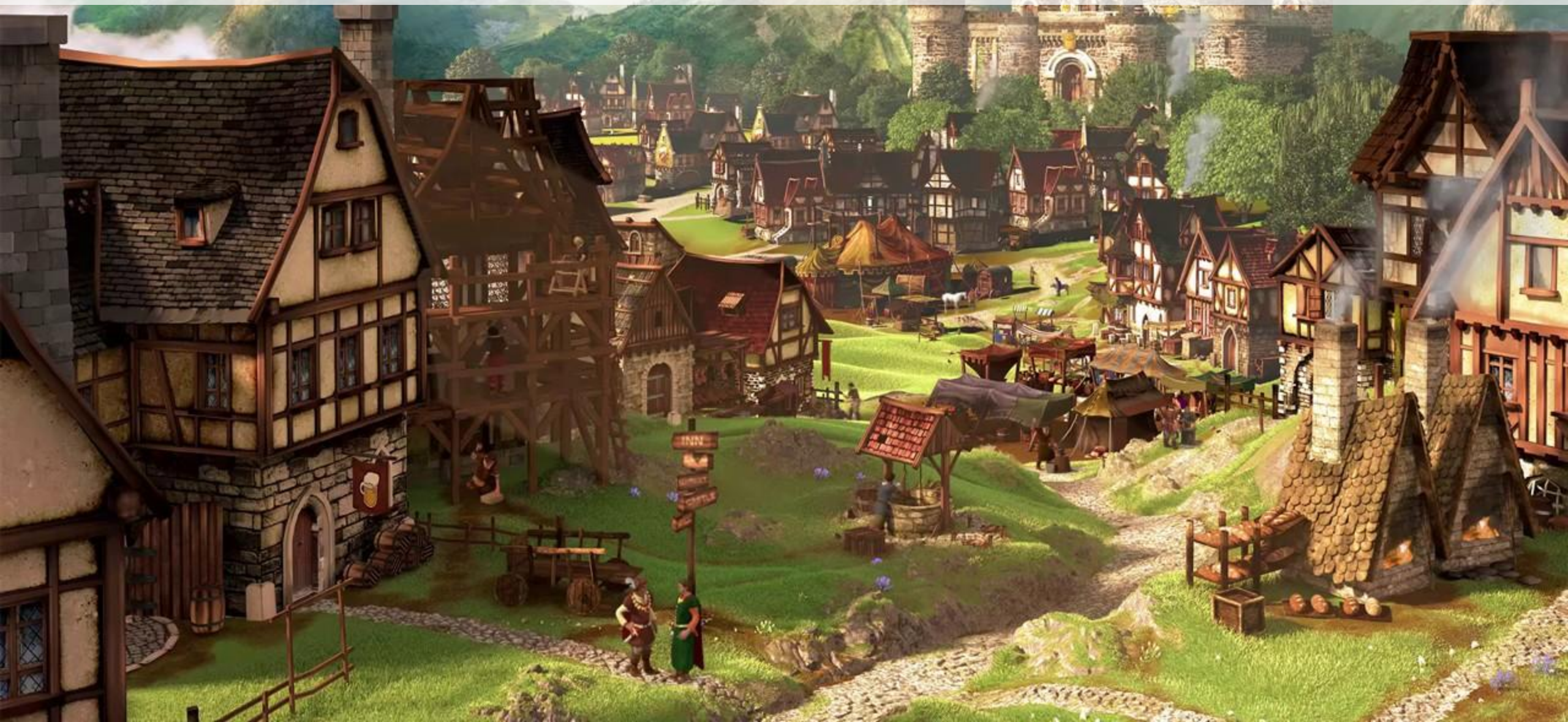
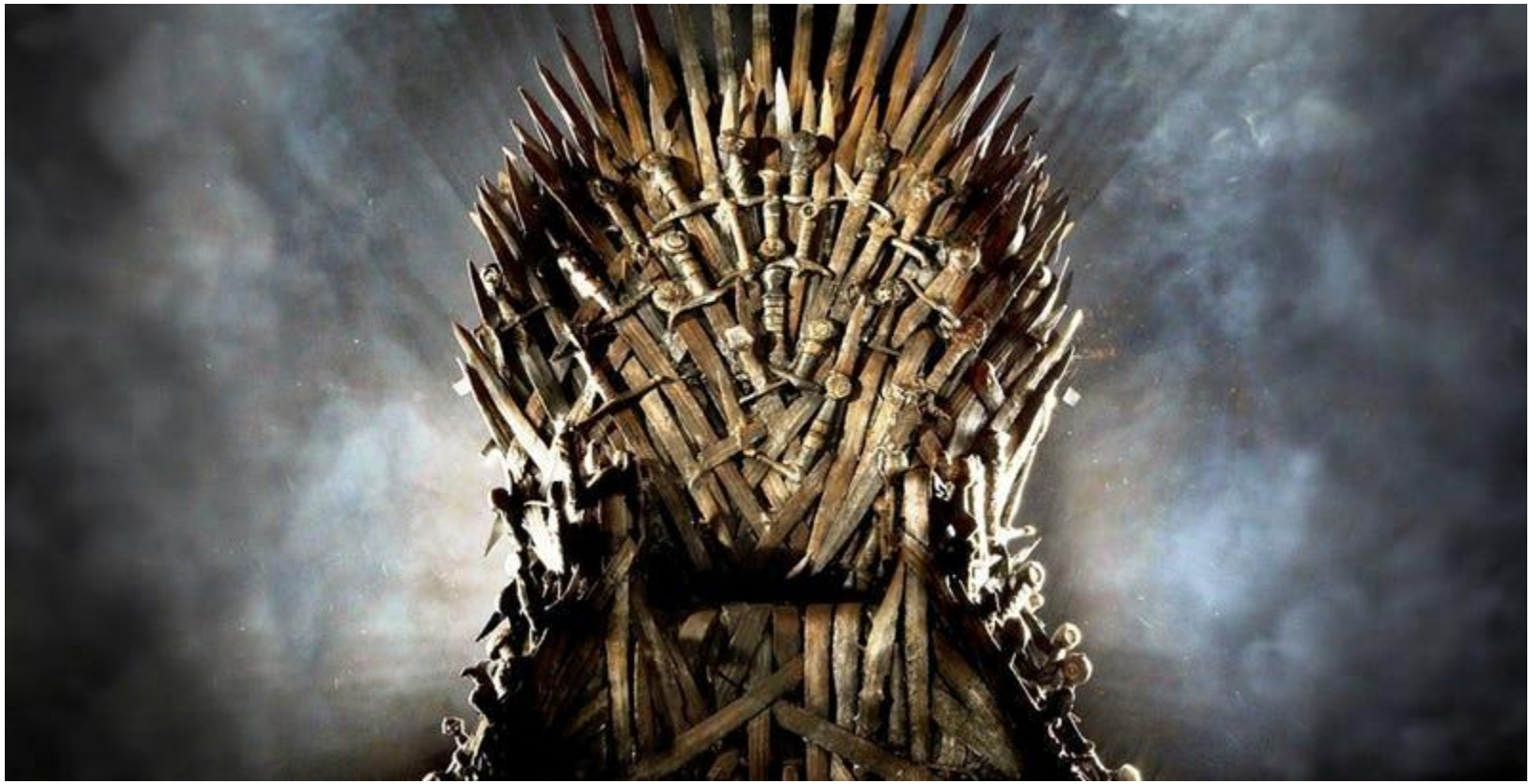
Mini Juego de rol con JAVA



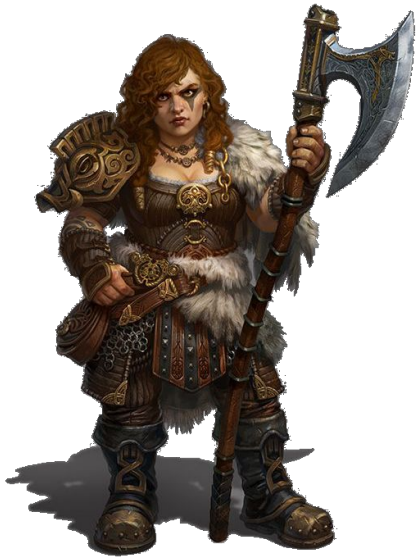
Mini Juego de rol con JAVA

Crear un juego de rol de cartas donde dos grupos de personajes se enfrenten..
Qué jugador será el ganador del trono?

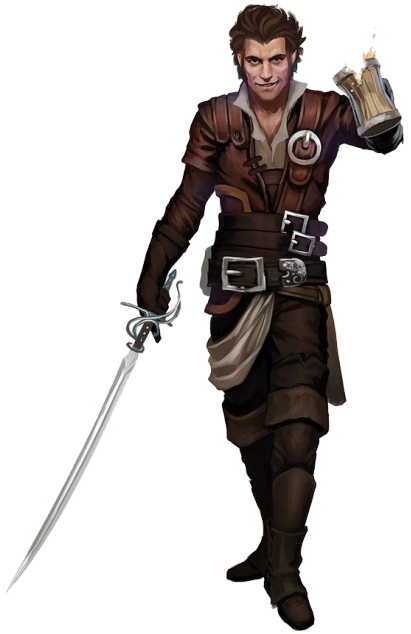


Generación de personaje

El resultado de la batalla se obtendrá por un sistema de combate basado en turnos y por un cálculo matemático/probabilístico utilizando las habilidades de cada personaje

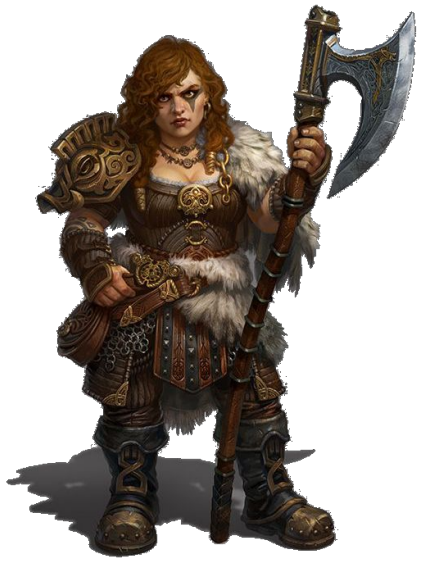


Vs

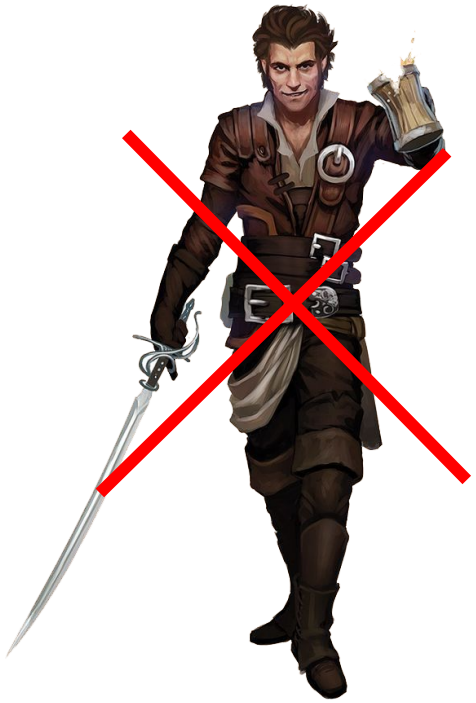


Generación de personaje

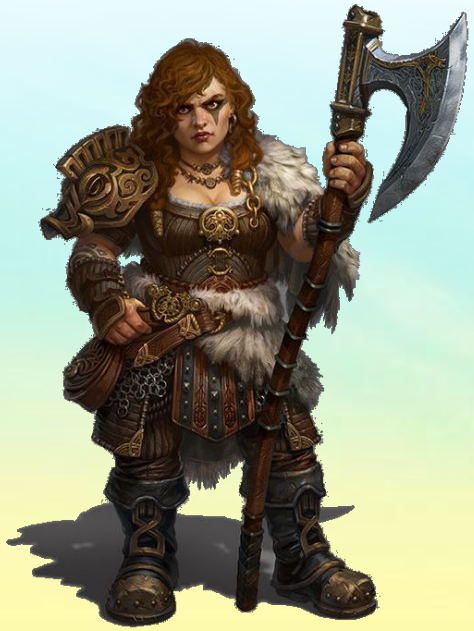
El/la ganador/a de cada ronda continuará podrá volver a pelear en las subsiguientes, mientras que el otro será eliminado de la partida.



Vs



Generación de personaje



Datos:

Raza (humanos/orcos/elfos)

Nombre;

Apodo;

Fecha de Nacimiento;

Edad; //entre 0 a 300

Salud://100

Imagen(opcional)

Características:

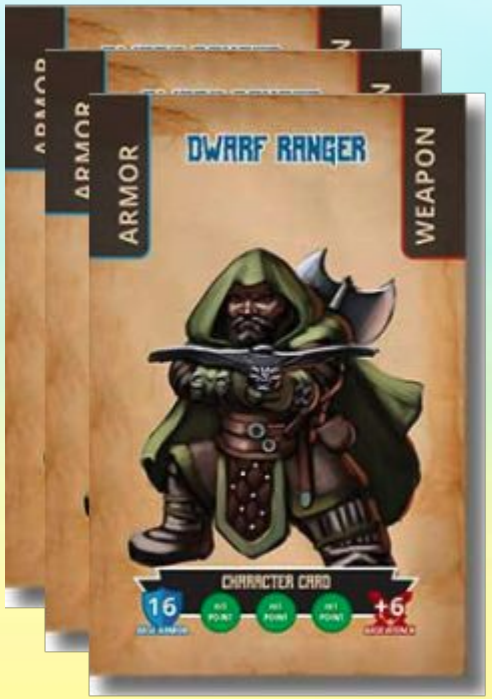
velocidad;// 1 a 10

destreza; //1 a 5

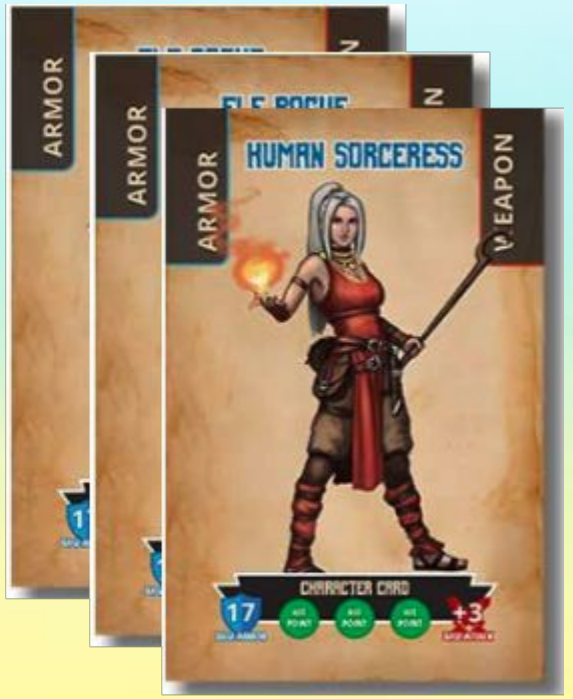
fuerza;//1 a 10

Nivel; //1 a 10

Armadura; //1 a 10



VS



Jugador1

Jugador 2

Mecánica del Combate

El sistema debe permitir generar 6 personajes al azar (y también debe permitir ingresar los datos de cada personaje a mano si se desea).

Una vez creados los 6 personajes(cartas), **se asignan 3 personajes(cartas)** a cada jugador.
El sistema elegirá al azar uno de los 3 personajes del jugador 1 y lo enfrentará con uno de los 3 personajes (al azar) del jugador 2.

En la primera ronda el sistema sortea quién empieza atacando, en las siguientes rondas empieza atacando el jugador que perdió la ronda anterior.

En cada ronda cada personaje tendrá **7 ataques** que irán debilitando el personaje (carta) del oponente. El flujo de ataque sería entonces:
Ronda 1:
Jugador 1 ataca , jugador 2 ataca , jugador 1 ataca, jugador 2 ataca.. ***hasta llegar a 7 ataques cada jugador o hasta que uno de los personajes muera (lo que ocurra primero).***
Ronda 2:
Jugador que perdió ronda 1 ataca , el otro ataca .. ***hasta llegar a 7 ataques cada jugador o hasta que uno de los personajes muera (lo que ocurra primero)***
Ronda N …

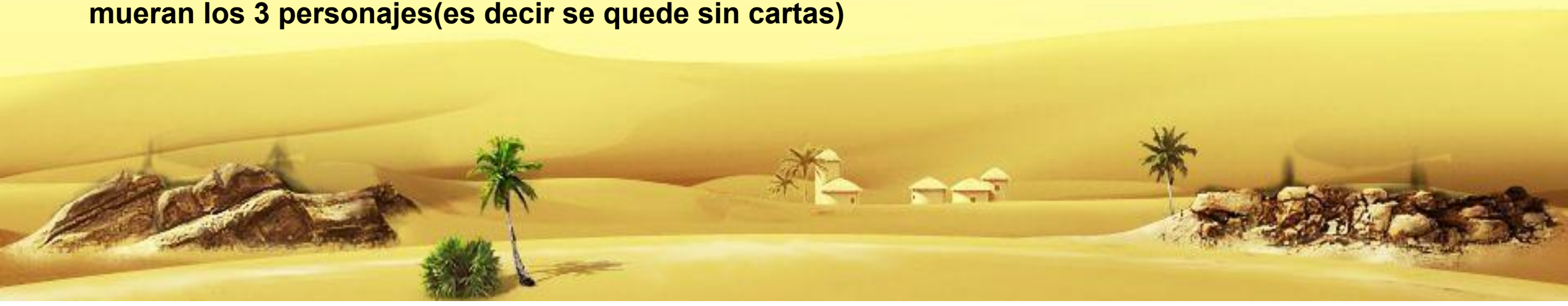
Mecánica del Combate

El personaje(carta) que pierda la ronda (pierde el personaje que quedó con Salud <= 0) será eliminado de la lista y el que gane será beneficiado con una mejora en sus habilidades que puede ser aleatorio o no, por ejemplo: +10 en salud, subir un nivel (libre elección).

En caso de luego de los 7 ataques de cada jugador, que ningún personaje llegue a Salud<=0 no se elimina ningún personaje. Podrán volver a salir sorteados en futuras rondas.

Al finalizar la ronda, se vuelve a sortear qué personaje del Jugador 1 luchará contra qué personaje del Jugador 2.

Se jugarán todas las rondas que sean necesarias hasta que a uno de los dos jugadores se le mueran los 3 personajes(es decir se quede sin cartas)



Cálculo del ataque

- **Poder de Disparo (PD)**: Haga el producto de Destreza * Fuerza * Nivel del personaje que ataca

- **Efectividad de Disparo**:Genere un valor aleatorio de 1 a 100. Considerarlo como valor porcentual (ED)

- **Valor de Ataque(VA)**:Al Poder de Disparo lo multiplico por la Efectividad de Disparo.

- **Poder de Defensa**: Haga el producto de Armadura * Velocidad del personaje que defiende (PDEF)

Cálculo del ataque

Resultado del enfrentamiento

- **Daño provocado**:
  si ataca un humano : ((VA*ED)-PDEF)/500)*100
  si ataca un elfo : (((VA*ED)-PDEF)/500)*100) * 1.05
  si ataca un orco : (((VA*ED)-PDEF)/500)*100) * 1.1

- **Actualizar salud del personaje** que se defiende Restándole a Salud el Daño provocado.

Desarrollando el Gameplay

Se debe poder ver en la consola(o en una ventana si se utilizó una UI web o algún JAVA UI) cómo va sucediendo la partida(lo que llamaremos un **Log**), lo sgte. es sólo un ejemplo, hacer a gusto y volar :)

- **Se generaron 6 personajes:**
  --------Personaje 1 Jugador 1-------
  nombre: Uhov, Tipo : Troll, Apodo:Chanerium , Velocidad: 8, …etc
  …..
  --------Personaje 2 Jugador 1 -------
  nombre: Rand, Tipo : Wizard, Apodo:Trumer , Velocidad: 1, …etc
  …
  Ronda 1
  El sistema sorteó al Jugador 1 para iniciar la ronda
  El sistema eligió al personaje Trumer del jugador 1 y al personaje XXXX de jugador dos para que se enfrenten en esta ronda.

Desarrollando el Gameplay

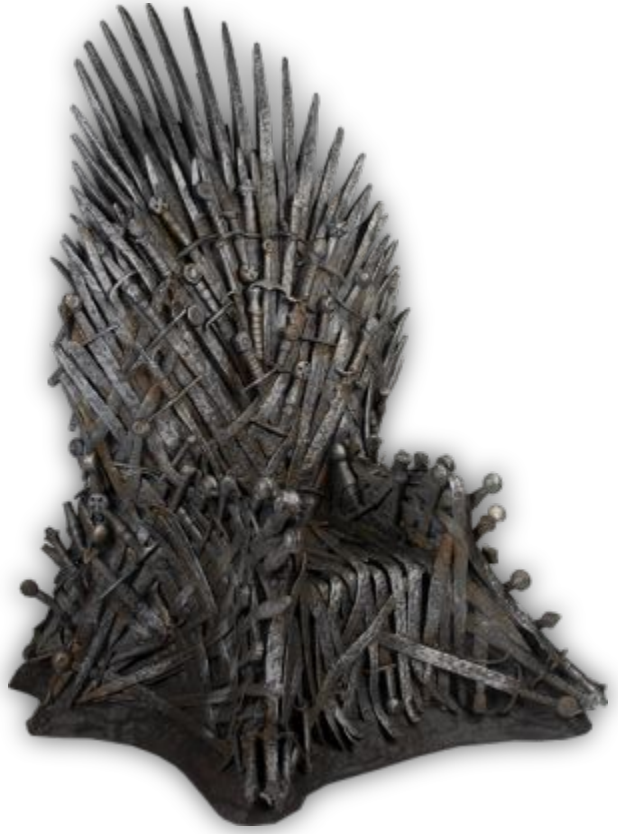
Chanerium ataca a XXXX y le quita 20 de salud. XXXX queda con 80 de salud.
XXXX ataca a Chanerium y le quita 12 de saluds. Chanerium queda con 88 de salud.
Chanerium ataca a XXXX y le quita ….
….
Muere XXXX.
Chanerium gana 10 de salud como premio, quedando con 90 de salud.

Ronda 2
Empieza atacando Jugador 2 por perder la ronda 1.
El sistema eligió al personaje Trumer del jugador 1 y al personaje YYYY del Jugador 2 para que se enfrenten en esta ronda.

YYYY ataca a Trumer y le quita 2 de salud. Trumer queda con 98 de salud
….
Ronda X
…..
Gana Jugador 2, le quedo/aron vivos los sgtes. personajes:
……
Felicitaciones Jugador 2 , las fuerzas mágicas del universo luz te abrazan!
Fin.

Mecánica del Combate

Al final de los enfrentamientos el jugador que sea declarado el ganador será merecedor del **Trono de Hierro**. Haga los honores correspondientes mostrando los datos de sus personajes por pantalla, indicando cuáles murieron y cuáles quedaron vivos.. y un mensaje destacado.



Criterio de aceptación del proyecto

-Crear diagrama UML.
-Implementar en Java la mecánica de combate descripta anteriormente.
-Hay total libertad para implementar la interfaz de usuario: solo consola, windows UI, web, mobile .. etc
Luego de finalizar cada partida, todo el log que se mostró se debe persistir en un archivo.

El menú del juego debe tener las opciones de :

- Iniciar partida y que el sistema genere los 6 personajes aleatoriamente.
- Iniciar partida pero que los personajes se ingresen a mano
- Leer desde el archivo los logs de todas las partidas jugadas
- Borrar archivo de logs
- Salir

Otros comentarios

- Las mejoras y ampliaciones son bienvenidas!
- nuevas razas, nuevas dinámicas
- Conectarse con APIs para generar nombres, apodos, que la calidad del aire en Bahía influya en el poder de los ataques o defensa…
- Unit tests…
- etc

Otros comentarios

- En caso de que el alumno quiera proponer otro proyecto, enviar un mail al profesor con la propuesta, el proyecto propuesto debe incluir sí o sí para ser aceptado:

  - Herencia de clases y polimorfismo
  - Manejo de archivos (creación / eliminación)
  - Log de lo sucedido en cada ejecución
  - Un Menú de uso
  - Cierta complejidad en la lógica de las clases.

Si detecto un proyecto con copia parcial o total de otro, el alumno será reportado a la autoridades de la UTN.

Otros comentarios

Recuerden que todo lo que hagan acá : manejo de archivos, programación JAVA, Tests, UML, etc.. es para SU currículum.

Aprovechen, si ven esto como una densitud total están a tiempo de replantearse su vida :)

éxitos grosos!!!

¿Play again?