**Programación Orientada a Objetos**

**Práctica 1**

**Grupo 6**

**Brayan David Caballero Fernández**

**John Mauricio Mesa Echeverri**

**Eider Alejandro Peña Dagua**

**Santiago Rivera Mejía**

**CALABOZOS EN LA NACHO**

**Introducción**

La actual aplicación surge con la idea de platear un demo jugable que rememora a los clásicos juegos de rol de texto, para ello implementaremos los conceptos de la programación orientada a objetos vistos en clase. Se desarrollará la narración en un clásico ecosistema de fantasía medieval, inspirado en el juego *Calabozos & Dragones*.

Los RPG (Role-Playing Game) se caracterizan por hacer que el jugador asuma el rol de un personaje dentro de una historia. El personaje tiene una serie de características que permiten desarrollar las mecánicas del juego, como combatir, lanzar hechizos, levantar objetos pesados y un ilimitado conjunto de opciones. Estas características describen al personaje tanto física como psicológicamente. Existe una narrativa que contextualiza al jugador y le impone retos, estos retos han de ser superados por el jugador mediante sus elecciones y un componente de azar, que es otorgado por el lanzamiento de dados para determinar el éxito o no de sus acciones.

**Detalle y construcción:**

La aplicación surge como en base de poder construir un demo de un juego que solo tuviera la interacción de un usuario, Luego de diseñar a partir de esa idea iniciamos por identificar las clases, atributos y métodos que nos podrían servir, al igual que se llegó a un acuerdo de que funciones debería cumplir este demo del juego, esto se logro al realizar un investigación acerca y revisión de diferentes juego de rol, en esta encontramos que calabozos y dragones se acercaba mucho a lo que queríamos realizar.

Después de esto se empezó la realización, y algunas cosas se tuvieron que modificar para lograr obtener los que se quería, no solo se hicieron modificaciones también se agregaron nuevos métodos y atributos que eran necesarios para el correcto funcionamiento de la aplicación. Esto se dio a partir de aquí dividimos el trabajo, que, aunque al principio todo está demasiado confuso, la interacción con lo programado por los miembros del equipo, logra darnos una visión global.

Como primera parte decidimos implementar las clases planteadas con sus respectivos atributos, constructor y métodos, esto nos da un punto de partida para ver que clases podrían estar en un mismo paquete y cuales podrían ser heredar.

Así se crean las clases **Player, NPC, Clase, Narrador, Raza, Inventario, Arma y Armadura** con sus respectivos atributos y constructores, al realizar esto se evidencio una clara relación de herencia que podía surgir por parte de la clase NPC hacia la clase Player. También se logra la separación de las clases por paquetes esto al ver que estas podían tener una mayor relación entre ellas ayudándonos con el encapsulamiento y manejo de los atributos. Luego de todo esto se nota que la clase Raza comparte atributos y algunos métodos con NPC esto hace que también plantemos esa relación, con un cambio en su nombre paso de ser **Raza** a llamarse **Enemigo** esto con el fin de poder reducir un poco el dominio y alcance del desarrollo. Las clases **NPC**, **Player** y **Enemigo** se ponen un mismo paquete “**pls**”, junto con la clase **Clase.** Esto da una perspectiva mejor de lo que se puede diseñar con las habilidades aprendidas hasta el momento en el curso. Así se decide crear nuevas clases, como lo son **Tienda** y **Poción,** esto con la finalidad de incluir funcionalidades que se puedan interactuar fácilmente entre ellas. Estas dos clases se ponen en un mismo paquete llamado **Loadout** junto a **Inventario, Arma** y **Armadura.** La clase **Narración** esta en el paquete mecánicas.

Para el desarrollo de la clase **NPC** debimos tener en cuentaya que será la clase padre para otras dos clases, y será abstracta al igual que debemos serializarla, las clases **Player** y **Enemigo,** las clases del paquete pls, sufrieron varios cambios ya que son fundamentales para la aplicación.

Se implementó una clase **Narrador** la cual funciona como eje central del juego, siendo esta clase la que coordina todos los elementos dentro del juego: tanto las mecánicas como los actores involucrados. Las mecánicas del juego se resumen en la presentación de textos que cuentan una historia, desde la clase Narrador se hace un llamado a la clase InterfazUsuario que se encuentra en el paquete uiMain, la clase Narrador cuenta con una variable de tipo entero, llamada escenaActual, que identifica a la escena particular que se ha de narrar en ese momento, InterfazUsuario crea objetos de cada escena y las almacena en un ArrayList, cada objeto de escena se compone de una narrativa y de las opciones que se ofrecerán al usuario al final de dicha narración. La instancia de la clase Narrador invocará a la clase InterfazUsuario y, según sea el valor de la variable escenaActual, este retornará por pantalla la escena correspondiente y sus respectivas opciones. A continuación, se le permite al usuario ingresar por teclado la opción que desee y el ciclo se repite, a menos que ocurra un combate o el jugador decida hacer uso de las otras opciones que ofrece el juego, tales como “guardar partida”, “ver hoja de personaje”, entre otras, a las cuales se accede mediante la inserción de comandos, los cuales se pueden ver usando el comando ‘help’.

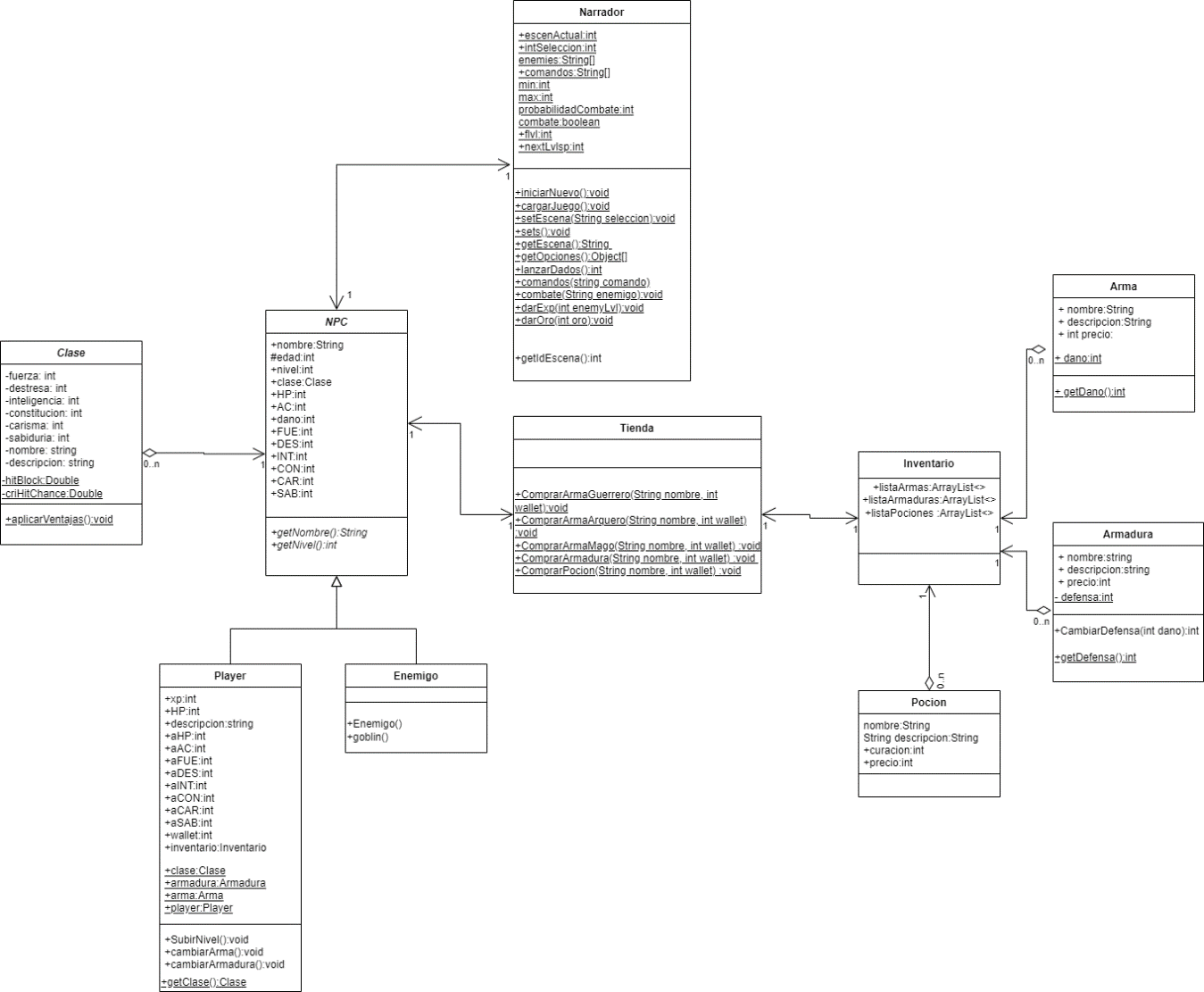
Para el desarrollo de la clase **Tienda** se tenía pensadohacer varios métodos con switch cases para que al momento de la compra retornaran Arma, Armadura Y pociones, sin embargo, también necesitábamos cumplir la condición de que si el jugador no tenía dinero suficiente entonces retornada un mensaje indicándoselo.

Para la clase **Arma** la pensamos simplemente como una modificación del atributo fuerza del jugador dependiendo del arma en cuestión, para que el daño total al enemigo sea el total de (FuerzaJudador + DañoArma)

Para la clase **Armadura,** pensamos en implementar las armaduras como un equipamiento que aumentara la defensa del jugador dependiendo del tipo de armadura en cuestión, pero que se fuera desgastando cada vez que recibe daño por lo que implementamos un método llamado “cambiarDefensa” el cual recibía el daño hecho por el enemigo al jugador y se lo restaba al atributo defensa de la armadura, para después retornar este valor actualizado.

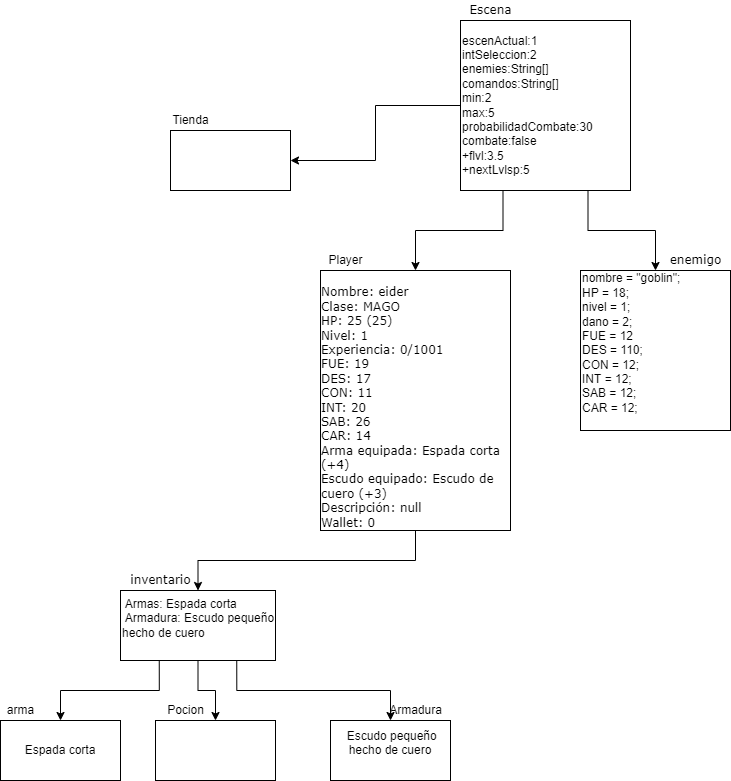
**Diagrama de clases y objetos del sistema:**

**Diagrama de clases**



*Link para ir al diagrama en drive:* [*https://drive.google.com/file/d/1dqgntypQ5ILRPiHngcD6vhLPK2f3BUjK/view?usp=sharing*](https://drive.google.com/file/d/1dqgntypQ5ILRPiHngcD6vhLPK2f3BUjK/view?usp=sharing)

**Diagrama de objetos:**



*Link para ir al diagrama en drive:*

*https://drive.google.com/file/d/1eCi-4PJOAbT4ENhqDnVrh0e6MrFdn\_oQ/view?usp=sharing*

**Implementación de características de programación orientada a objetos:**

**Clases Abstracta y Métodos Abstractos:** la clase abstracta es NPC la cual tiene dos métodos abstractos, getNombre y getNivel.

**Herencia:** la podemos evidenciar en las clases Player y Enemigo que redan de NPC

**Ligadura dinámica:** asociadas al modelo lógico de la aplicación.

**Atributos de clase y métodos de clase:** Atributos de clase tenemos a clase Clase en Player ya que el jugador elige un personaje con un clase definida, algunos métodos de clase ejemplo: goblin()

**Uso de constante:** listas de comandos son constantes, sirven para ayudarnos a interactuar más fácil en la consola

**Encapsulamiento:** (private, protected y public):

**Los siguientes conceptos asociados a la POO:**

**sobrecarga de métodos y constructores:**player, NPC tiene sobrecarga de métodos, linea 22, get sobrecarga de metros getnombre y get nivel en player .

**Manejo de referencias en** los constructores de casi todas las clases.

**Implementación de un caso de enumeración:**Clase, para definir las clases por defecto del juego ya que son constantes.

**Funcionalidades:**

* **Inventario:** el jugador puede modificar las armaduras, armas actuales, seleccionar objetos.
* **Narración:** Cumple la función que cumple un jugador normalmente en el juego de rol, al cual se le encarga guiar al resto de jugadores a través de distintos escenarios mediante la narración de situaciones. En este caso solo guiara a un jugador. Este también es el encargado de ejecutar las reglas de juego, como por ejemplo el desarrollo de un combate.
* **Combate:** enfrentamiento entre el jugador y un enemigo, esta funcionalidad nos permite realizar subfunciones como atacar, ver posición o rendirse.

El combate es gestionado dentro de la clase Narrador, cuando Narrador determina que habrá un combate, determina quienes participarán de dicha batalla (siempre será el personaje del jugador en contra de un rival controlado por el programa), el combate se da por turnos, se compara el atributo DES (destreza) de ambos luchadores , y aquel personaje cuya DES sea mayor actuará primero, durante el combate el jugador dispone de comandos especiales de batalla, como “atacar”, “pocion” y “escapar”. Cuando hay un ataque de cualquiera de los dos contendientes se determina si este ataque es exitoso o no lanzando un dado de 20 caras (función lanzarDados) y si el resultado es menor o igual al número obtenido mediante la siguiente fórmula, el ataque es exitoso y se procede a calcular el daño.

objetivo = 10 + nivel del atacante + clase de armadura del atacado - armadura del atacado

El daño se resta del atributo HP del atacado y es un valor correspondiente al atributo “dano” del arma usada por el atacante.

Cuando los “HP” (Hit points - puntos de vida) de alguno de los contendientes llega a 0, se da por terminado el combate y resulta victorioso quien conserve aún puntos de vida. Si el contendiente victorioso es el jugador, se le otorgan puntos de experiencia que le permitirán subir de nivel y mejorar sus atributos, también se le conceden monedas de oro con las cuales podrá adquirir en la tienda nuevas y mejores armas y armaduras. Estas recompensas se dan en relación al nivel del enemigo derrotado.

Se determinan los involucrados en el combate. Siempre sería el PJ vs una herencia de NPC.

Se decide quien tiene el primer turno sigue quien tenga un puntaje de destreza mayor entre los contendientes

En el turno del jugador, este podría: Atacar con arma principal, atacar con hechizos (habilidad especial), usar objeto del inventario.

En el turno del NPC este atacaría y usaría una habilidad especial cada x turnos

Cuando se ataca se determina si el golpe es exitoso o no de la siguiente manera: objetivo = 10 + nivel del atacante + clase de armadura del atacado - armadura del atacado, luego se lanza 1 dado de 20 caras,

- si el resultado es menor o igual al número obtenido de la sumatoria anterior, el golpe es exitoso y se procede a determinar el daño realizado.

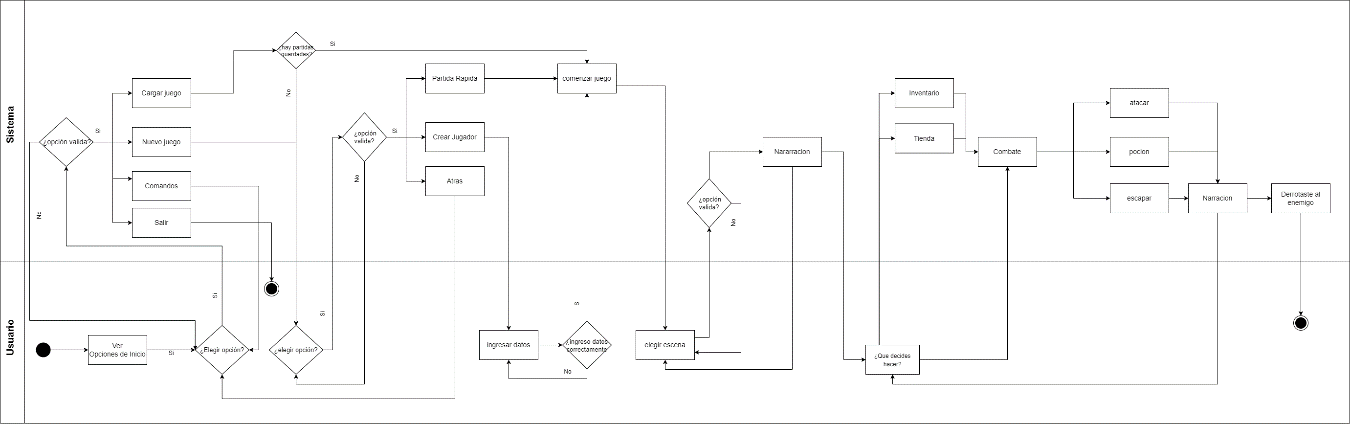
- El daño realizado es igual al daño del arma

- Luego de calcular el daño realizado se termina el turno y comienza el

- El combate termina cuando uno de los combatientes obtiene HP menor o igual

* **Tienda:** el jugador tendrá una tienda en donde podrá adquirir o vender artículos de tipo arma, poción y armadura. Esto se hace con el fin de hacer más dinámico el juego ya que cada vez que se derrote a un enemigo, se obtendrá oro que podrá ser utilizado para comprar e interactuar en la tienda
* **Crear personaje:** se hace con el fin de que el usuario ingrese personalice más el juego.

**DIAGRAMA DE PROCESOS**



*Link para ir al diagrama en drive:*

[**https://drive.google.com/file/d/1m7HXL29KbyctKc\_HNPGza1hZalffI1uA/view?usp=sharing**](https://drive.google.com/file/d/1m7HXL29KbyctKc_HNPGza1hZalffI1uA/view?usp=sharing)

**Manual de usuario**:

Antes de correr la aplicación debemos tener en cuenta algunos comandos numéricos como 1, 2… que se usaran para elegir opciones de narrativa. Y tendremos otros comandos de escritura que nos ayudaran en el juego. Los comando son \* solo se usan en combate.

| **Comando** | **Descripción** |
| --- | --- |
| help | Muestra la lista de comandos disponibles |
| save | Guarda el juego en la escena y con la configuración actual |
| hoja | Muestra la hoja de vida del personaje |
| atacar\* | Atacar al rival |
| pocion\* | El jugador puede usar una poción de recuperación de vida |
| escapar\* | El jugador intenta escapar del combate actual, tiene un 30% de probabilidaes de lograrlo |
| inv | Muestra los objetos del personaje en inventario |
| tienda | Muestra los objetos disponibles para comprar |

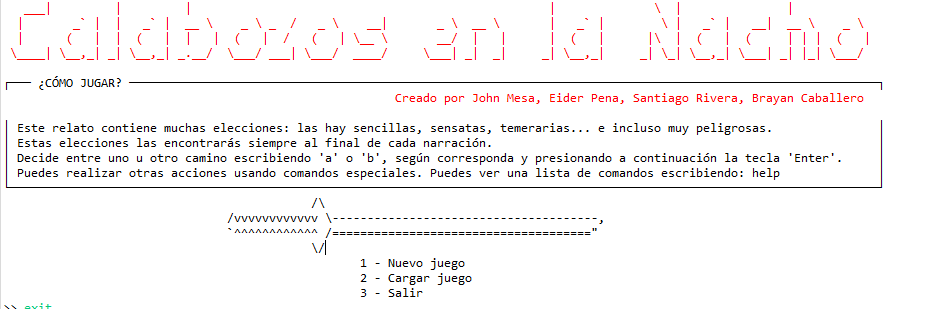
Al correr la aplicación se muestra por consola un enunciado corto dando la bienvenida al jugador, con una pequeña introducción de lo que se trata el demo y las opciones que tiene para elegir.

1 - Nuevo juego

2 - Cargar juego

3 – Salir

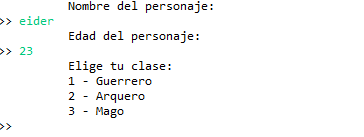
Estas opciones se ejecutarán, solo indicando el numero que precede al nombre de la opción. Ejemplo para cargar juego se debe marcar el número 2, tener en cuenta que esta opción solo funcionara si, anteriormente se empezó un juego nuevo.



Si se elige la opción 1-Nuevo juego, se mostrará por consola:

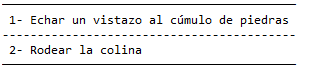


Con lo cual podemos personalizar el juego creando un personaje o simplemente tener una partida rápida con un personaje por defecto. Si se elije crear un personaje se pedirá que ingrese algunos datos y elija una clase para el personaje:

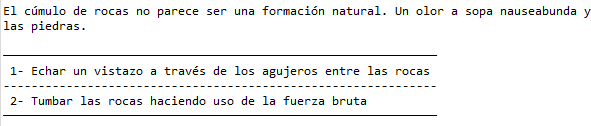


Luego de esto se dará un resumen del personaje creado y se dará inicio con la narración del juego.

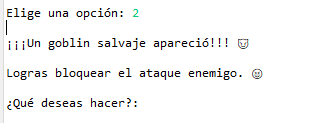
el capítulo 1- Oculto tras las sombras, con su respectiva narrativa al fino se dan dos opciones:



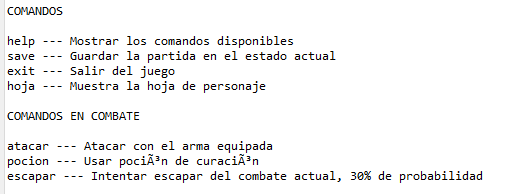
Esta vez en lo mostrado por la consola, ya dependerá de lo opción escogida, supongamos que es la opción 1- Echar un vistazo al cumulo de piedras. Eso desplegara en la consola lo siguiente:



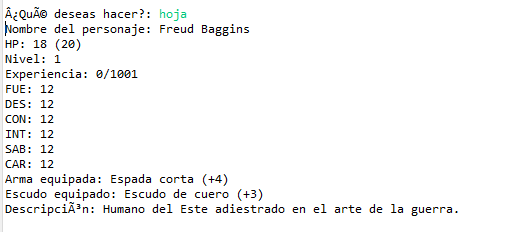
Luego de leer la narrativa decidimos seleccionar 2- Tumbar las rocas haciendo uso de la fuerza bruta., esta opción nos indica que debemos decir sobre qué acción realizar con este encuentra, acá ya se puede evidencia la función de combate.



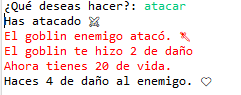
Para decidir que hacer tenemos en cuenta, los comandos no numéricos, de combate entre ellos, para ejemplificación ejecutamos help, el cual no ayudará a dar un vistazo en nuestras opciones:



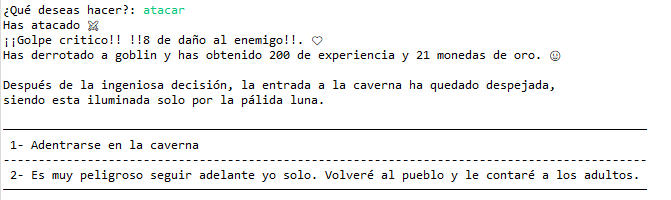
Ya con esto también ejecutamos hoja la cual nos brinda un resumen de nuestro personaje, con lo cual podemos tomar una decisión sobre el juego.

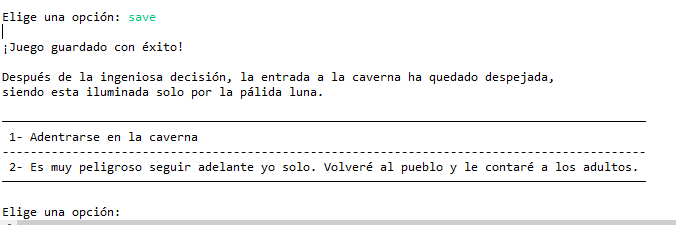


Se decide atacar,



De acá en adelante se pueden ejecutar comandos de combate hasta se acabe el combate y alguno gane o puedes guardar la partida, si ejecutas **save** el, guarda la partida y te vuelve a indicar la escena en la que quedas para saber si quieres seguir en el juego y si no es así deberás ejecutar **exit.**







Después de que se haya empezado el juego se podrá hacer uso de los comandos inventario y tienda los cuales nos brindan funciones muestra del equipamiento del personaje tanto en las armas, armaduras y pociones, la tienda nos ayuda a adquirir nuevo equipamiento.

Si ejecutamos tienda:

