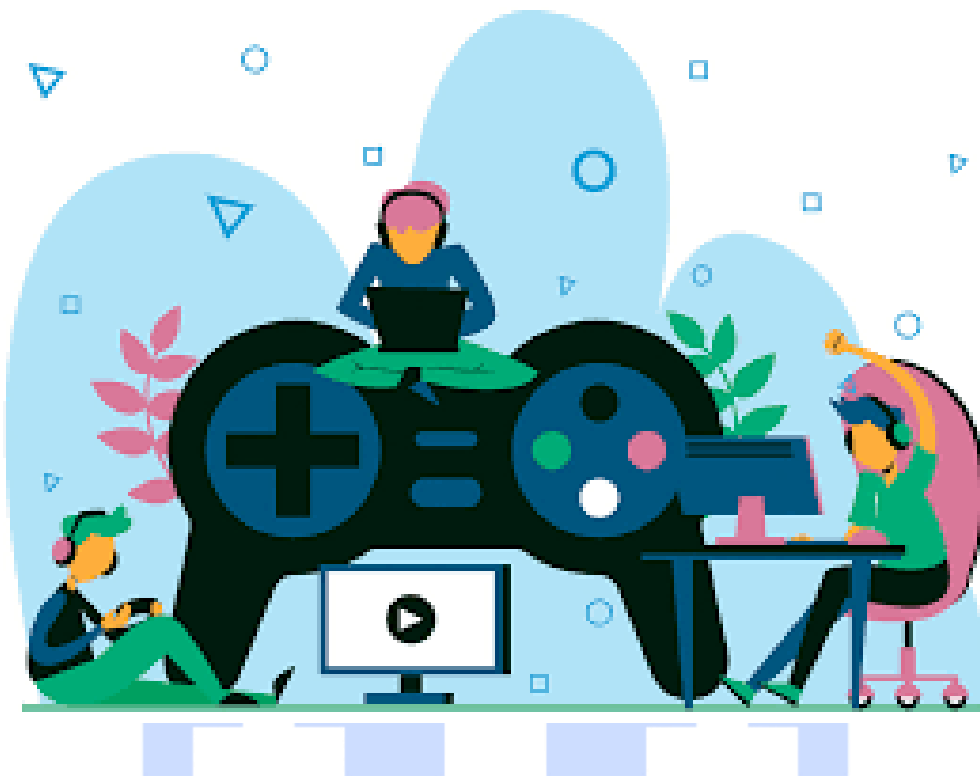


DISEÑO DE SOFTWARE

TRABAJO FINAL



Héctor Asorey

David García

Alexis Gómez



ÍNDICE

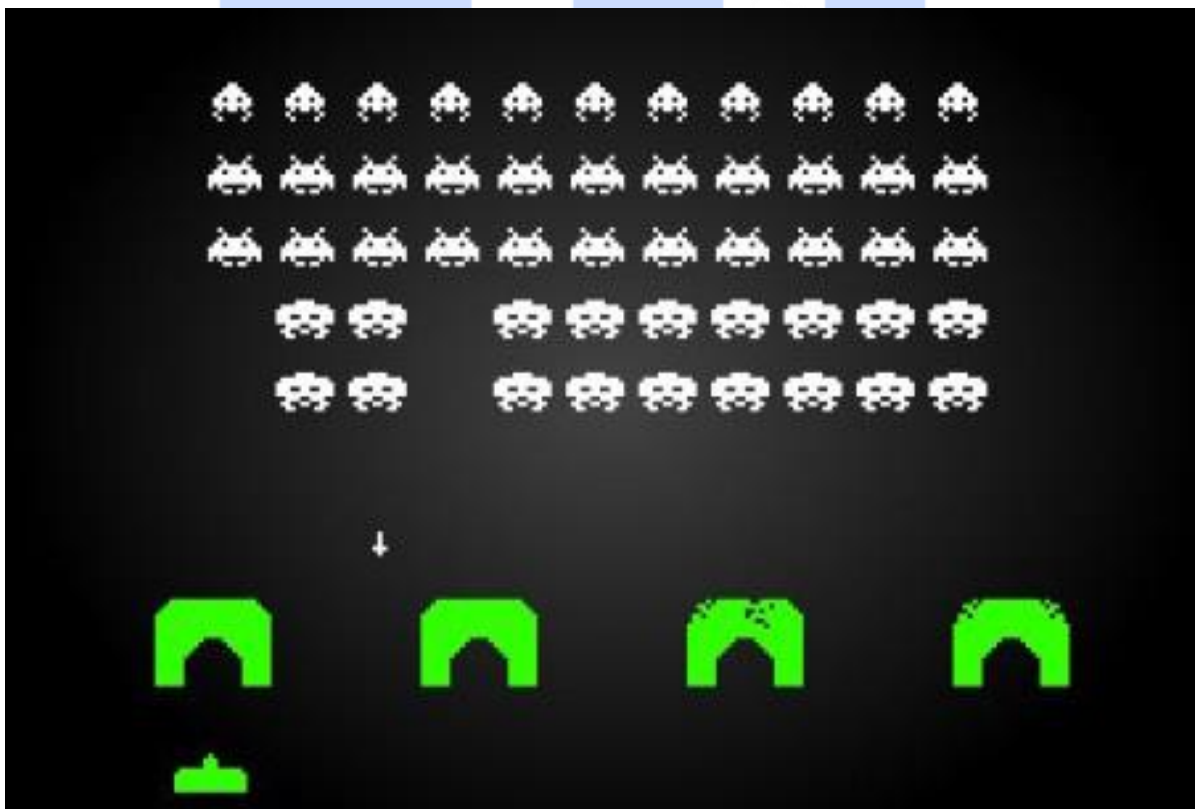
| | |
|--|-----------|
| DESCRIPCIÓN DEL JUEGO | 2 |
| MANUAL DE USO | 6 |
| ABSTRACT FACTORY PATTERN | 13 |
| MÉTODO “CREATEFELBAT()” | 14 |
| MÉTODO “CREATEINQUISITOR()” | 14 |
| MÉTODO “CREATEINFERNAL()” | 14 |
| MÉTODO “CREATEENEMY()” | 14 |
| MÉTODO “CREATEBOSS()” | 15 |
| STATE PATTERN | 16 |
| DIAGRAMA DE ESTADOS..... | 17 |
| EXPLICACIÓN DE LOS DISTINTOS ESTADOS | 17 |
| CAMBIOS EN FUNCIÓN DEL ESTADO | 18 |
| STRATEGY PATTERN | 19 |
| EXPLICACIÓN DE LAS DISTINTAS ESTRATEGIAS | 19 |
| Defensiva: | 19 |
| Ofensiva: | 20 |
| Curativa:..... | 20 |
| Buffos / Aumento de estadísticas:..... | 20 |
| TEMPLATE PATTERN | 21 |
| RANDOM PROBABILITY GENERATOR FAIR..... | 22 |
| RANDOM PROBABILITY GENERATOR UNFAIR | 23 |
| DECORATOR PATTERN | 24 |
| DECORATOR PATTERN II | 26 |
| SINGLETON PATTERN | 27 |
| GAME CONTROLLER | 29 |
| PERSONAJES | 30 |
| DESCRIPCIÓN DE LOS ENEMIGOS..... | 31 |
| Infernales | 31 |
| Inquisidores..... | 32 |
| Murciélagos Viles | 33 |
| Jefes..... | 34 |

DESCRIPCIÓN DEL JUEGO

El proyecto final realizado es un juego arcade basado en rondas. El objetivo del jugador es ir avanzando, derrotando a distintos enemigos y lograr la máxima puntuación posible

¿Qué es un juego arcade? Un juego arcade es aquel que recuerda a los juegos de las máquinas recreativas. Las máquinas recreativas funcionaban con monedas, es decir, tu pagabas para jugar y cuando se terminaban las oportunidades o vidas que tenías, era necesario volver a pagar. Así, los juegos arcade se caracterizaban por ser relativamente difíciles, pero sencillos de jugar, para que el jugador perdiese siempre pero que siempre quisiera volver a intentarlo. En estos juegos, siempre había una “leaderboard”, o tablón de puntuaciones donde se mostraban los puntos que habían conseguido cada jugador.

Nuestro juego sigue los pasos de los juegos arcade modernos. Actualmente no hay que pagar por cada vez que quieres intentar jugar, por lo que este género de juegos se ha tenido que actualizar. De esta manera, un juego arcade actualmente se entiende como un juego sencillo de jugar mecánicamente (controles y concepto de juego sencillos), que ofrece una experiencia desafiante, rejugable, y basado en obtener la mayor cantidad de puntos posible.



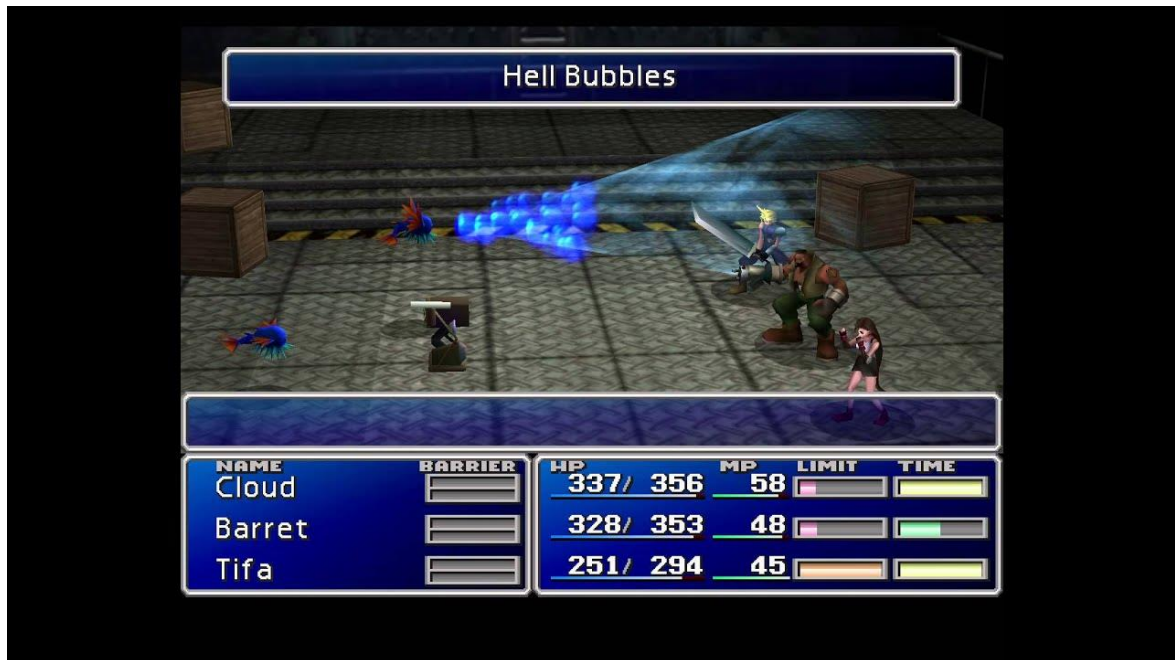
Space Invaders, 1978. “El rey de las recreativas”.

Entonces, ¿cómo es nuestro juego? Nuestro juego pertenece al género de los juegos de ataque por turnos. Este género de juegos se basa en rondas. Así, el jugador escogerá una acción a realizar (atacar, usar un objeto, defenderse), y tras realizar la acción, será el turno del enemigo. El orden también puede ser el inverso. Una vez hecho todo esto, se completa la ronda, y vuelta a empezar hasta que termine el combate. Depende de cómo se plantee el juego, pueden ser combates de 1v1 (un personaje contra un enemigo), un personaje contra varios enemigos, o varios personajes controlados por el jugador contra varios enemigos. Hemos escogido la primera opción.

Cabe destacar también, que a medida que avancemos, es decir, a medida que triunfemos en los combates, el juego será cada vez más difícil. Así, los enemigos del mundo 1 harán menos daño al jugador y tendrán menos vida que un enemigo del mundo 2.

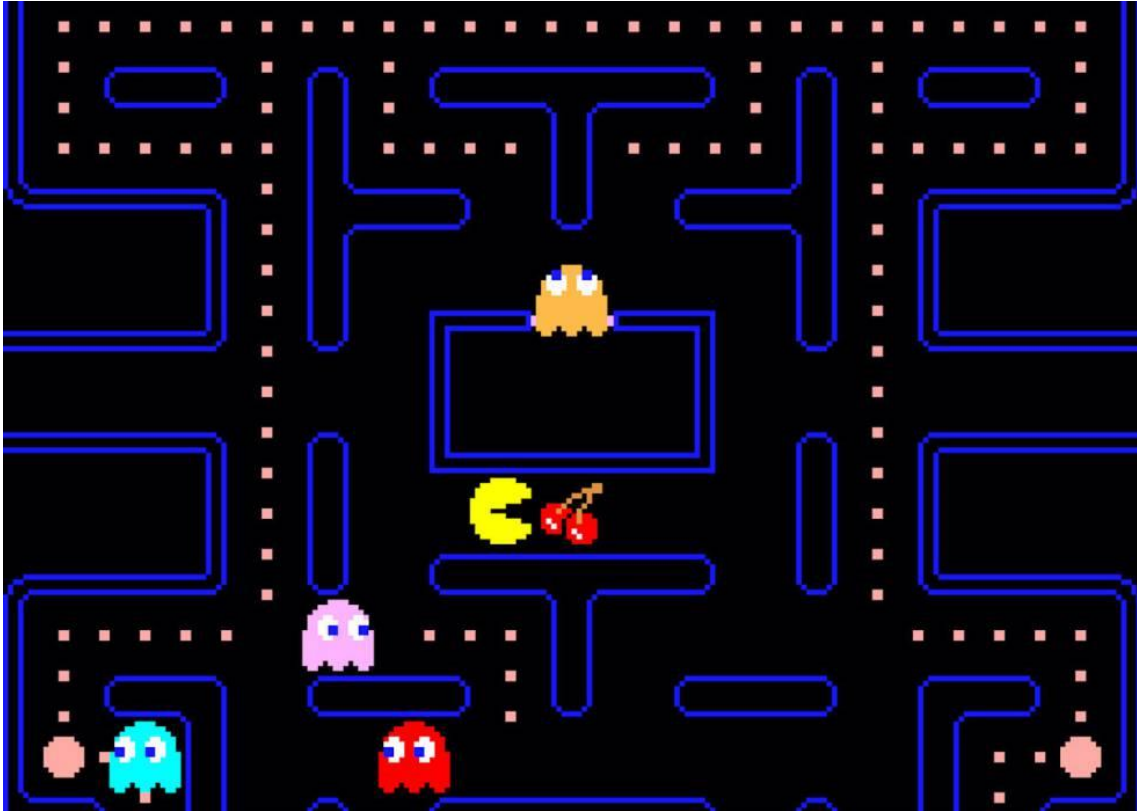


Pokemon. Máximo exponente de juegos basados en combate por turnos de 1 personaje contra un enemigo.



Final Fantasy VII, 1997. Ejemplo de juego de combate por turnos de varios personajes contra varios enemigos.

Por lo tanto, nuestro juego pertenece al género de los juegos de combate por turnos, en específico, a aquellos cuyos combates son de uno contra uno. El objetivo del jugador será obtener la máxima puntuación. El juego solo terminará si el jugador así lo quiere al terminar los mundos o al morir, pues se ofrece la opción de volver a empezar para conseguir más puntos. Habrá varios mundos, de dificultad incremental. Nos iremos enfrentando a distintos enemigos en cada mundo que se generarán de forma aleatoria. Al final de cada mundo, nos enfrentaremos a un enemigo especial, un jefe o “boss”, que será el enemigo más desafiante del mundo en el que estemos. En las rondas, empezará siempre el enemigo.

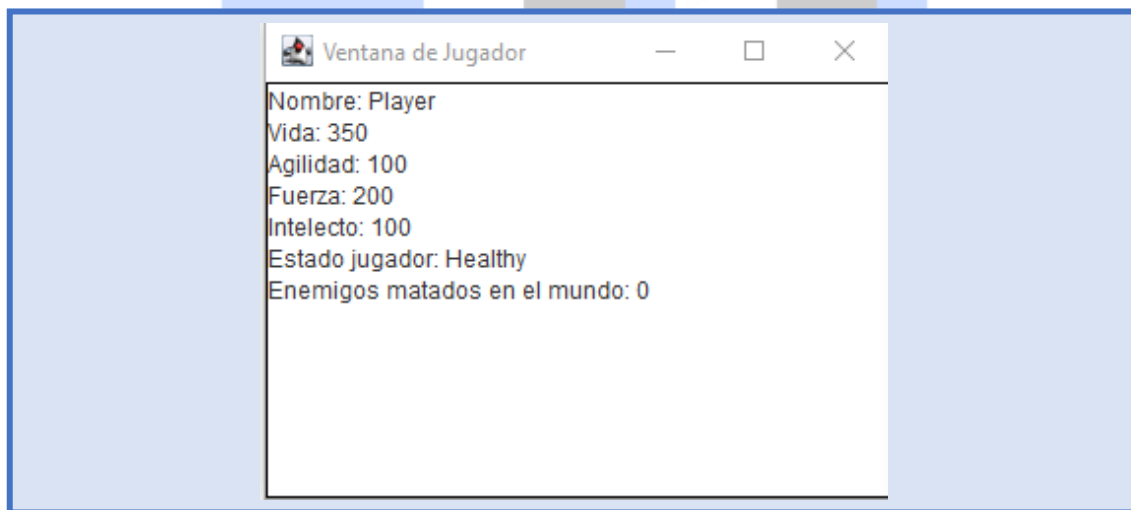
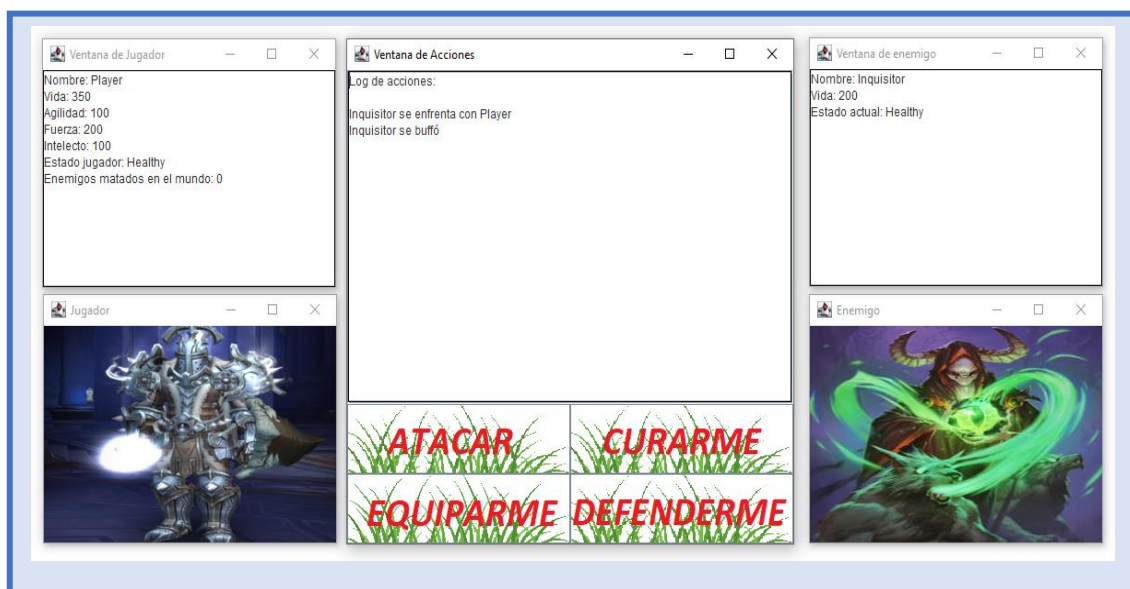


Pac-man, 1980. El juego solo terminaba cuando perdías. El objetivo era conseguir el mayor número de puntos.

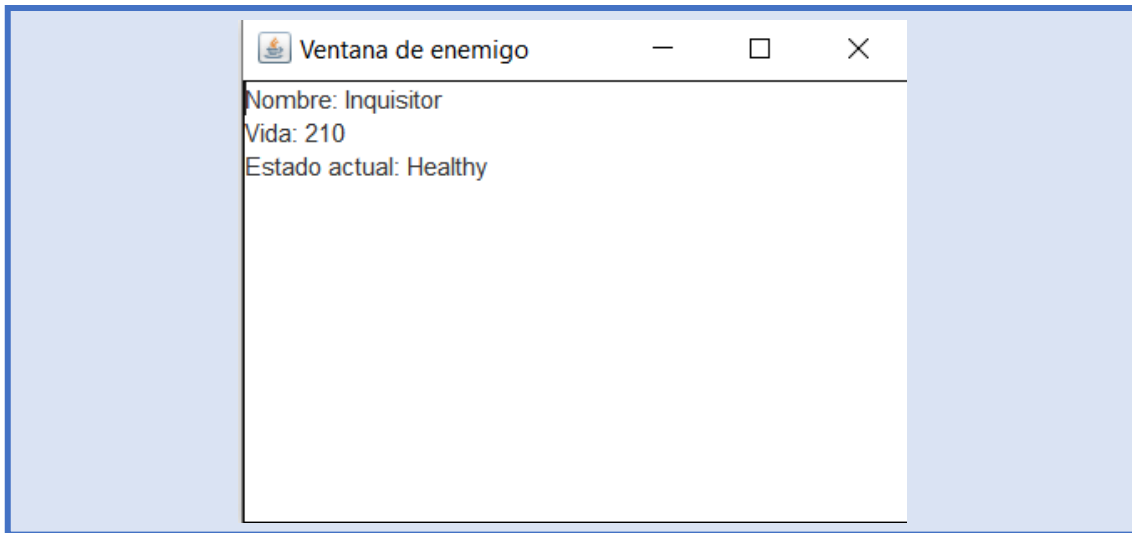
Tal como los juegos arcade, no tendremos una historia, sino que solo se ofrece un contexto al jugador para empezar a jugar. El jugador controla a un caballero maldito, condenado a estar atrapado en el infierno. Éste intentará ganarse la redención destruyendo a las hordas de los demonios. Debido a la maldición, nunca puede morir del todo, pero cuando lo derrotan, se reinicia el ciclo al cuál está atado.

MANUAL DE USO

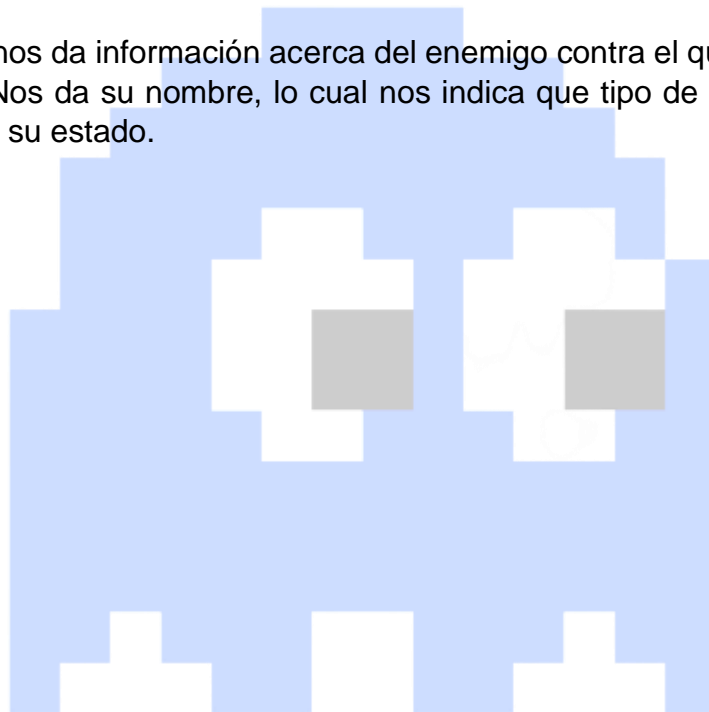
Al empezar a jugar, nos saldrán cinco ventanas como estas:

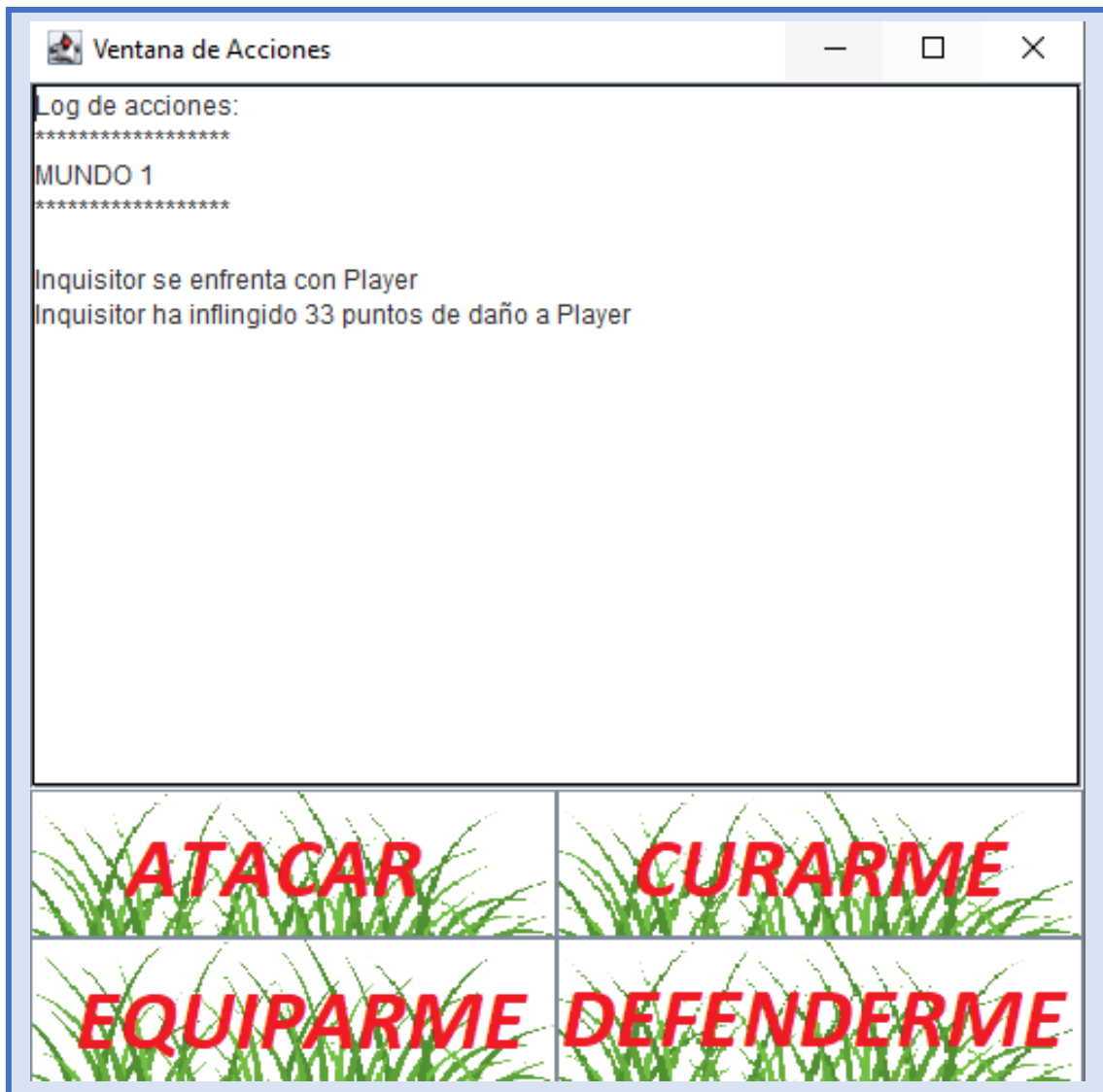


Esta ventana nos da información en todo lo relativo al jugador. Así, podemos ver en todo momento la información de nuestro personaje. Se muestra la vida que tenemos, los atributos que van a determinar el daño que realizamos, cuanto nos curamos (agilidad, fuerza, intelecto), el estado del jugador en ese momento y los enemigos que hemos derrotado en el mundo en el que nos encontramos.



Esta ventana nos da información acerca del enemigo contra el que nos estamos enfrentando. Nos da su nombre, lo cual nos indica que tipo de enemigo es, su nivel de vida y su estado.

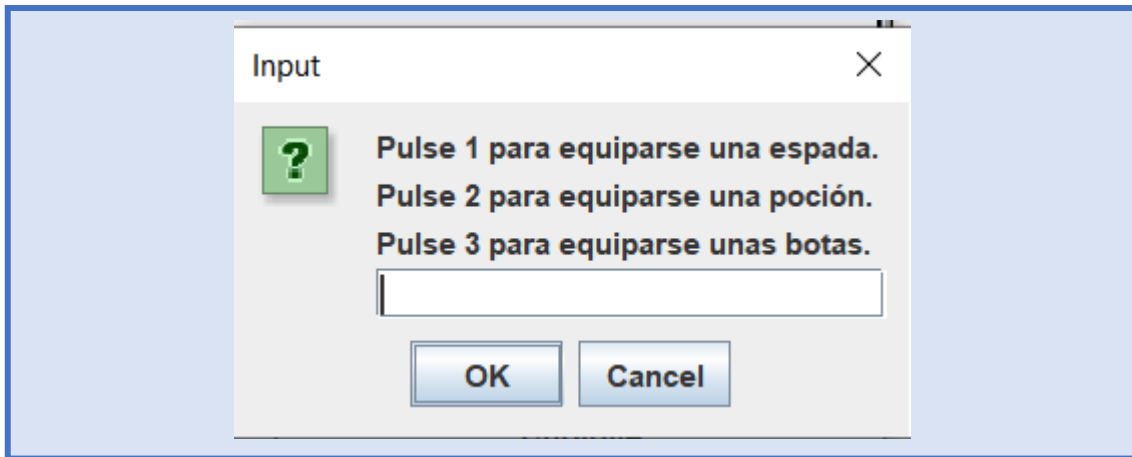




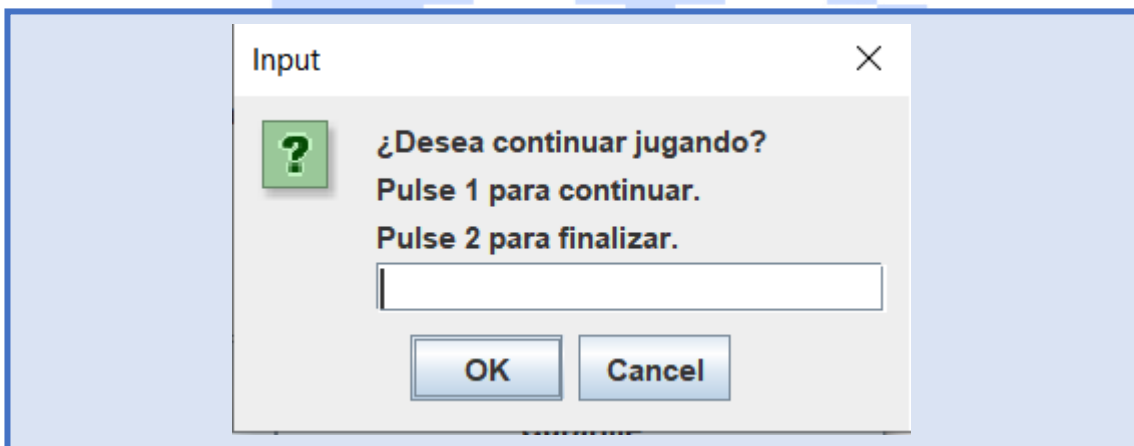
Esta ventana es la encargada de interaccionar con el usuario.

Consta de una caja de texto donde nos aparecerá información relativa al juego. En esta caja de texto, nos aparecerá el mundo en el que estamos, y en cada ronda nos dirá contra que enemigo estamos luchando, que acción ha realizado el enemigo, que acción ha realizado el jugador y si ha muerto el jugador o el enemigo.

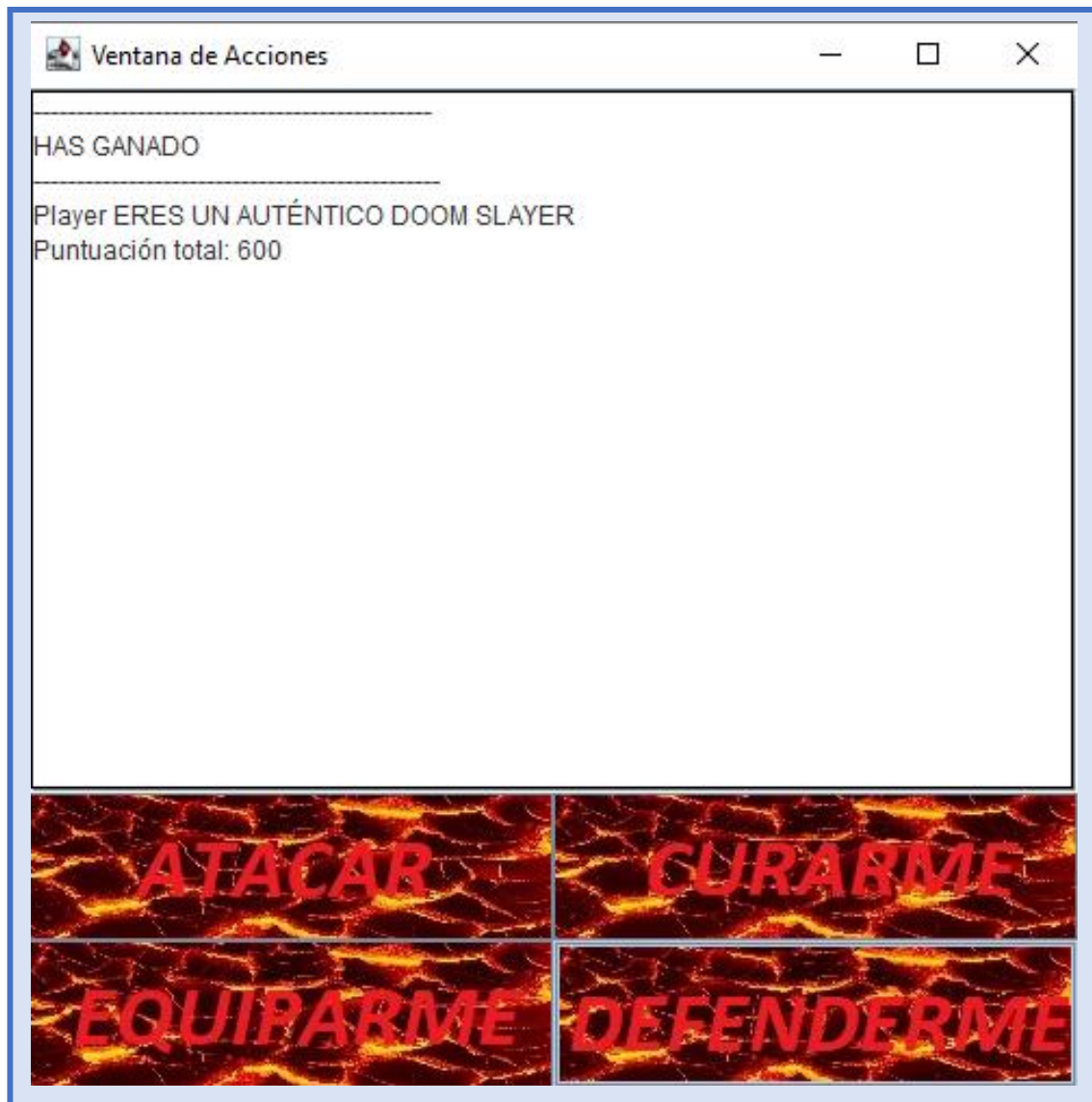
Los botones de debajo servirán para interactuar con el jugador. Así, podremos elegir cuatro opciones. Si pulsamos el botón "atacar", realizaremos un ataque severo, el cual quitará una gran cantidad de vida. Si pulsamos el botón "curarme", recuperaremos puntos de vida, pero nunca recuperaremos más de la vida máxima del jugador. Si pulsamos el botón "defenderme", realizaremos un ataque que quitará una mediana cantidad de vida, y nos curaremos un poco. Si pulsamos el botón "equiparme", podremos equiparnos con distintos objetos que nos aumentarán las estadísticas.



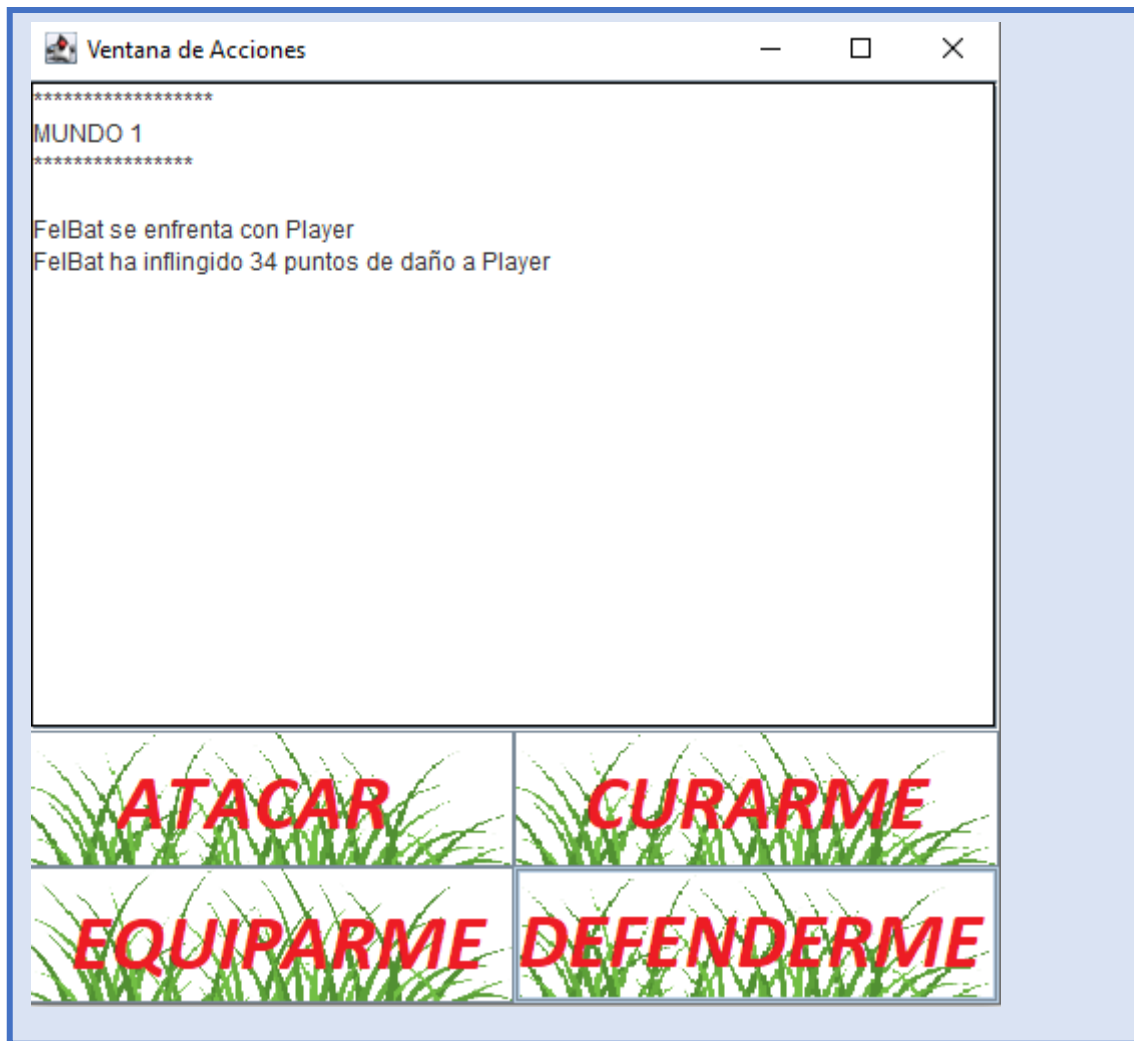
Si pulsamos el botón “equiparnos” aparecerá esta ventana. Si pulsamos 1, se nos equipará una espada, la cual aumenta de forma notable la fuerza, resultando en que causas más daño al enemigo. Si pulsamos 2, se nos equipará una poción, la cual aumenta el intelecto resultando en mayor recuperación de puntos de vida al curarnos. Si pulsamos 3, se nos equiparán unas botas, las cuales aumentan nuestra agilidad, resultando también en una mejora de daño.



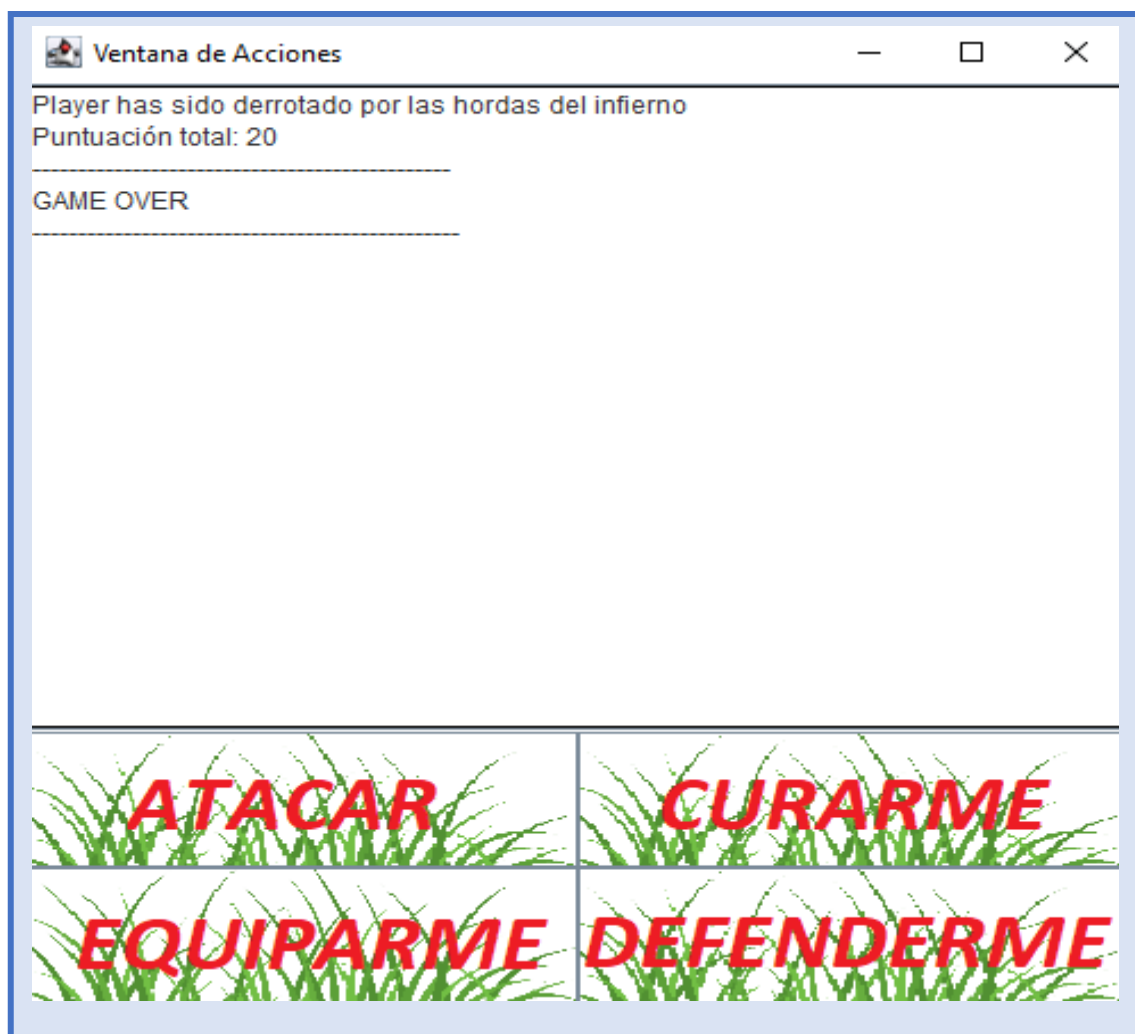
Al terminar con todos los enemigos del mundo 3, se nos preguntará si queremos terminar la partida o si queremos continuar para obtener más puntuación.



Si decidimos finalizar, se nos mostrará un mensaje de victoria y los puntos que hemos obtenido por derrotar a los diferentes enemigos. Tras unos segundos de espera, se cerrará el juego.

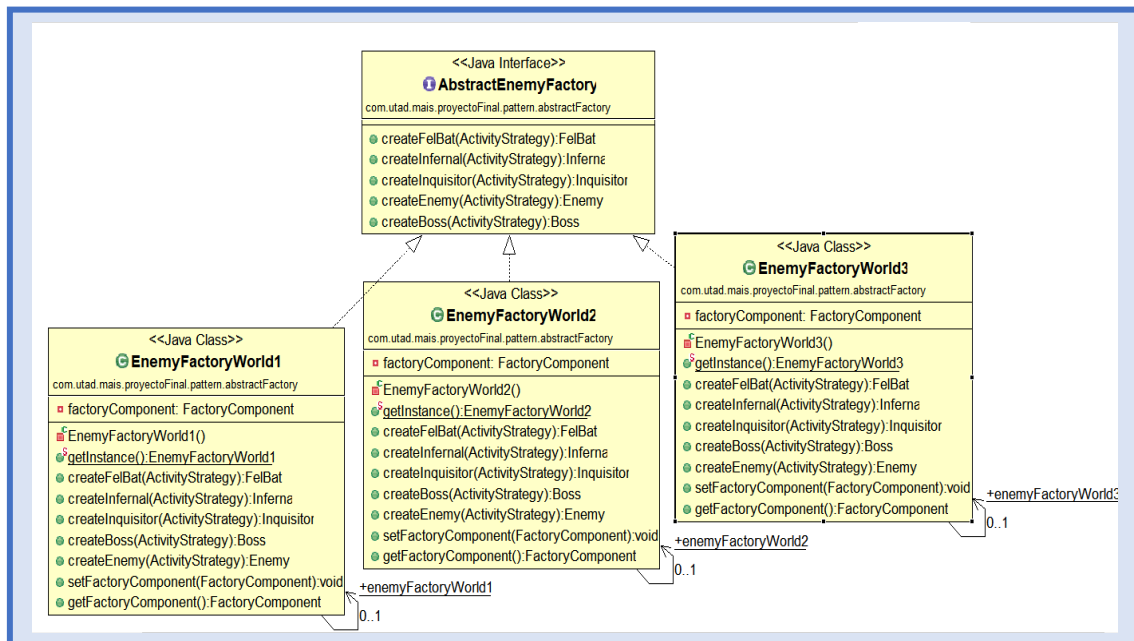


Si decidimos continuar, se reinicia el juego, aumentando la dificultad (los enemigos tendrán más vida y harán más daño al jugador).



Si morimos, se nos mostrará un mensaje de derrota y nos aparecerán los puntos que hemos obtenido. Tras unos segundos de espera, se cierra el juego.




ABSTRACT FACTORY PATTERN







El patrón Abstract Factory nos va a servir para unificar la instanciación de los personajes enemigos.

Consiste en factorías fundamentales cuyo número depende del número de mundos del juego y una interfaz común que implementa cada una de las factorías. El contexto de la ejecución de este patrón será el gameController quien delegará por composición la creación de los enemigos a un atributo del tipo factoría. De esta manera, se consigue que exista una fábrica independientemente por cada mundo del juego.

Las tres fábricas fundamentales asociadas son:

-  EnemyFactoryWorld1: Mundo 1
-  EnemyFactoryWorld2: Mundo 2
-  EnemyFactoryWorld3: Mundo 3

Cada una de estas factorías fundamentales cuenta con los siguientes cuatro métodos:

-  createFelBat().
-  createInquisitor().
-  createInfernal().
-  createEnemy.

MÉTODO “CREATEFELBAT()”

El método createFelBat (ActivityStrategy strategy) devolverá un objeto del tipo FelBatWorld1, FelBatWorld2 o FelBatWorld3; dependiendo del tipo de factoría que se utilice. En la instanciación del objeto se le deberá pasar una estrategia para que el constructor pueda inicializar la estrategia de combate asignada a este tipo de enemigo.

MÉTODO “CREATEINQUISITOR()”

El método createInquisitor (ActivityStrategy strategy) devolverá un objeto del tipo InquisitorWorld1, InquisitorWorld2 o InquisitorWorld3; dependiendo del tipo de factoría que se utilice. En la instanciación del objeto se le deberá pasar una estrategia para que el constructor pueda inicializar la estrategia de combate asignada a este tipo de enemigo.

MÉTODO “CREATEINFERNAL()”

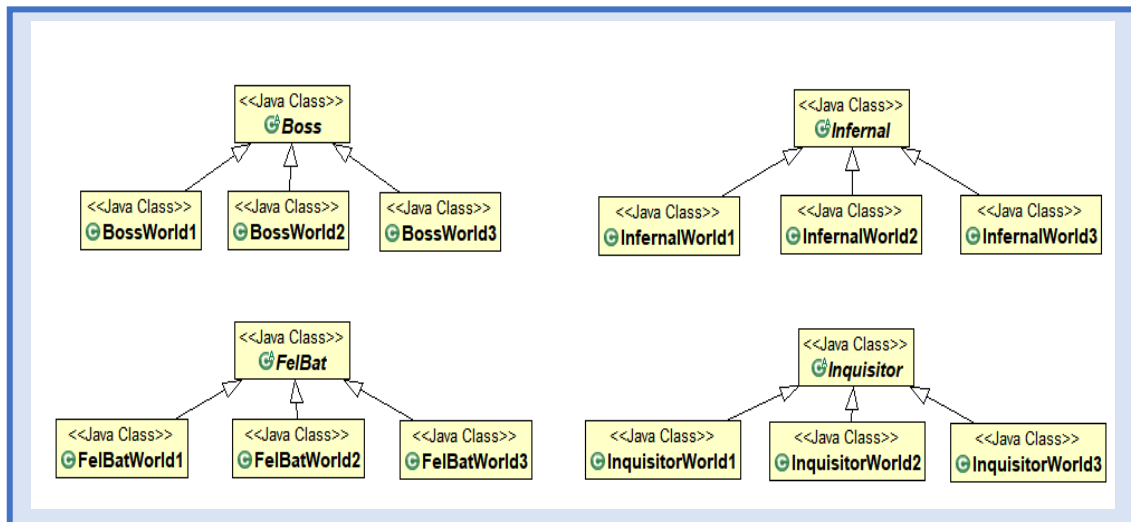
El método createInfernal (ActivityStrategy strategy) devolverá un objeto del tipo InfernalWorld1, InfernalWorld2 o InfernalWorld3; dependiendo del tipo de factoría que se utilice. En la instanciación del objeto se le deberá pasar una estrategia para que el constructor pueda inicializar la estrategia de combate asignada a este tipo de enemigo.

MÉTODO “CREATEENEMY()”

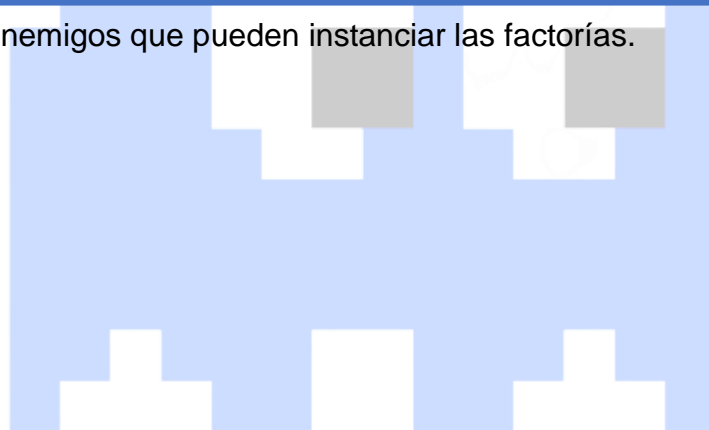
El método createEnemy (ActivityStrategy strategy) llamará aleatoriamente a uno de los anteriores métodos cada vez que se le invoque. Esto permitirá una generación aleatoria de enemigos en base a unas probabilidades definidas en la hoja de personajes correspondiente. Al igual que el resto, le pasará en la llamada la estrategia que cada enemigo decidirá implementar.

MÉTODO “CREATEBOSS()”

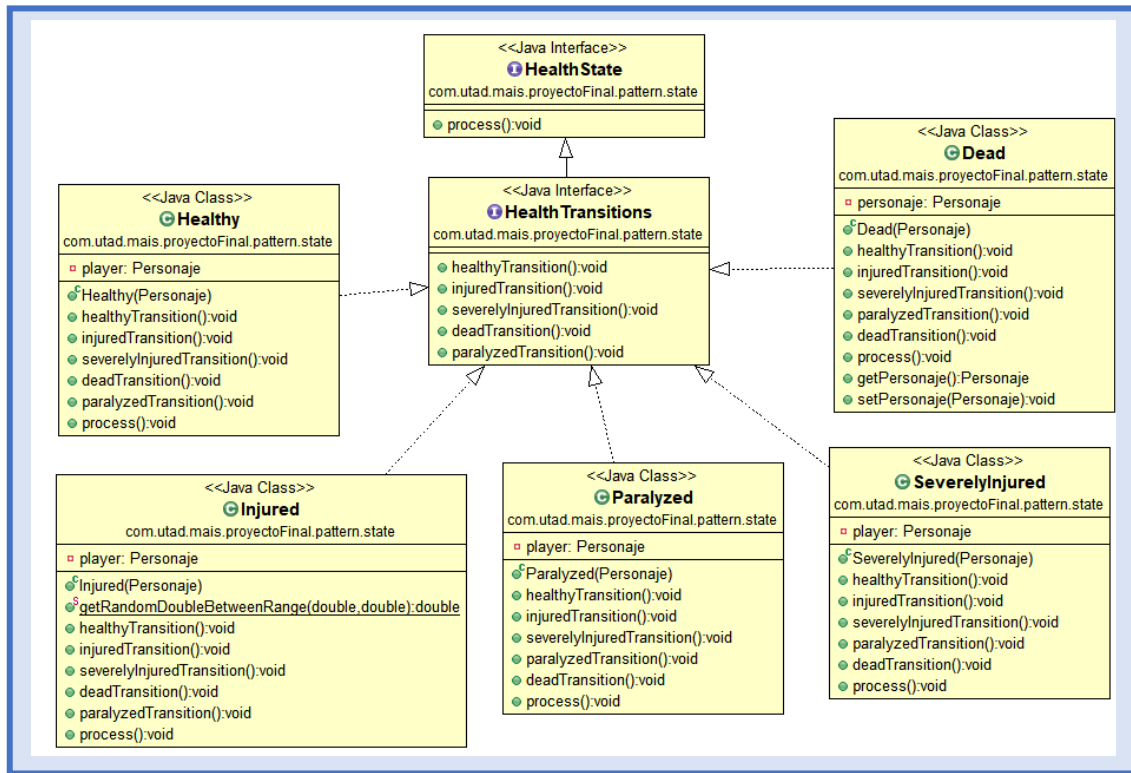
El método createBoss (ActivityStrategy strategy) devolverá un objeto del tipo BossWorld1, BossWorld2 o BossWorld3; dependiendo del tipo de factoría que se utilice. En la instanciación del objeto se le deberá pasar una estrategia para que el constructor pueda inicializar la estrategia de combate asignada a este tipo de enemigo.



Los distintos enemigos que pueden instanciar las factorías.



STATE PATTERN

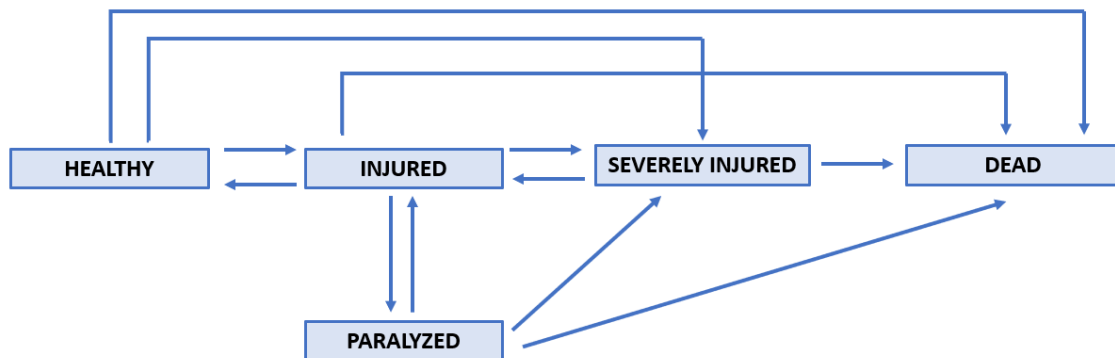


El patrón State nos va a servir para darle algo más de dinamismo al juego. Así, cada estado afectará al jugador de alguna manera (bajará estadísticas o incluso, nos dejará una ronda sin poder llevar a cabo ninguna acción). También nos permitirá saber cuando un enemigo ha muerto para generar el siguiente o cuando el jugador ha muerto, para terminar el juego.

Los estados que tendremos son los siguientes:

- 🧑 Saludable ("Healthy").
- 🧑 Herido ("Injured").
- 🧑 Muy herido ("Severely Injured").
- 🧑 Paralizado (Paralyzed).
- 🧑 Muerto (Dead).

DIAGRAMA DE ESTADOS



EXPLICACIÓN DE LOS DISTINTOS ESTADOS

Del estado Healthy podemos pasar a Injured, Severely Injured o Dead en función de la vida del personaje. Así, si la vida está entre 125 o 50 puntos, el personaje estará Injured. Si baja de 50 puntos estará Severely Injured, y si está por debajo de 1 punto, estará Dead.

Del estado Injured podemos pasar a Healthy, Paralyzed, Severely Injured o Dead. Si estoy Injured, podremos pasar a 4 estados distintos. Tendremos una probabilidad del 50% de entrar en el estado paralizado (solo un turno), en el cual no podremos realizar ninguna acción. Con el otro 50%, en función de nuestras acciones, podremos pasar a Healthy si nuestra vida se recupera por encima de los 125 puntos, quedarnos en el estado Injured, pasar al estado Severely Injured, si nuestra vida baja por debajo de los 50 puntos o Dead si baja por debajo de 1 punto de vida.

Del estado Severely Injured, podremos pasar a Healthy, Injured o Dead en función de los puntos de vida, como se ha visto anteriormente.









En el estado Paralyzed, podremos pasar a Injured, Severely Injured o Dead, de nuevo, en función de nuestros puntos de salud.

Del estado Dead no se podrá pasar a ningún otro estado, puesto que el juego finalizará si somos el jugador, o creará a otro enemigo distinto si es el enemigo el que muere.

Estadísticas que cambian en función del estado: booleano de muerte, booleano de paralizado, fuerza y agilidad.

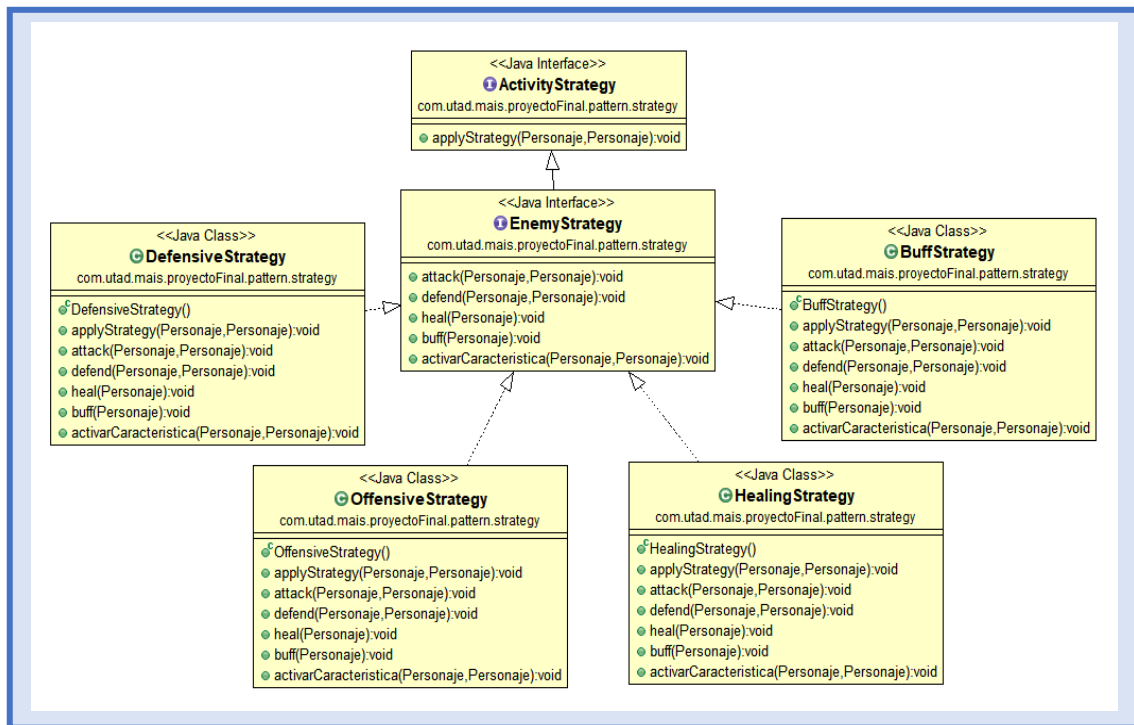
CAMBIOS EN FUNCIÓN DEL ESTADO

Como mencionamos previamente, el estado cambiará algunos de los atributos de los personajes a los que afecte. De esta manera:

-  Si pasamos de Healthy a Injured, se nos reducirá la agilidad en 25 puntos y la fuerza en 50, resultando en un menor daño.
-  Si pasamos de Healthy a Severely Injured, se nos reducirá la agilidad en 50 puntos y la fuerza en 75, resultando en un menor daño que si estamos en Injured.
-  Si pasamos de Injured a Severely Injured, se nos vuelve a reducir la agilidad y la fuerza en 25 puntos. De esta manera, la reducción por pasar de Healthy a Injured y luego a Severely Injured es la misma que de pasar de Healthy a Severely Injured.
-  Si de Injured pasamos a Paralyzed, nuestro personaje tendrá su atributo “paralizado” fijado a true. De esta forma, no podremos realizar ninguna acción.
-  De Injured podremos pasar a Healthy si nuestra salud se recupera llegando a estar en los 125 puntos o más de salud. La reducción de atributos se revertirá.
-  De Paralyzed podremos pasar a Injured, Severely Injured o Dead en función de nuestros puntos de vida, y al cambiar de estado, el atributo “paralizado” se fijará a false, para que el jugador pueda volver a realizar acciones.
-  De Severely Injured, podremos pasar a Injured o Healthy si nos curamos lo suficiente, revirtiendo las penalizaciones a la fuerza y agilidad del personaje.
-  El personaje puede pasar de cualquier estado al Dead, en el cual, dependiendo si es el jugador o el enemigo, actuaremos de formas distintas.

Todo lo que se ha explicado aquí se aplica tanto al personaje jugador como a los personajes enemigos. Como se ha expuesto previamente, si el enemigo pasa a Dead, se generará un nuevo enemigo, pero si es el jugador el que pasa a este estado, se terminará el juego.

STRATEGY PATTERN



Utilizaremos el patrón de estrategia para dotar a los enemigos de “inteligencia”. Así, en función del estado del enemigo, éste actuará de formas distintas.

Distinguimos cuatro estrategias: estrategia ofensiva, estrategia defensiva, estrategia curativa o estrategia de aumento de estadísticas o “buffado”.

EXPLICACIÓN DE LAS DISTINTAS ESTRATEGIAS

Defensiva:

El enemigo trata de bloquear el ataque reduciendo el daño que le infligen en un 30%.

El enemigo tiene un 50% de probabilidad de defenderse, un 30% de atacar a su objetivo, un 10% de probabilidad de curarse y un 10% de activar su característica especial.

En estado defensivo, el daño que se realiza contra el enemigo cuando este se encuentra en dicho estado se decrementa en un 10% adicional.

Ofensiva:

El enemigo trata de atacar sin cesar y se cura con muy poca frecuencia. Tiene un 80% de probabilidad de atacar, un 10% de curarse y un 10% de activar su característica especial.

Curativa:

El enemigo adopta por atacar, bloquear y curarse paulatinamente.

El enemigo tiene una probabilidad de curarse de un 55%, una probabilidad de atacar del 25% y una probabilidad de defenderse del 20%.

Buffos / Aumento de estadísticas:

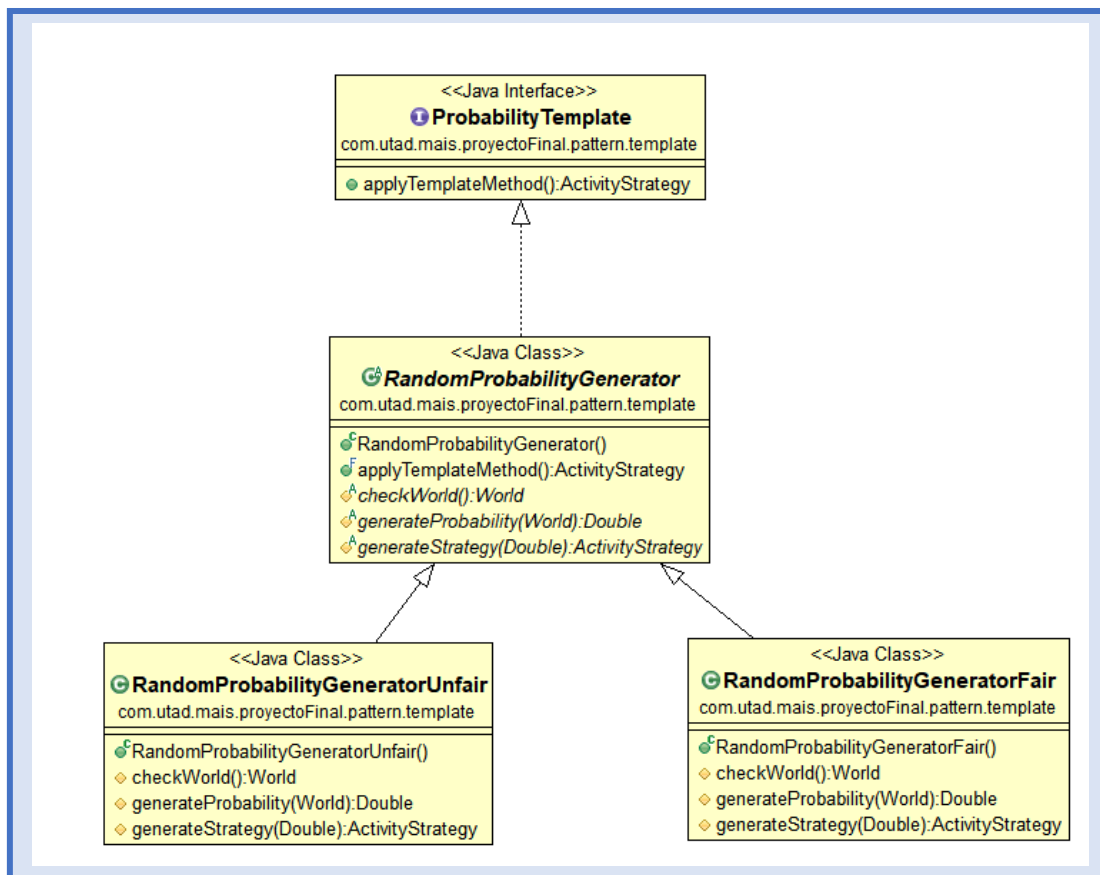
El enemigo opta por aumentar sus características, curarse y atacar.

El enemigo tiene una probabilidad de aumentar sus características de manera aleatoria. Cada vez que el enemigo se buffe, tendrá cierta probabilidad de aumentar una de las siguientes características: puntos de ataque, puntos de agilidad, puntos de intelecto.

La probabilidad de aumentar los puntos de agilidad es del 40%, los puntos de intelecto un 30%, y los de fuerza el 30% restante.

Además, tendrá un 50% de probabilidad de buffarse, un 30% de atacar y un 20% de curarse.

TEMPLATE PATTERN



El objetivo del template pattern es proporcionar dos formas diferentes de instanciar la estrategia que van a utilizar los personajes enemigos. El tipo de estrategia que instancian dependerá del mundo en el que se encuentren y de la probabilidad generada por el template.

De esta manera, podemos ver que el flujo de ejecución del template pattern será el siguiente:

1. Comprobar el mundo actual
2. Establecer un factor de probabilidad en base al mundo en el que nos encontremos.
3. Devolver una estrategia en función del factor de probabilidad y del método `Math.Random()`.

Las dos clases concretas del template son: `RandomProbabilityGeneratorFair` y `RandomProbabilityGeneratorUnfair`.

RANDOM PROBABILITY GENERATOR FAIR

La clase RandomProbabilityGeneratorFair tiene las siguientes probabilidades asociadas.

- 🧛 Mundo 1: 10%
- 🧛 Mundo 2: 15%
- 🧛 Mundo 3: 20%

La probabilidad de devolver cada una de las estrategias viene dado por la siguiente tabla:

| | Mundo 1 | Mundo 2 | Mundo 3 |
|--------------------|---------|---------|---------|
| Defensive Strategy | 20% | 15% | 10% |
| Healing Strategy | 20% | 20% | 20% |
| Offensive Strategy | 20% | 20% | 20% |
| Buff Strategy | 40% | 45% | 50% |

RANDOM PROBABILITY GENERATOR UNFAIR

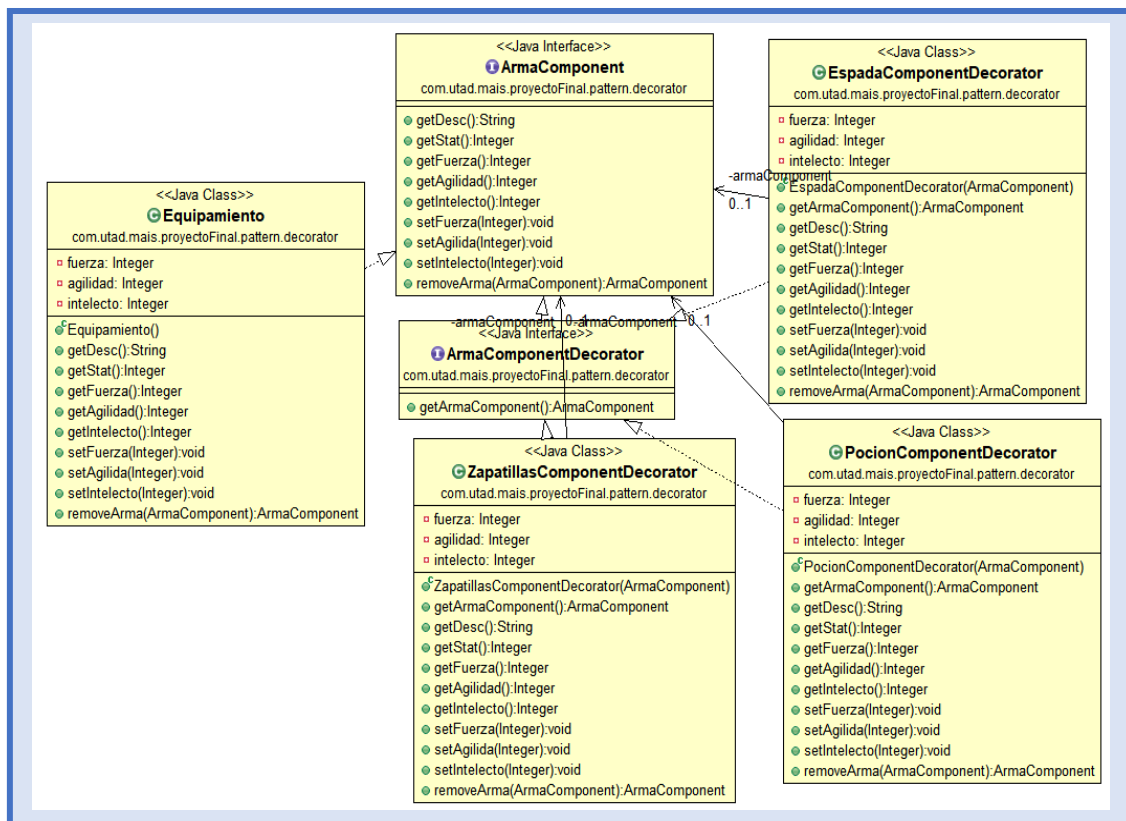
La clase RandomProbabilityGeneratorFair tiene por factores de probabilidad los siguientes:

- 👾 Mundo 1: 20%
- 👾 Mundo 2: 30%
- 👾 Mundo 3: 40%

La probabilidad de devolver cada una de las estrategias viene dado por la siguiente tabla:

| | Mundo 1 | Mundo 2 | Mundo 3 |
|--------------------|---------|---------|---------|
| Buff Strategy | 10% | 0% | 0% |
| Defensive Strategy | 20% | 20% | 10% |
| Healing Strategy | 20% | 20% | 20% |
| Offensive Strategy | 50% | 60% | 70% |

DECORATOR PATTERN

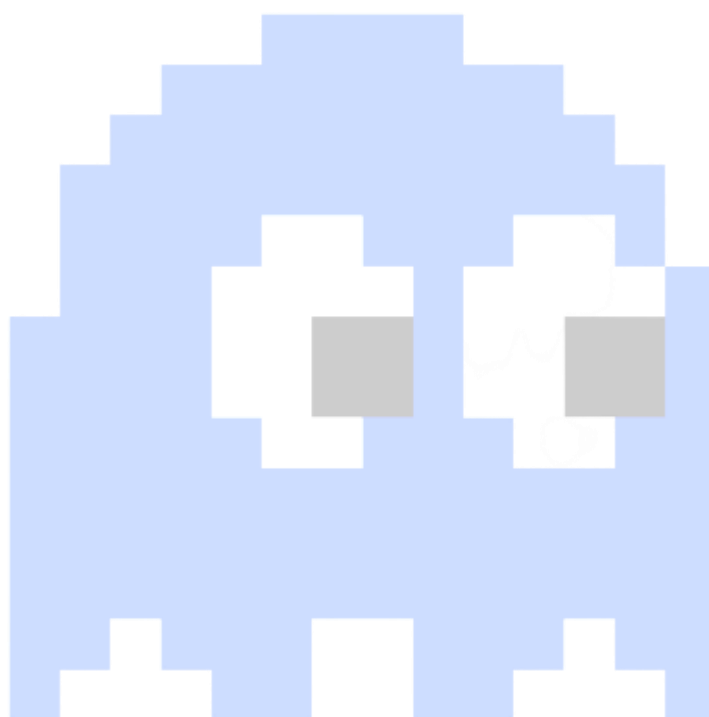


Debido a que nuestro juego se basa en distintos personajes con estadísticas, ya sea el jugador o los enemigos, vamos a implementar un sistema de aumento de éstas. Ya que los enemigos pueden aumentar sus estadísticas si se crean con una estrategia determinada, es justo que el jugador también pueda aumentar las suyas. Esto permite también que el juego no se vuelva difícil hasta el punto de ser injusto, porque si permitimos que los enemigos aumenten sus estadísticas con la estrategia “buffo” y cada vez que pasamos del mundo 3, llegará un punto en el que, si nosotros no aumentamos también nuestras estadísticas, no podremos hacer nada.

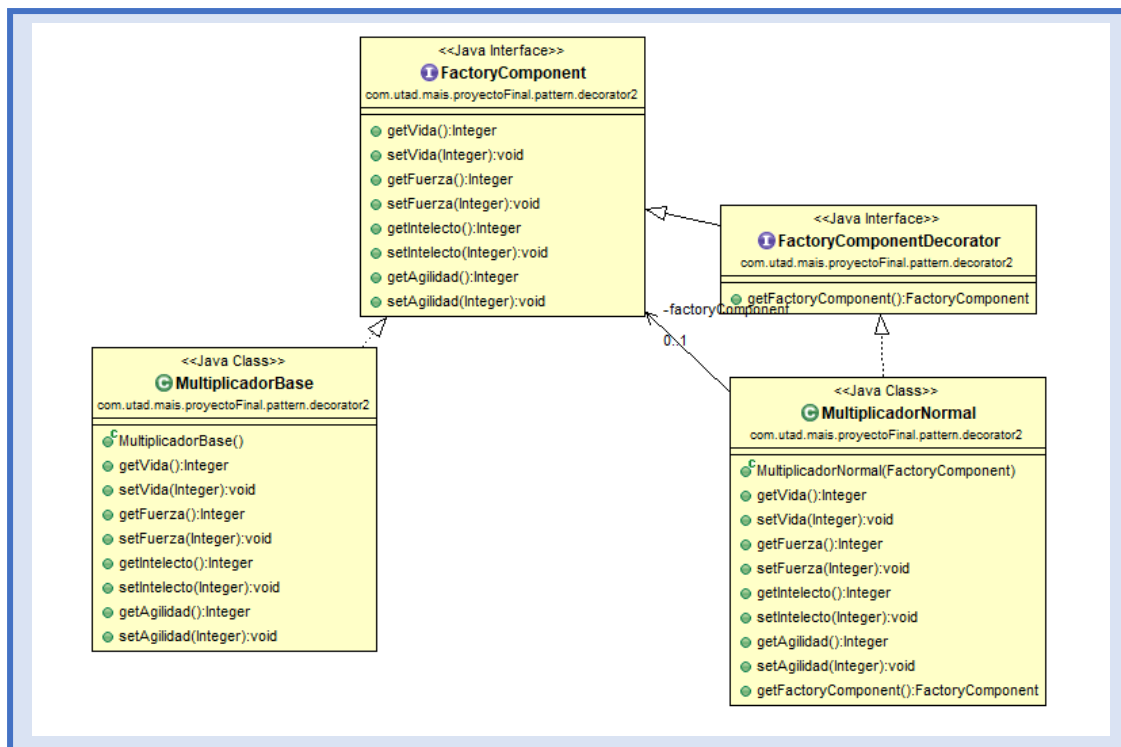
Para ello utilizaremos el patrón decorador. Tendremos tres decoradores, la espada, la poción y las botas. Cada una afectará a una estadística distinta. La espada nos aumentará la fuerza, las botas la agilidad y la poción el intelecto. Todo ello afectará al daño que causamos a los enemigos y a la cantidad de vida que recuperamos al curarnos. Los números exactos son los siguientes:

- 🧙 La espada aportará 150 puntos adicionales de fuerza.
- 🧙 La poción aportará 50 puntos adicionales de intelecto.
- 🧙 Las zapatillas aportarán 30 puntos adicionales de agilidad.

La forma de actualizar dichas estadísticas será mediante ir devolviendo la estadística a la que afectan, y por cuántos puntos se subirá la estadística concreta.



DECORATOR PATTERN II



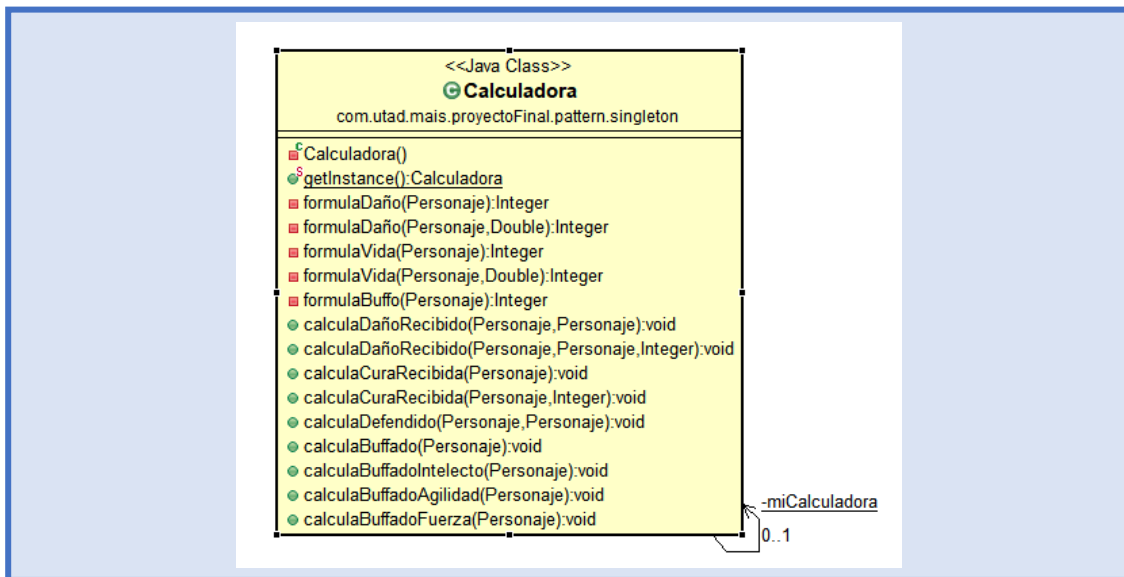
Utilizaremos también el patrón decorador en las factorías abstractas.

El patrón decorador va a servir a la hora de modificar la dificultad del juego siguiendo una temática de arcade.

El objetivo que se plantea conseguir con el patrón es incrementar la dificultad del juego y permitir al usuario jugar de nuevo cada vez que se haya pasado los 3 mundos predefinidos por defecto. Esta nueva partida tendrá en cuenta sus registros anteriores, le quitará todo el equipamiento que tenga el personaje e incrementará las estadísticas por defecto que tengan los enemigos. Como estas estadísticas parten de una base por defecto, lo que se obtiene aplicando el patrón es que la instanciación difiera y se les sume una bonificación a dichas estadísticas. El objeto base del decorador no aportará ningún tipo de bonificación (multiplicador inicial), mientras que el objeto decorador aportará una bonificación de 10 puntos en todas las estadísticas.

Gracias a esto, se consigue que el juego suponga un reto mayor para los jugadores, se permite una mayor escalabilidad sin modificar la forma de la instanciación de personajes y se consigue incrementar la dificultad o reducirla (si fuera oportuno) añadiendo decoradores.

SINGLETON PATTERN



La clase calculadora implementará el patrón singleton y será la encargada de realizar los cálculos oportunos para reducir o incrementar las estadísticas de todos los personajes.

Por una parte, la clase calculadora necesita el patrón singleton para asegurarse de que solo haya una instancia de dicha clase que realice los cálculos. No tiene sentido tener varias instancias de la clase calculadora, ya que todas realizan la misma función y no varían en ninguno de sus aspectos. Además, la calculadora debe ser accesible desde cualquier parte del código, ya que esto permite una escalabilidad mayor.

Por otra parte, la clase calculadora está compuesta por las fórmulas que calculan el daño, la sanación o el incremento de estadísticas realizado, en base a las estadísticas del personaje atacante. Como a lo largo del juego se pueden dar situaciones en las que se quiera modificar dichas fórmulas levemente, dependiendo de alguna situación concreta, las fórmulas presentarán un parámetro extra llamado multiplicador. Este parámetro permitirá modificar las fórmulas levemente. Además, los métodos fórmulas serán privados, ya que no se debe permitir que ninguna otra clase pueda modificarlos salvo la propia calculadora por lo anteriormente mencionado.

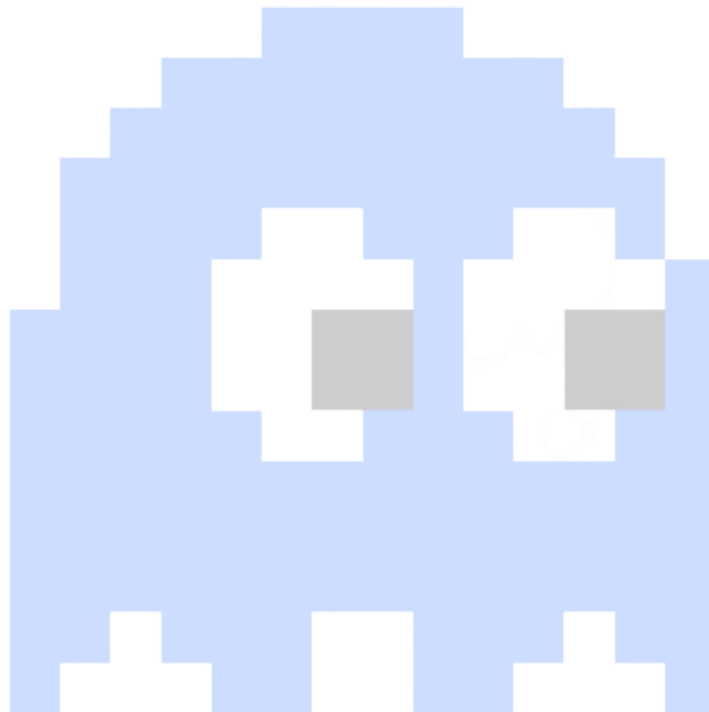
Para la realización de los ataques, sanaciones e incrementos se tendrán jerarquías simples de métodos que se llamarán unos a los otros y estos finalmente a sus fórmulas correspondientes. Todos los personajes serán quienes llamen a estos métodos públicos para realizar su acción correspondiente.

Por último, se debe mencionar que por motivos de simplicidad en el código se ha preferido que sean los métodos públicos quienes informen de las acciones

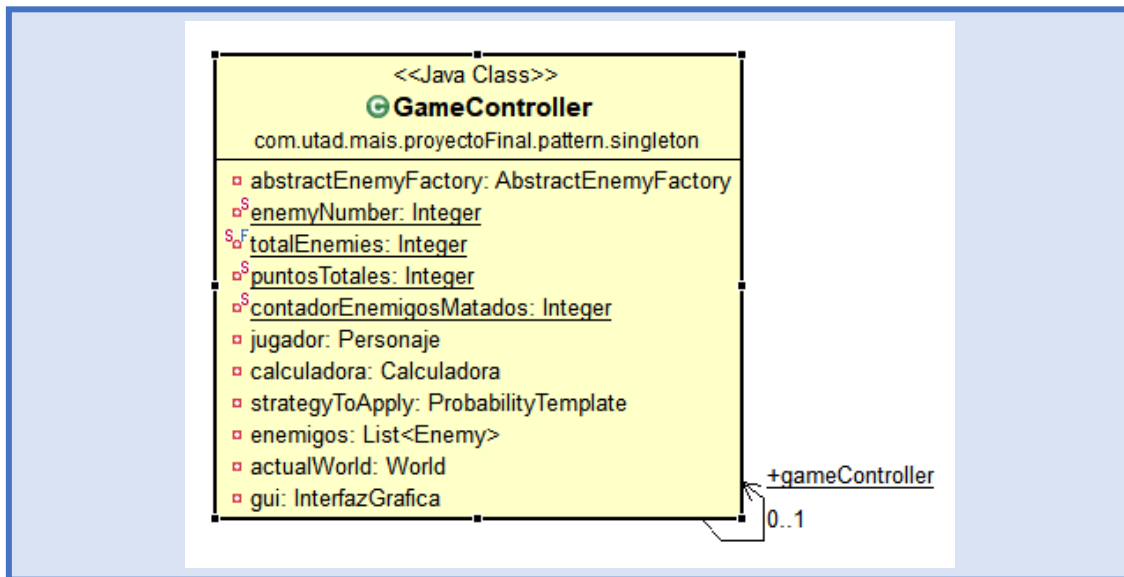
realizadas con todo lo que estas conllevan. Este comportamiento podría verse mejorado utilizando un observer que notificará cada vez que se modifican los atributos, pero por motivos de simplicidad se ha preferido unificarlo en la clase calculadora.

Las situaciones en las que se ha visto necesario modificar las fórmulas son las siguientes:

- 👾 El personaje jugador ataca a un enemigo con actitud defensiva. Esto hace que el multiplicador base del daño realizado se reduzca a la mitad.
- 👾 Cualquier personaje que tenga menos de 20 puntos de vida ve su sanación incrementada en un 30%.



GAME CONTROLLER



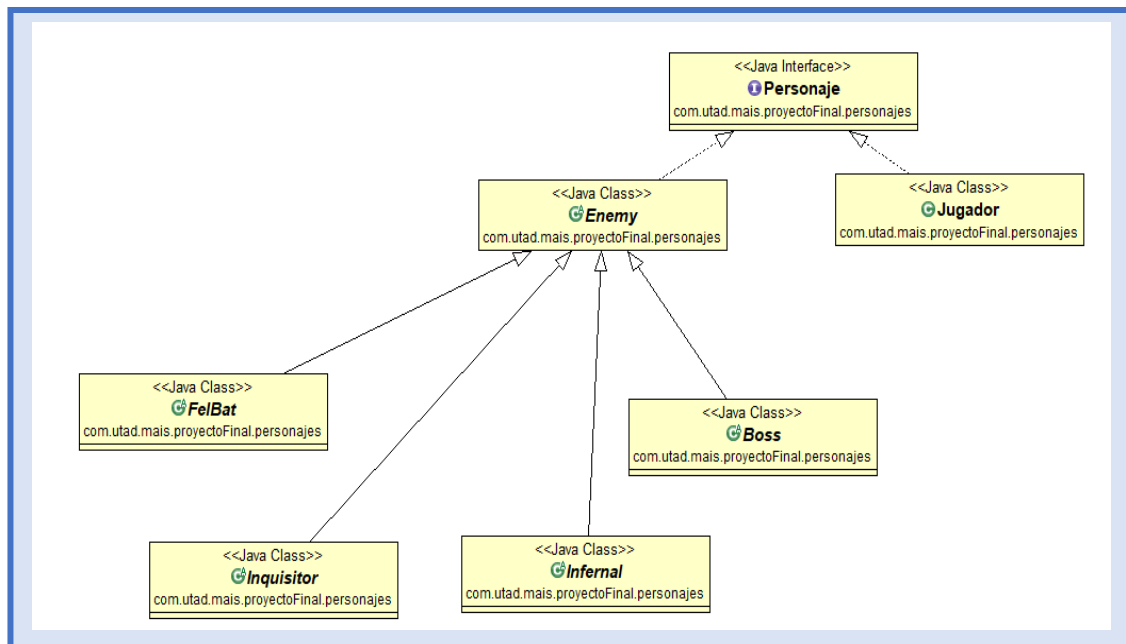
El Game Controller seguirá el patrón de diseño Singleton.

Se va a explicar en qué consiste el bucle principal de juego.

El método `Play()` hace lo siguiente: generamos a los enemigos del mundo en el que está el jugador. Tras esto, comprobaremos que hay aún enemigos restantes o que el mundo no sea el último. Si esta comprobación es correcta, lo primero que haremos será comprobar el mundo. Esto es para que si hemos derrotado a todos los enemigos del mundo 1 o del mundo 2, volvamos a generar enemigos. Tras esto, el enemigo realizará acción en función de la estrategia, si no está paralizado, y comprobaremos el estado del jugador por si el enemigo lo ha matado. Si el jugador sigue vivo, y no está paralizado, podrá escoger cuatro opciones: atacar, curarse, equiparse equipamiento o defenderse. Tras la acción comprobaremos el estado del enemigo, y si está muerto, pasaremos al siguiente enemigo. Este bucle se ejecutará de continuo a no ser que muramos. Otra opción de salir del bucle es si cuando nos pasemos el mundo 3, no queremos continuar jugando. En ese caso, mostraremos un mensaje de victoria con los puntos que ha obtenido el jugador. En caso de querer repetir, haremos una llamada recursiva al `Play()`, para volver a empezar el juego, solo que esta vez con un incremento de dificultad.

Para hacer todo esto, usaremos métodos propios del `GameController` para tener el método `Play()` lo más sencillo posible.

PERSONAJES



La interfaz “Personaje” es común tanto a los distintos enemigos que tenemos en el juego como al jugador. “Personaje” tiene métodos de obtención de las distintas estadísticas que tienen todos los personajes y de sus estados.

La clase abstracta “Enemy” implementa la interfaz “Enemy”, y a su vez, los distintos enemigos que se instanciarán en las factorías heredan de “Enemy”. Los métodos básicos están dentro de la clase abstracta, que incluyen todos los getters y setters de las estadísticas y estados de los enemigos, y de las estrategias, así como las características que cambiarán el comportamiento del enemigo.

A su vez, tendremos una clase abstracta por cada tipo de enemigo, en nuestro caso “FelBat”, “Inquisitor”, “Infernal” y “Boss”. En estas clases abstractas definimos las características básicas de los enemigos. Cada una de estas clases abstractas tendrá una especialización por mundo. Así, podemos instanciar enemigos muy similares, cambiando solo las estadísticas del enemigo mediante el factor de complejidad del mundo, lo que nos facilita mucho la escalabilidad.

La clase “Jugador” sin embargo, no hereda de nada, simplemente implementa la interfaz “Personaje”. Esta clase implementará todos los métodos relativos a obtener y fijar sus estadísticas y estados, pero también implementa el decorador de equipamiento, pues como mencionamos en el patrón decorador, solo el jugador puede equiparse objetos. Esta es una de las diferencias principales del jugador con los enemigos.

DESCRIPCIÓN DE LOS ENEMIGOS

Infernales

Puntos de fuerza: 600.0.

Puntos de agilidad: 10.

Puntos de intelecto: 0.

Puntos de resistencia física: 250.

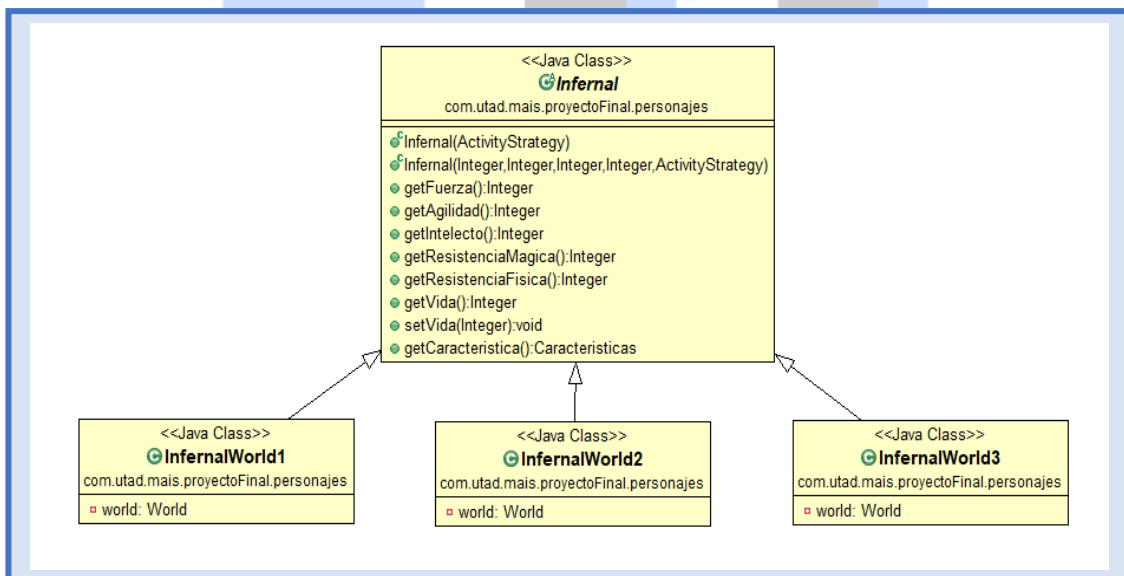
Puntos de resistencia mágica: 0.

Puntos de vida: 800.

Característica: Tamaño colosal. Aumenta el factor fuerza en 0.01%.

Probabilidad de aparición por mundos:

- 👹 Mundo 1: 0%
- 👹 Mundo 2: 0%
- 👹 Mundo 3: 20%



Inquisidores

Puntos de fuerza: 10.0.

Puntos de agilidad: 20.0.

Puntos de intelecto: 300.0.

Puntos de resistencia física: 50.0.

Puntos de resistencia mágica: 300.0.

Puntos de vida: 200.

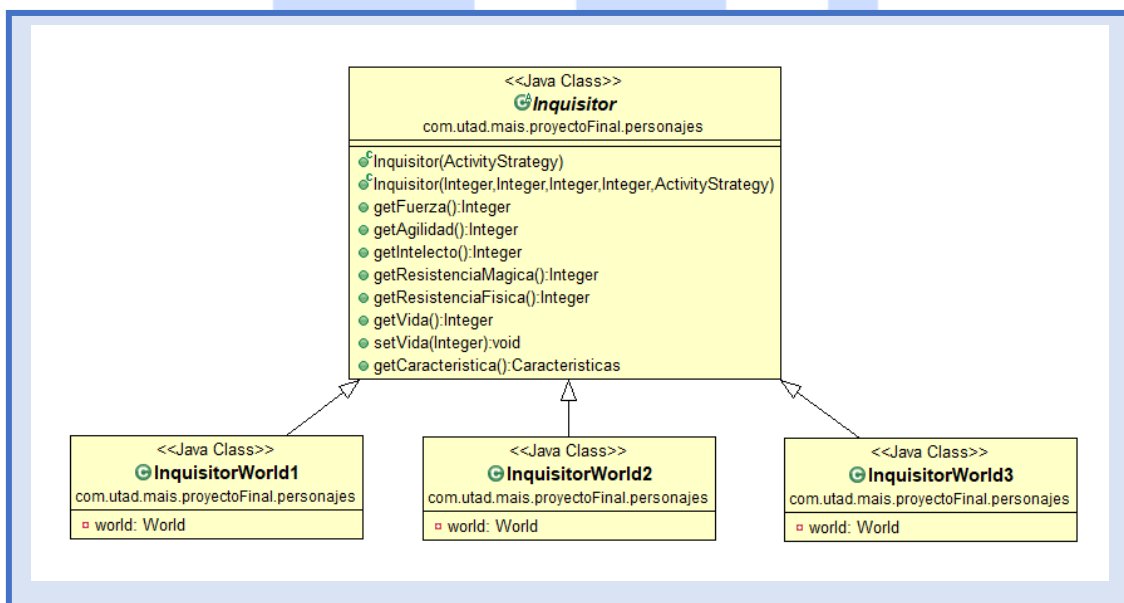
Característica: Voluntad del Inquisidor. Aumenta el factor intelecto en 0.01%.

Probabilidad de aparición por mundos:

👾 Mundo 1: 30%

👾 Mundo 2: 50%

👾 Mundo 3: 50%



Murciélagos Viles

Puntos de fuerza: 100.0.

Puntos de agilidad: 200.0.

Puntos de intelecto: 0.0.

Puntos de resistencia física: 150.

Puntos de resistencia mágica: 0.

Puntos de vida: 2 00.

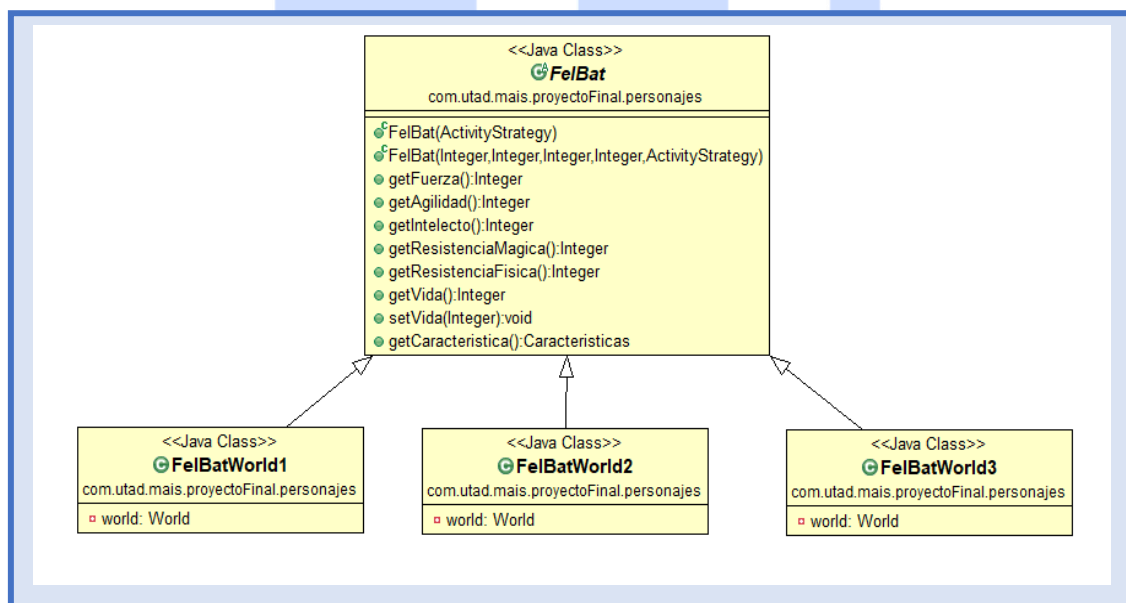
Característica: Chirrido Aullante. Aumenta el factor agilidad en 0.01%.

Probabilidad de aparición por mundos:

🦇 Mundo 1: 70%

🦇 Mundo 2: 50%

🦇 Mundo 3: 30%



Jefes

Puntos de fuerza: 200

Puntos de agilidad: 200

Puntos de intelecto: 200

Puntos de vida: 1200

Característica: Boss Factor: Se buffa, ataca y se cura en un turno. Los bosses siempre aparecen al final de cada mundo.

Los bosses otorgan 100 puntos en vez de 10 como el resto de los enemigos.

