Los Miserables

Proyecto DesignPatterns

Diseño y desarrollo de un Juego de Combate

Índice

INTRODUCCIÓN 2
Descripción del juego 2
Manual de uso 2
PATRONES 3
1. Strategy 3
2. Decorator 4
3. State 5
4. Abstract Factory 6
5. Singleton 7
6. Template Method 8
7. Facade 9

INTRODUCCIÓN

Descripción del juego

Nuestro juego se basa en el texto de la Ilíada, concretamente en la Guerra de Troya. El juego consta de 2 bandos, Grecia y Troya, de los cuales el jugador elegirá a uno de los 3 héroes posibles de su bando. Una vez elegido, eliges a un Dios, que le otorgará una bendición, la cual configura nuestra estrategia de juego.

Tras los preparativos iniciales, nuestro héroe se dispone a luchar contra los enemigos del bando contrario en una serie de combates individuales hasta llegar al dios líder de la facción contraria. Una vez derrotado, el jugador gana el juego.

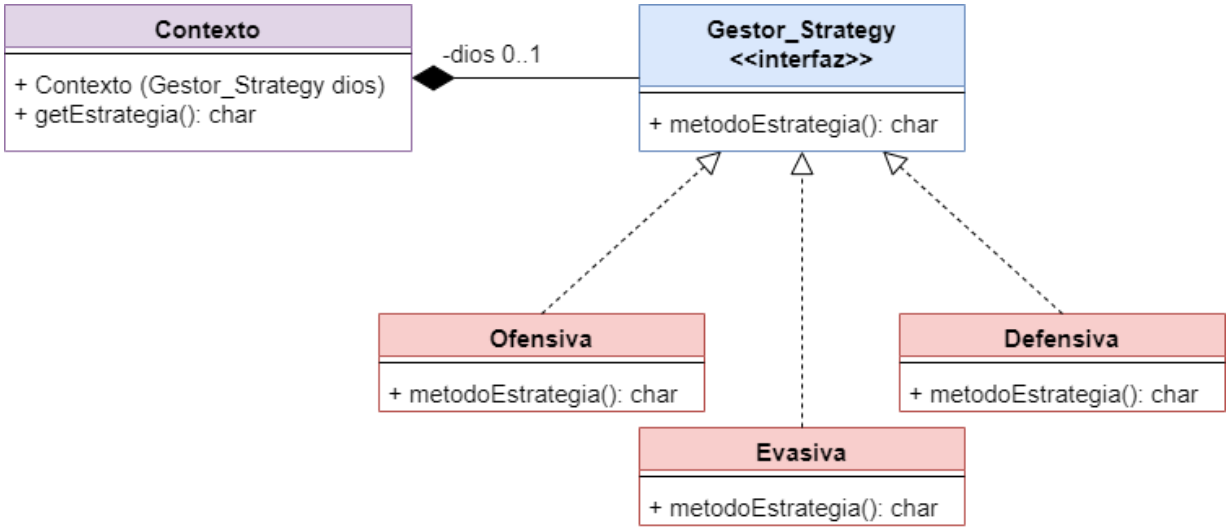
Manual de uso

El programa está configurado para que en la clase main, la clase principal, sólo se ejecute la acción del GameController que pone en marcha el juego. Esa acción, que es la función start(), pone el juego en funcionamiento, con todos sus menús y pequeños textos informativos. Lo único que el jugador va a tener que hacer es introducir los datos iniciales del héroe con el que jugará y la selección de las distintas acciones que adoptará en el combate contra sus enemigos.

PATRONES

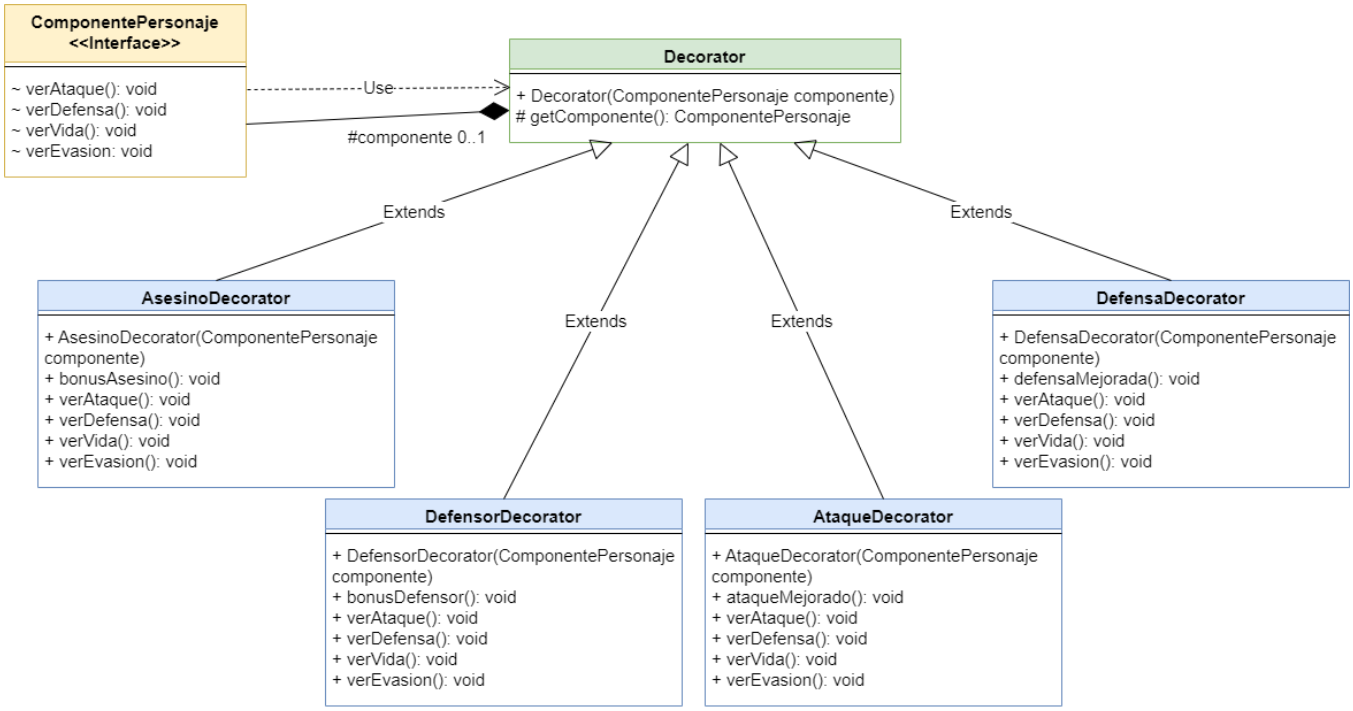
1. Strategy

Este patrón lo que hace es a partir del parámetro introducido en el gestor devuelve una estrategia la cual puede ser 'O' de Ofensiva, 'D' de Defensiva, y 'E' de evasiva.



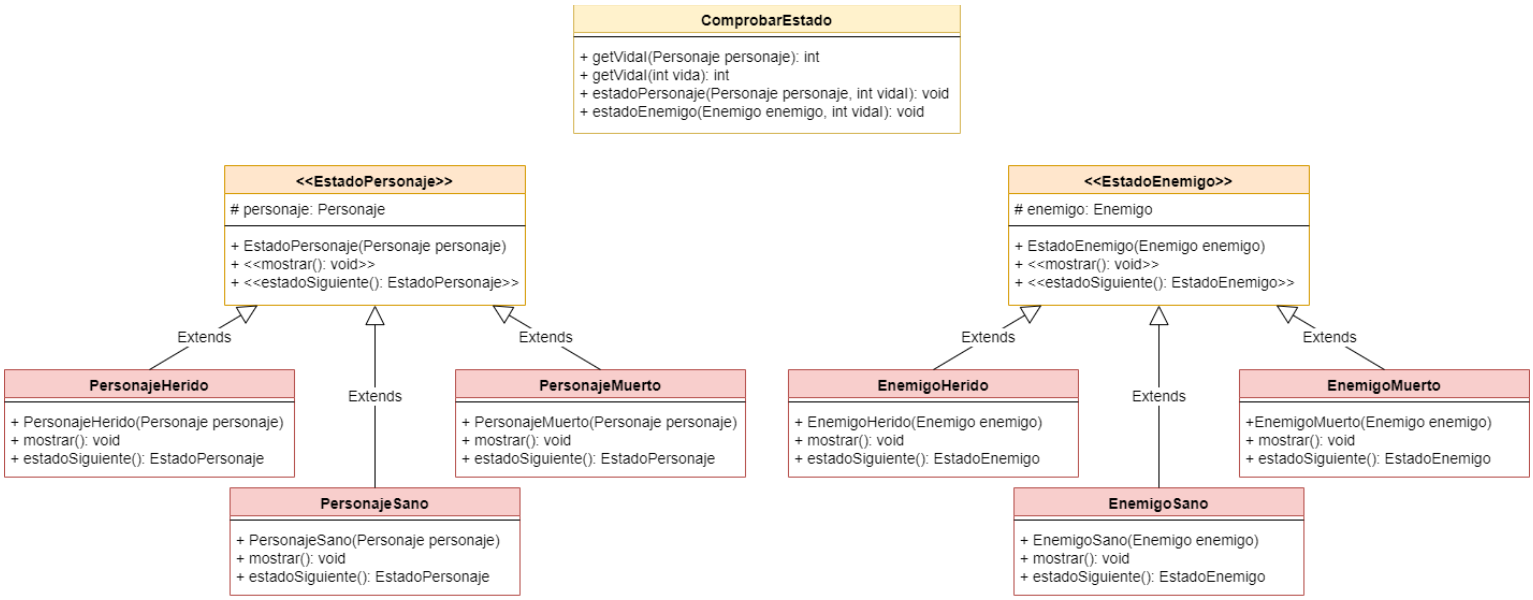
2. Decorator

Este patrón decorará las opciones en combate del héroe según sus estadísticas y estrategias adoptadas en el momento. Si nuestro héroe tiene el rol de asesinos, y se disponen a atacar, se decorará la opción acorde a su rol de asesino. Así mismo, si adoptan una estrategia ofensiva, se decorará el ataque para que 'supuestamente' haga más daño. Si posee el rol de asesino y adopta una estrategia ofensiva, saldrán los 2 decoradores y estará doblemente decorado.



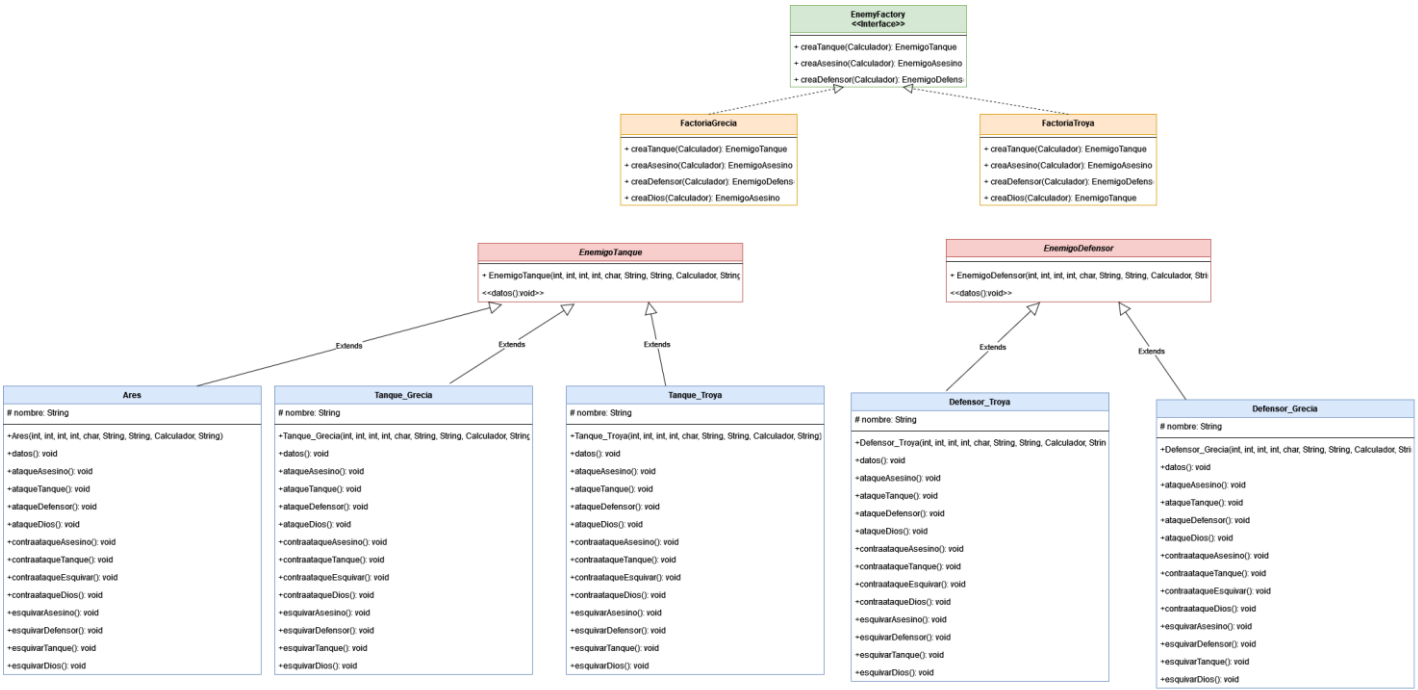
3. State

Este patrón revisa el estado del personaje durante la partida, este funciona con una clase de comprobar estado (“ComprobarEstado”) esta es la que une el patrón con el game controller, la función principal de la clase es mirar en qué estado se encuentra el personaje, para ver eso comprueba la vida y si tiene la vida por debajo del 50% cambia al estado herido y te manda un mensaje por la consola para comunicárselo al usuario. Si la vida baja de 0 el estado cambia a muerto y te manda otro mensaje diciendo que el estado ha cambiado a muerto.



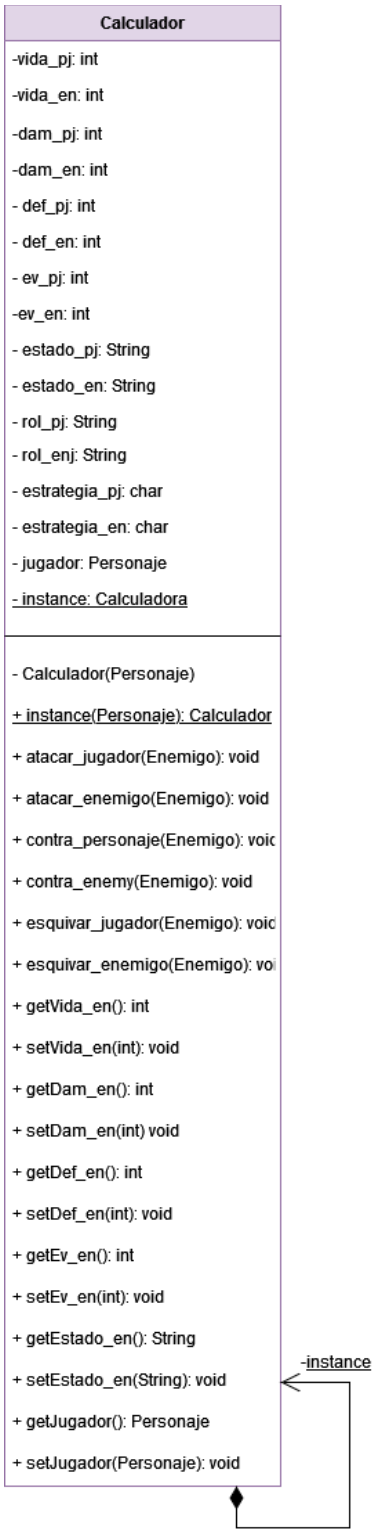
4. Abstract Factory

Este patrón crea a los diferentes enemigos, conteniendo sus atributos y acciones a realizar. Podemos ver distintos enemigos según el combate en el que se encuentre el jugador además de 2 jefes finales. A nivel general son 3 tipos de enemigos que varían sus estadísticas según al nivel de dificultad al que pertenezcan.



5. Singleton

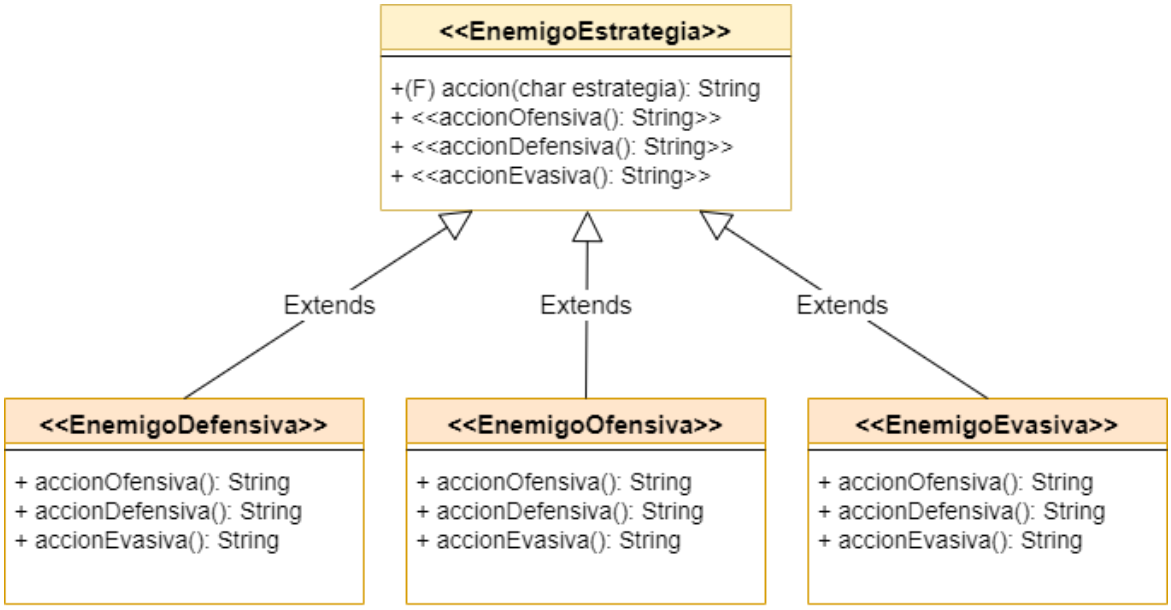
Este patrón se encarga de realizar todos los cálculos pertenecientes al combate, los cuales tienen operadores asociados según la acción que se realice. Además, controla indirectamente algunos atributos del enemigo y el jugador, como su vida.



6. Template Method

El Template Method se ocupa de devolver la siguiente acción que va a llevar a cabo el enemigo. En el método ”accion” se le pasa por parámetro el tipo de estrategia que tiene el enemigo, ya que, en función de esta, la probabilidad de que sea una acción u otra varía de la siguiente manera:

- La estrategia ofensiva tiene una probabilidad del 80% de atacar, del 5% de esquivar y del 15% de contraatacar.
- La estrategia defensiva tiene una probabilidad del 50% de atacar, del 20% de esquivar y del 20% de contraatacar.
- La estrategia evasiva tiene una probabilidad del 60% de atacar, del 30% de esquivar y del 10% de contraatacar.



7. Facade

El patrón Facade consiste en hacerle una fachada al proyecto para que el usuario pueda interactuar de forma sencilla con este. Básicamente desde el main solo crearemos un objeto de tipo GameController y haremos una llamada a su método start() para iniciar el juego.

El método start() lleva el orden de las llamadas a los métodos a ejecutar.

Lo primero de todo nos muestra tres menús. En el primero elegimos la facción con la que queremos jugar, Grecia o Troya. Una vez elegido el bando escogemos uno de los tres héroes de cada facción con un nuevo menú. Un tercero nos pregunta por la bendición del dios que queremos recibir.

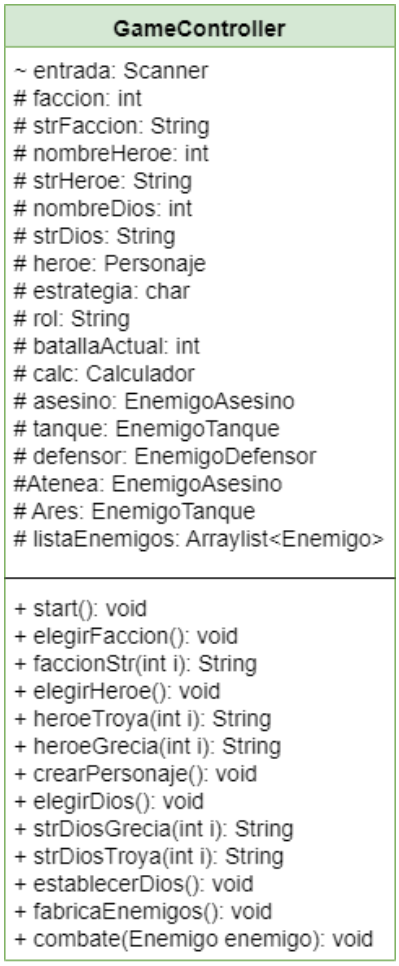
Al escoger facción automáticamente crearemos un array de enemigos, que serán sol héroes de la facción contraria, además de un dios rival como jefe final del juego.

Una vez tenemos estos datos creamos nuestro héroe con sus atributos. Los más importantes son: La estrategia, definida por el dios que hemos elegido. Pudiendo ser Ofensiva, Evasiva o Defensiva. Y el rol del protagonista, definido por el héroe que hayamos elegido. Los roles para escoger son: asesino, tanque o defensor. Estos dos atributos van a condicionar nuestro juego.

El siguiente paso dentro del método start() es hacer un bucle, iterando sobre la lista de enemigos y enfrentándonos a ellos de uno en uno. Este enfrentamiento se hace a través del método combate. Este método se ejecuta por cada enemigo del array o hasta que el héroe muera.

El sistema de combate es por turnos, con lo que a cada turno podremos elegir entre atacar al enemigo, preparar una esquiva o preparar nuestra defensa. Dependiendo de la acción a realizar llamaremos a la calculadora para que nos retorne, según los porcentajes de probabilidad en los que influyen la estrategia y el rol, el resultado del ataque, la esquiva o la defensa.

Este proceso se repite hasta que hemos derrotado a todos nuestros enemigos, en cuyo caso ganamos el juego, o nos derrotan, en el que perdemos.


GameController
~ entrada: Scanner
# faccion: int
# strFaccion: String
# nombreHeroe: int
# strHeroe: String
# nombreDios: int
# strDios: String
# heroe: Personaje
# estrategia: char
# rol: String
# batallaActual: int
# calc: Calculador
# asesino: EnemigoAsesino
# tanque: EnemigoTanque
# defensor: EnemigoDefensor
#Atenea: EnemigoAsesino
Ares: EnemigoTanque
listaEnemigos: Arraylist<Enemigo>
+ start(): void
+ elegirFaccion(): void
+ faccionStr(int i): String
+ elegirHeroe(): void
+ heroeTroya(int i): String
+ heroeGrecia(int i): String
+ crearPersonaje(): void
+ elegirDios(): void
+ strDiosGrecia(int i): String
+ strDiosTroya(int i): String
+ establecerDios(): void
+ fabricaEnemigos(): void
+ combate(Enemigo enemigo): void

8. Otros y Main

En estos paquetes gestionamos las clases que no son patrones, pero fundamentales para el proyecto, como son la clase Enemigo o la de Personaje, así como el propio main.