Resumen 2: Proyecto Final Reality

Nombre: Lucas Oyarzun Mendez

RUT: 20.214.801-8

Enlace a repositorio: https://github.com/CC3002-Metodologias/final-reality-LucasOyarzun

Último Pull Request:

https://github.com/CC3002-Metodologias/final-reality-LucasOyarzun/pull/11

En esta entrega, separada en entrega parcial 3, 4 y 5, se corrigen los problemas de Diseño de la primera Tarea, se añaden los métodos de equipar armas y atacar, y finalmente se añade un controlador.

Los métodos de equipar armas y atacar se implementan con Double Dispatch, donde el personaje intenta equipar un arma, el arma responde llamando al método del personaje que la equipe. Si el personaje la puede equipar, la equipa, si no, no ocurre nada.

Para el método de atacar es algo parecido, un ICharacter ataca a otro ICharacter, el que es atacado responde llamando al método necesario del personaje que lo atacó, si ambos son personajes del jugador, no pasa nada, si son oponentes, se calcula el daño, se reduce la vida del atacado y se acaba el turno del personaje.

El controlador puede crear y asignar objetos del modelo, saber cuáles son los personajes del jugador y sus respectivos datos, igualmente para los personajes del computador(Enemigos), puede manejar el invetario del jugador, equipar armas y hacer que un personaje ataque a otro.

El controlador además sabe cuando un personaje comienza y termina su turno, para esto, toma un personaje de la cola(Empieza su turno), le permite atacar a otro personaje(O equipar armas antes de atacar, esto se implementará más adelante), si un personaje ataca termina su turno, se espera un tiempo y se saca otro personaje de la cola.

Y la última funcionalidad del controlador implementada por ahora es la de saber si un jugador perdió o ganó el juego. Cada vez que un personaje muere, se le avisa al controlador, este, usa Double Dispatch para saber si es el Player o el Computer el que perdió un personaje, y luego le ordena que lo elimine de su lista de personajes. Si el Player o el Computer llegan a tener la lista sin personajes restantes, entonces se gana/pierde el juego.

Para los anuncios de ataque, inicio de turno, fin de turno, victoria o derrota, por ahora se usa print. Ya que no se ha implementado un método Main() (Dice que aún no se implemente la interacción con el jugador), si se hace quiet testing, no quedan ocasiones para ver cómo funcionan los anuncios del controlador, por esto se decidió dejar los prints en los tests. De no ser así, se habría implementado quiet testing(más adelante, si se implementa la interacción con el jugador a través de un Main() o algo parecido, se silenciará a los prints durante los tests).

Se crean tests para el 100% del código, cumpliendo en que no existan tests vacíos, se documenta todo el programa y se modifica el archivo README.md para indicar las funcionalidades del proyecto.

Diagrama del package character. Con sus respectivos métodos y tests. (Imagen pequeña pero con buena calidad, hacer zoom si es necesario).

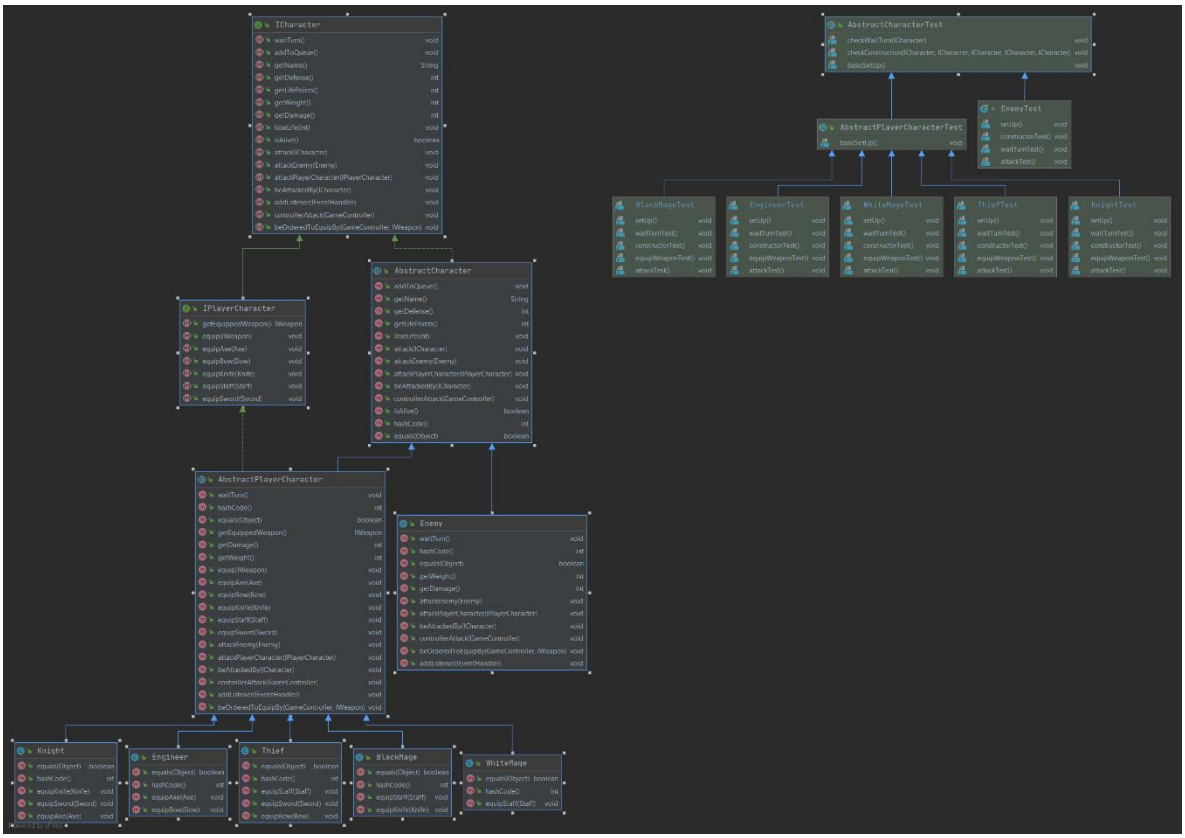

Diagrama del package weapon. Con sus respectivos métodos y tests. (Imagen pequeña pero con buena calidad, hacer zoom si es necesario).

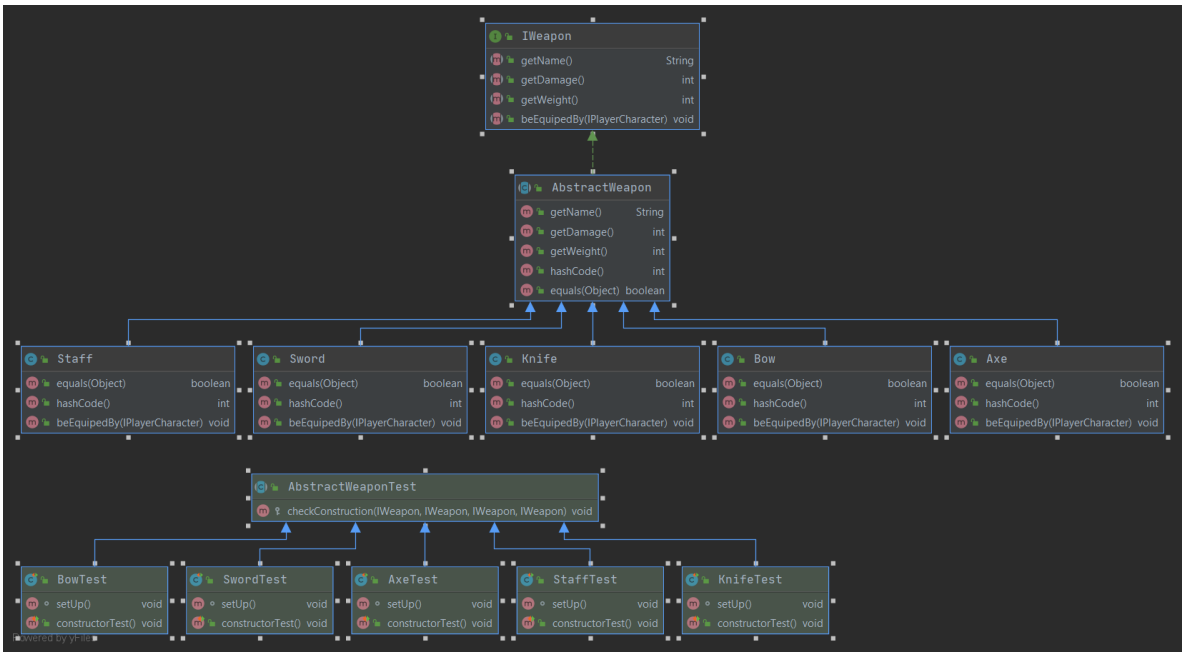

Diagrama del package controller y de las clases Player y Computer. Con sus respectivos métodos y tests. (Imagen pequeña pero con buena calidad, hacer zoom si es necesario).