cada uno de estos requerimientos está asociado a un tipo de patrón de diseño a aplicar. Existe un requerimiento que implica creación de objetos(creacional) , otro que implica definir una estructura de las entidades involucradas en el código(estructural) y finalmente uno que permite generar comportamientos en los objetos del software(comportamiento).

Solución del problema :

A continuación, se presenta la ficha para cada tipo de requerimiento.

- Requerimiento Creacional

Ficha de requerimiento	
Requerimiento	Crear y definir 3 tipos de luchadores tanques, asesinos y magos, cada uno de estos posee una mejora de atributos en base a su categoría. Estos atributos son: • Vida • Maná • Armadura • Ataque • Poder Los magos hacen mucho daño con la habilidad que poseen (Ataque), además de tener mucho maná. Los tanques tienen una mayor durabilidad y aguante para los combates (Mayor vida). Los Asesinos poseen mucho daño de ataques base, sin embargo, tienen poco aguante (Menor vida).
Problema	La distribución de mejora de atributos por categorías supone una implementación en particular para cada tipo de peleador.
Solución	Crear una abstracción para la creación de tanques, asesinos y magos.
Patrón a aplicar	Factory https://www.dofactory.com/net/abstract-factory-design-pattern

- Requerimiento Estructural

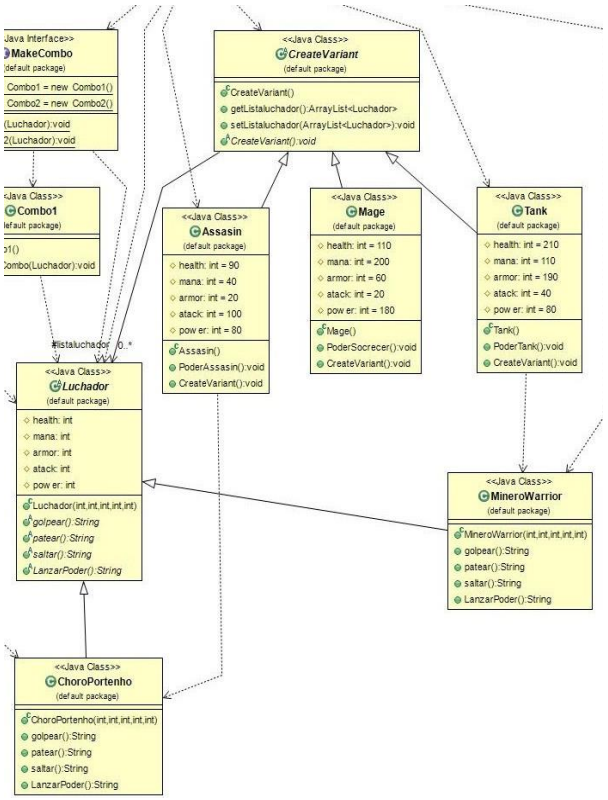
Ficha de requerimiento	
Requerimiento	Los luchadores pueden ejecutar diversos combos con los ataques básicos que poseen, cada luchador tiene un set de combos únicos en base a sus propios ataques.
Problema	La creación de combos supone una estrategia de implementación y creación, para los cuales puedan ser accedidos con cada peleador.
Solución	Crear una interfaz la cual tenga por objetivo agregar a los luchadores los combos.
Patrón a aplicar	Fecade https://www.dofactory.com/net/abstract-factory-design-pattern

- Requerimiento Comportamiento

Ficha de requerimiento	
Requerimiento	Las acciones de ataque de los luchadores pueden variar dependiendo de la postura que tengan, es decir si están de pie ejecutar ataques altos, si están agachados realizar ataques bajos.
Problema	Es preciso definir y ejecutar los métodos de los luchadores con la finalidad de que el tipo de golpes se ejecuten en función de la postura en que se encuentren.
Solución	Crear una interfaz de estados que modifique el comportamiento de los ataques según la postura del luchador (De pie o agachados).
Patrón a aplicar	State https://www.dofactory.com/net/abstract-factory-design-pattern

Diagrama de clases:

- Requerimiento creacional (Factory method)

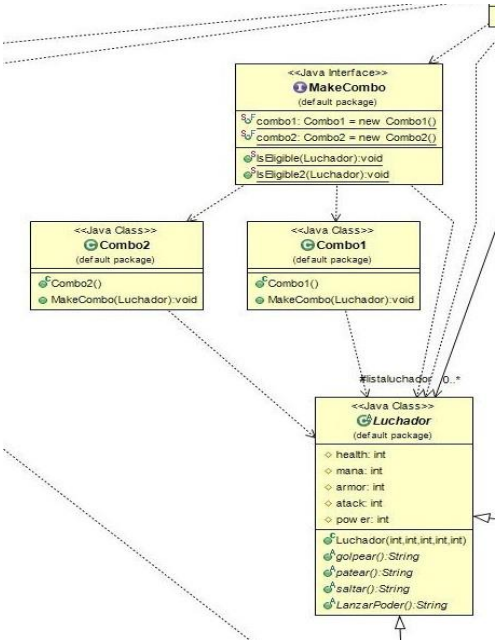


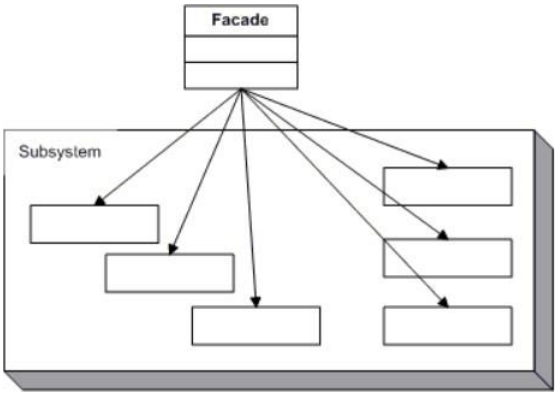
Tal como se muestra en las imágenes anteriores correspondiente a los diagramas de clases, el patrón factory se implementa con las siguientes clases.

- CreateVariant (Creator)
- Luchador (Product)
- Assasin(ConcreteCreator)
- Mage(ConcreteCreator)
- Tank(ConcreteCreator)
- MineroWarrior(ConcreteProduct)
- ChoroPortenho(ConcreteProduct)

Assasin,Mage y Tank heredan los métodos de CreateVariant, estos métodos se caracterizan por agregar un producto concreto a una variante en particular, por ejemplo, se tiene la variante de Tank, esta variante toma un luchador minerowarrior (concrete product) y lo agrega a la variante Tank permitiendo obtener métodos y atributos específicos de la variante Tank.

- Requerimiento estructural (Fecade)



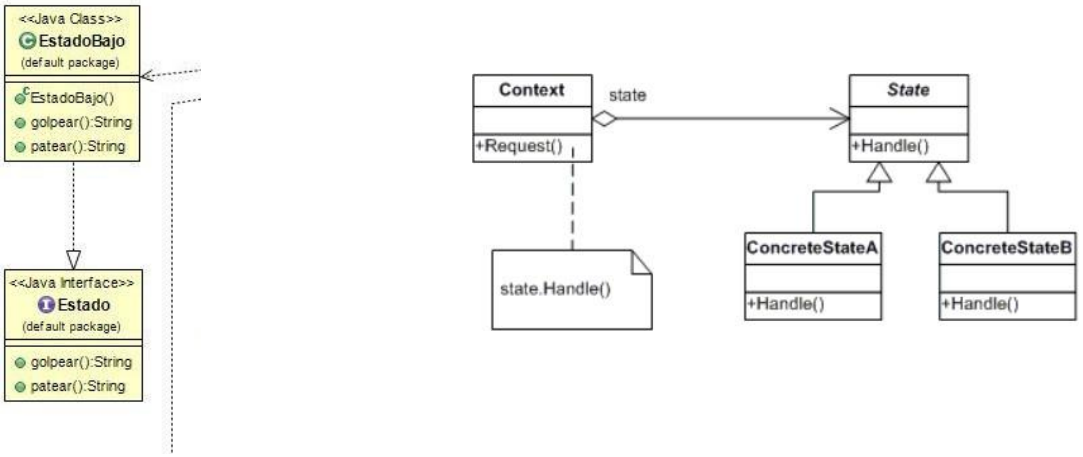


Para el segundo requerimiento se opta por implementar el patrón de diseño Fecade(fachada). Este patrón define una interfaz de nivel superior que controla el uso de un subsistema, para nuestro caso se tiene las siguientes clases.

- MakeCombo(Fecade)
- Luchador (Part of Subsystem)
- Combo 1(Part of Subsystemr)
- Combo 2(Part of Subsystem)

La Interfaz Makecombo toma una clase combox y se lo añade a un luchadorx actuando como una interfaz superior que permite asignar métodos a una clase, de esta manera se pueden definir distintos tipos de combos y estos pueden ser asignados a distintos tipos de luchadores, basta con tener el luchador y el combo que se quiere seleccionar.

- Requerimiento comportamiento (State)

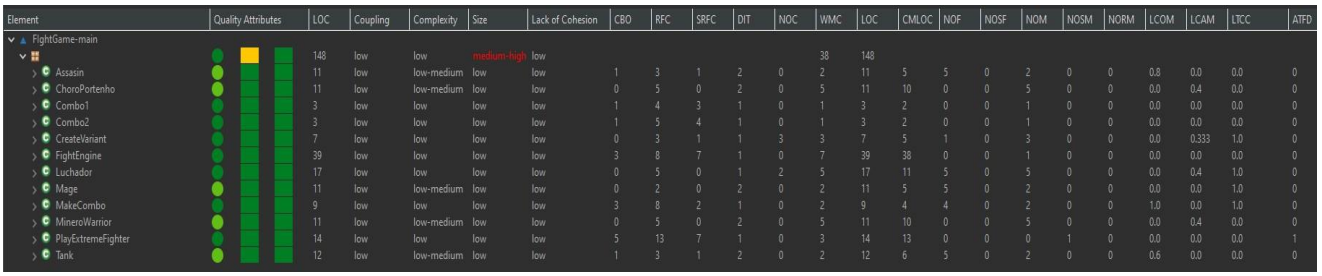


Por último, se presenta el patrón de diseño State para el requerimiento número tres , este patrón de diseño permite que una interfaz altere el comportamiento de un objeto y le asigne un nuevo estado.

- Estado(State)
- EstadoBajo (ConcreteState)

Para este caso en particular EstadoBajo toma los métodos del objeto Estado los cuales son patear y golpear transformándolos para realizar golpe y patada bajos respectivamente. Cabe destacar que el estado por defecto es de pie, este estado se altera permitiendo generar nuevos y distintos estados, para este caso solo se implementó el estado bajo, pero se pueden agregar muchos otros como, por ejemplo, estado volar o estado levitar.

Métricas y conclusión:



Element	Quality Attributes	LOC	Coupling	Complexity	Size	Lack of Cohesion	CBO	RFC	SRFC	DIT	NOC	WMC	LOC	CMLOC	NOF	NOSF	NOM	NOSM	NORM	LCOM	LCAM	LTCC	ATFD
FightGame-main																							
		148	low	low	medium-high	low						38	148										
Assasin		11	low	low-medium	low	low	1	3	1	2	0	2	11	5	5	0	2	0	0	0.8	0.0	0.0	0
ChoroPortenho		11	low	low-medium	low	low	0	5	0	2	0	5	11	10	0	0	5	0	0	0.0	0.4	0.0	0
Combo1		3	low	low	low	low	1	4	3	1	0	1	3	2	0	0	1	0	0	0.0	0.0	0.0	0
Combo2		3	low	low	low	low	1	5	4	1	0	1	3	2	0	0	1	0	0	0.0	0.0	0.0	0
CreateVariant		7	low	low	low	low	0	3	1	1	3	3	7	5	1	0	3	0	0	0.0	0.333	1.0	0
FightEngine		39	low	low	low	low	3	8	7	1	0	7	39	38	0	0	1	0	0	0.0	0.0	0.0	0
Luchador		17	low	low	low	low	0	5	0	1	2	5	17	11	5	0	5	0	0	0.0	0.4	1.0	0
Mage		11	low	low-medium	low	low	0	2	0	2	0	2	11	5	5	0	2	0	0	0.0	0.0	1.0	0
MakeCombo		9	low	low	low	low	3	8	2	1	0	2	9	4	4	0	2	0	0	1.0	0.0	1.0	0
MineroWarrior		11	low	low-medium	low	low	0	5	0	2	0	5	11	10	0	0	5	0	0	0.0	0.4	0.0	0
PlayExtremeFighter		14	low	low	low	low	5	13	7	1	0	3	14	13	0	0	0	1	0	0.0	0.0	0.0	1
Tank		12	low	low-medium	low	low	1	3	1	2	0	2	12	6	5	0	2	0	0	0.6	0.0	0.0	0

Las clases con mayor cantidad de líneas de código es “FightEngine” por lo que es la clase que tiene menos probabilidad de ser mantenida en futuras modificaciones.
La clase con mayor cantidad de hijos es “CreateVariant” por lo que es la clase con código más utilizado ya que de esta se crean los luchadores tipo mago, tanque y asesino.
La clase con menor cohesión entre métodos es “MakeCombo” por lo que esta clase tiene mayores probabilidades de ser modificada en el tiempo, ya que si se quiere agregar un combo para alguno de los distintos luchadores se tendrá que modificar esta clase.