MEMORIA PROYECTO GLOBAL DISEÑO DE SOFTWARE

autor: Gonzalo Fernández Suárez

MANUAL DE USO

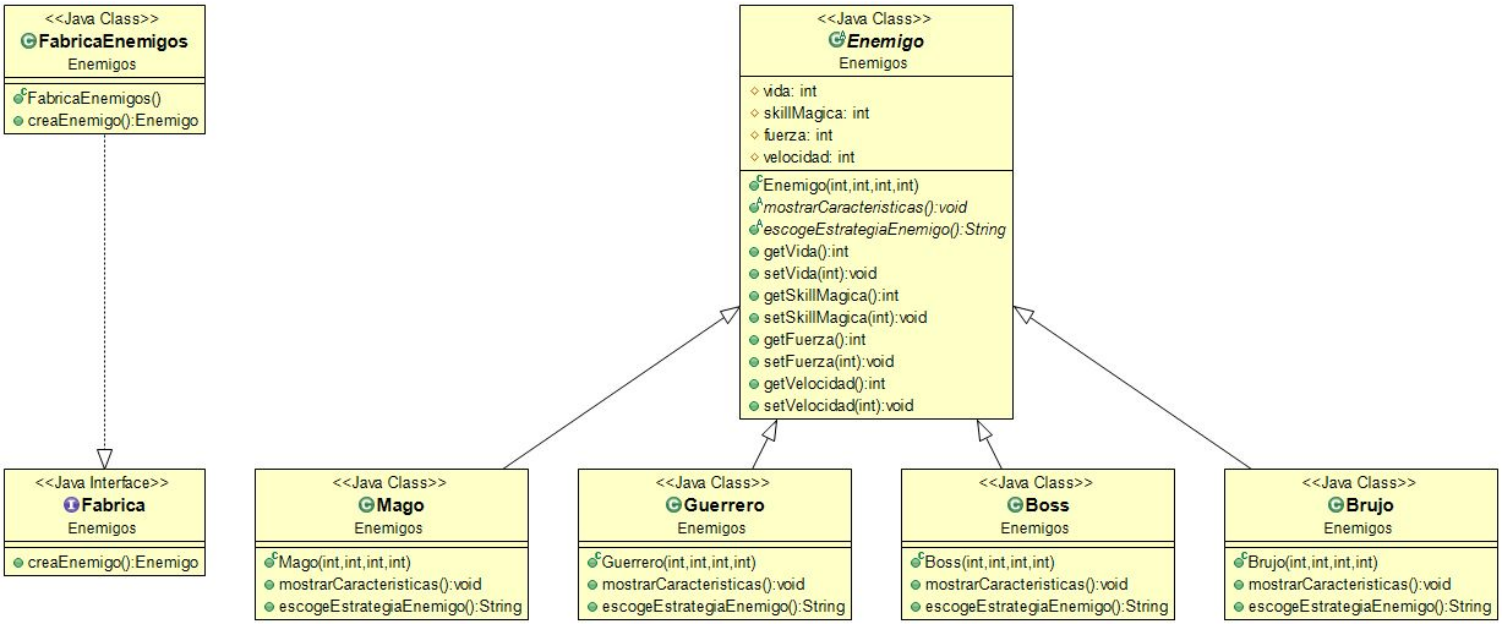
1º Ejecuta la clase main del paquete Singleton
2º Escoge qué características quieres que tenga tu personaje
3º Empieza el combate

A la hora de elegir estrategia ten en cuenta lo siguiente:

- El ataque directo se calcula utilizando la fuerza
- El ataque mágico en función del skill mágico
- El ataque sorpresa en función de la velocidad
- La posición defensiva se calcula en función del skill mágico y la velocidad. Este ataque no causa daño al contrincante, tiene un 50% de probabilidades de surtir efecto y si lo hace cura al que lo ha utilizado.

El proyecto está dividido en 5 paquetes: Enemigos, Jugador, Estrategias y Singleton.

Paquete: ENEMIGOS



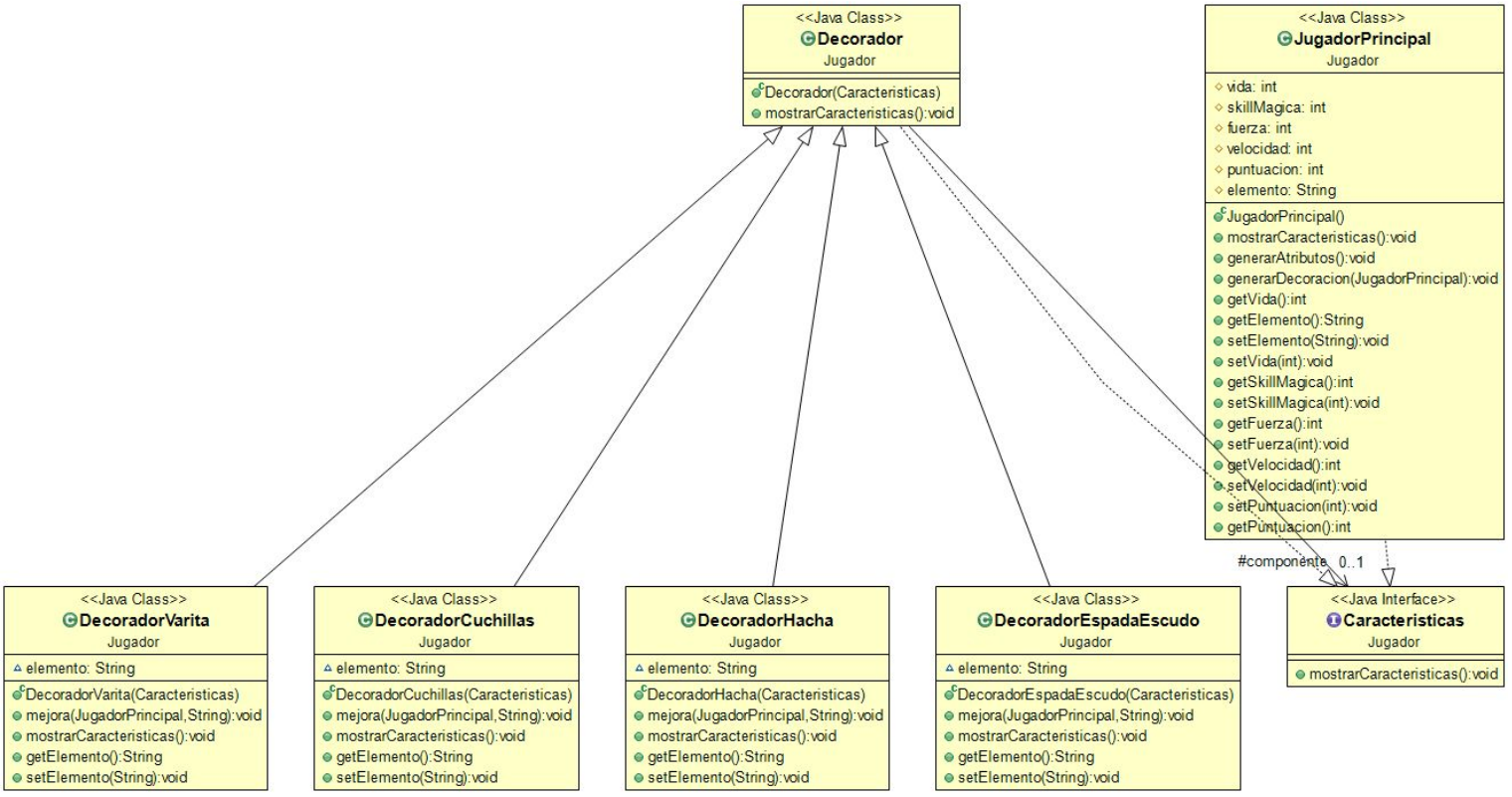
En este paquete está integrado el patrón **Abstract Factory**.

Los enemigos tienen los mismos atributos que el jugador: vida, fuerza, skill mágica y velocidad.

Como aspectos a destacar, en la clase *FabricaEnemigos* el método *creaEnemigo()* es el encargado en generar los objetos. Este método genera enemigos de forma aleatoria y lo clasifica siguiendo una serie de pautas. En función de el enemigo generado, este será de una vocación u otra (Mago, guerrero, brujo o monstruo).

Cada subclase de *Enemigo* tiene su propio método *escogeEstrategiaEnemigo()* que se utiliza para obtener el comportamiento en combate del enemigo y va variando en función del tipo de enemigo. Está implementado así debido a la utilización del patrón **Template Method.** Por ejemplo: el *Mago* tiene un 40% de probabilidades de elegir ataque mágico, 40% de ataque defensivo, 20% de ataque sorpresa y 0% de ataque directo. En cambio el *Guerrero* tiene: 40% ataque directo, 20% ataque sorpresa, 20% ataque mágico y 20% posición defensiva. Cada uno tiene su comportamiento.

Paquete: JUGADOR

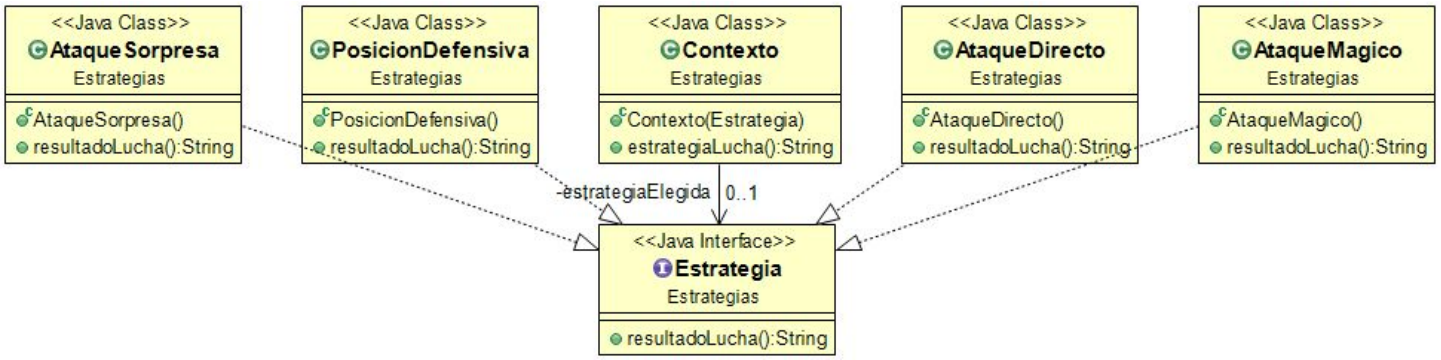


En este paquete podemos observar el patrón **Decorator** que se utiliza para que el usuario pueda decorar su personaje escogiendo el arma que desea utilizar, esto hace que cambien sus atributos. Por ejemplo: si eliges espada y escudo tu vida aumentará +40% y tu fuerza crecerá +20%.

Además, el jugador puede elegir también como hechizar su arma, escogiendo entre distintos elementos, en función del escenario de combate en el que se encuentre obtendrá una bonificación en sus ataques o no. Por ejemplo: si has hechizado tu arma con “Fuego” cuando combatas en el “Bosque” obtendrás +5 de bonificación en todos tus ataques, se calculará el ataque de forma normal y se sumará 5 al resultado, ya sea daño al enemigo o curación para el jugador.

En la clase *JugadorPrincipal* los métodos principales son *generarAtributos()* y *generarDecoracion()* que son los que permiten al usuario construir su personaje.

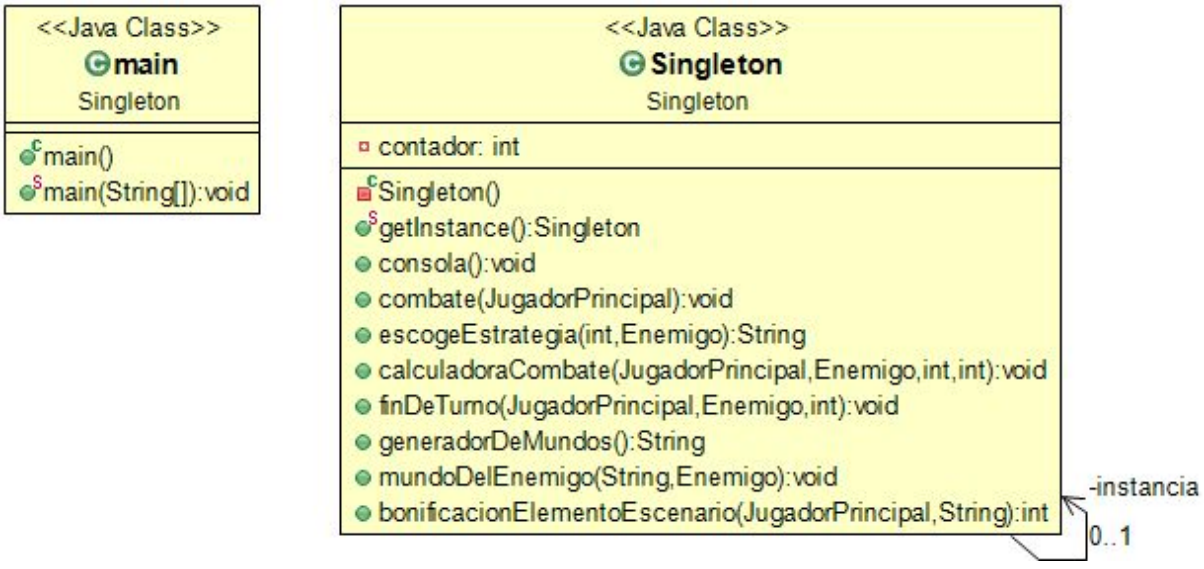
Paquete: ESTRATEGIAS



En este paquete se gestionan las posibles estrategias a elegir en combate utilizando el patrón **Strategy.**

El método *estrategiaLucha()* de la clase *Contexto* lo único que hace es llamar al método *resultadoLucha()* del tipo de objeto (estrategia) en cuestión. *resultadoLucha()* devuelve una string para identificar el tipo de estrategia escogida. Esta parte quedará más clara cuando lea el paquete Singleton.

Paquete: SINGLETON



Este paquete podría contener solo una clase, la clase Singleton, pero he preferido poner el main a parte porque a medida que programaba necesitaba ir haciendo pruebas y por cuestión de orden me era más cómodo así.

Como su nombre indica, aquí se encuentra el patrón **Singleton**, que me permite tener siempre una única instancia de la clase singleton, lo que me resulta perfecto para construir el cerebro del juego.

En esta clase considero que los métodos son bastante importantes para entender el juego, por lo que te dejo un breve resumen de cada método y su finalidad.

getInstance() : me devuelve un objeto Singleton que es la única instancia que hay, el único.

consola() : es el método con el que empieza el juego. Como podemos observar en el main lo único que hay son dos líneas:

```
Singleton singleton = Singleton.getInstance();
singleton.consola();
```

En este método llama a los métodos necesarios para que el usuario genere su personaje y, después, se empieza el combate llamando al método *combate()*.

combate() : en este método empieza el combate.

1º Se genera un escenario de lucha (Aleatoriamente entre: Bosque, Océano, Montaña, Cueva y Volcán) y se comprueba si el jugador tiene la bonificación por el elemento del arma.

2º A partir del método **Abstract Factory,** anteriormente explicado, se construye el enemigo a desafiar en ese combate. Este también puede recibir bonificaciones en sus atributos dependiendo del escenario, gracias metodo *mundoDelEnemigo(...)*.

3º Se obtiene un número aleatorio entre 1 y 2 que definirá quien empieza el combate, si el jugador o el enemigo, y se llama a los métodos *calculadoraCombate(...)* y *finDeTurno(...)* , respectivamente.

calculadoraCombate(...) : primero llama al método *escogeEstrategia(...)* que en función de a quien le toque, preguntará al jugador o al enemigo que estrategia desea utilizar. Una vez se sabe la estrategia elegida realiza los cálculos del ataque y modifica los atributos del jugador o el enemigo en caso de ser necesario. En este método se encuentran las fórmulas que definen cada tipo de ataque.

finDeTurno(...) : este método comprueba si, después del ataque,el jugador se ha quedado sin vida, terminando la partida, o si el enemigo se ha quedado sin vida, pasando al siguiente combate. Si el jugador sigue teniendo vida y el enemigo también prosigue el combate.