

Proyecto Final: Mini Juego de rol con c#

Recordatorio: Todos los alumnos deben defender el proyecto final de la materia para aprobar.

¡IMPORTANTE!

- El proyecto establece un conjunto de requisitos generales que sirven como base, no obstante, requiere de su diseño, imaginación y creatividad para definir todos los detalles específicos del desarrollo.
- Los invitamos a que presenten propuestas superadoras o variantes originales al proyecto aquí presentado siempre que conserve los requisitos mínimos establecidos en las rúbricas.
- Ante cualquier duda sobre los requisitos, diseño y mejoras que desee implementar, **no dude en preguntar.**



Mini Juego de rol con c#

El objetivo es crear un juego de rol donde personajes se enfrenten en una competencia hasta que solo uno sea el ganador del trono de hierro.



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 la contienda continuará para una próxima batalla mientras que el otro será eliminado de la partida.



Clonando el repositorio

Clone el repositorio antes de comenzar la partida

https://classroom.github.com/a/Mi_NJsN0



Generación de personaje



Personaje:

- Datos
- Características

Características:

velocidad; // 1 a 10

destreza; //1 a 5

fuerza; //1 a 10

Nivel; //1 a 10

Armadura; //1 a 10

Salud; //100

Datos:

Tipo;

Nombre;

Apodo;

Fecha de Nacimiento;




Edad; //entre 0 a 300

Salvando la partida

Tenga en cuenta que tiene que subir los cambios al repositorio remoto una vez logrados los avances actuales.

In case of fire



-  1. `git commit`
-  2. `git push`
-  3. `exit building`

Generación de valores aleatorios

- 1) Genere una clase para poder crear personajes aleatorios que se llame FabricaDePersonajes
- 2) Debe tener un método que retorne un personaje con sus respectivos datos y características cargadas.






Salvando la partida

Tenga en cuenta que tiene que subir los cambios al repositorio remoto una vez logrados los avances actuales.

In case of fire



-  1. `git commit`
-  2. `git push`
-  3. `exit building`

Desarrollando el Persistencia



Persistencia de datos (Lectura y guardado de Json)

1ra Parte

- 1) Armar una clase llamada PersonajesJson para guardar y leer desde un archivo Json
- 2) Crear un método llamado GuardarPersonajes que reciba una lista de personajes, el nombre del archivo y lo guarde en formato Json.
- 3) Crear un método llamado LeerPersonajes que reciba un nombre de archivo y retorne la lista de personajes incluidos en el son.
- 4) Crear un método llamado Existe que reciba un nombre de archivo y que retorne un True si existe y tiene datos o False en caso contrario.



Persistencia de datos (Lectura y guardado de Json)

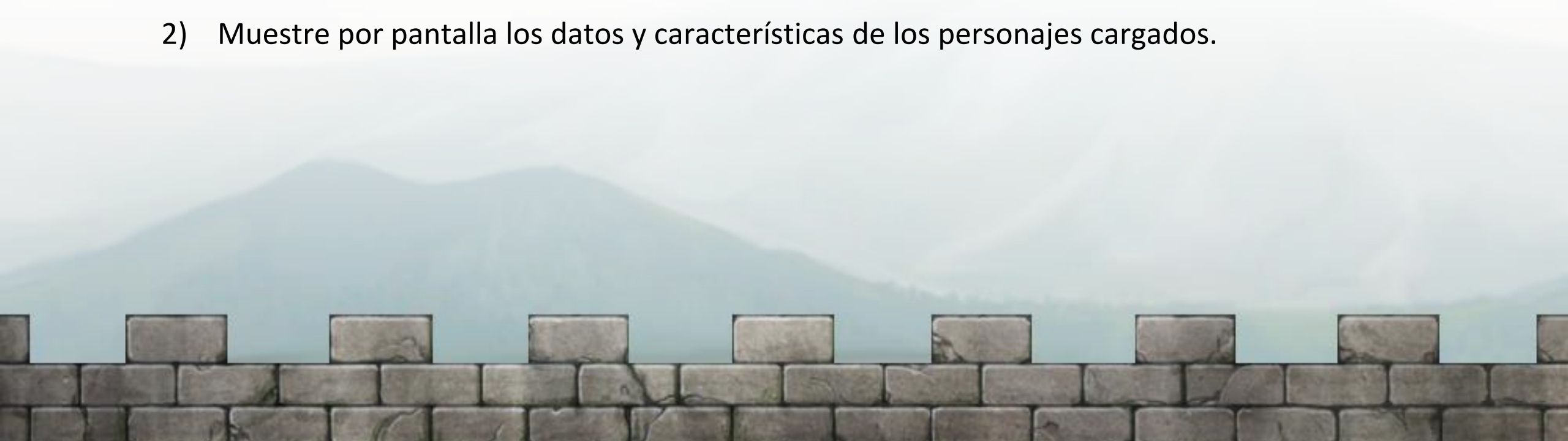
2da Parte

- 1) Armar una clase llamada HistorialJson para guardar y leer desde un archivo Json
- 2) Crear un método llamado GuardarGanador que reciba el personaje ganador e información relevante de las partidas, el nombre del archivo y lo guarde en formato Json.
- 3) Crear un método llamado LeerGanadores que reciba un nombre de archivo y retorne la lista de personajes ganadores e información relevante incluidos en el Json.
- 4) Crear un método llamado Existe que reciba un nombre de archivo y que retorne un True si existe y tiene datos o False en caso contrario.



Implementación del Juego

- 1) Verificar al comienzo del Juego si existe el archivo de personajes:
 - A. Si existe y tiene datos cargar los personajes desde el archivo existente.
 - B. Si no existe generar 10 personajes utilizando la clase FabricaDePersonajes y guárdelos en el archivo de personajes usando la clase PersonajesJson.
- 2) Muestre por pantalla los datos y características de los personajes cargados.






Salvando la partida

Tenga en cuenta que tiene que subir los cambios al repositorio remoto una vez logrados los avances actuales.

In case of fire



-  1. `git commit`
-  2. `git push`
-  3. `exit building`

Desarrollando el Gameplay



Mecánica del Juego

- 1) Elija 2 personajes para que compitan entre ellos.
- 2) El combate se realiza por turnos. Por cada turno un personaje ataca y el otro se defiende.
- 3) El combate se mantiene hasta que uno es vencido (salud ≤ 0)
- 4) El personaje que pierde la batalla es eliminado de la competencia
- 5) El que gane es beneficiado con una mejora en sus habilidades.
por ejemplo: +10 en salud o +5 en defensa.



Mecánica del Combate

La forma de calcular el daño provocado en cada turno es la siguiente:

- **Ataque:** Destreza * Fuerza * Nivel (del personaje que ataca)
- **Efectividad:** Valor aleatorio entre 1 y 100.
- **Defensa:** Armadura * Velocidad (del personaje que defiende)
- **Constante de Ajuste:** 500

$$\text{Daño provocado} = \frac{(\text{Ataque} * \text{Efectividad}) - \text{Defensa}}{\text{Constante de Ajuste}}$$

Una vez que se realice el turno actualizar la **salud del personaje que defiende**

$$\text{Salud} = \text{Salud} - \text{Daño provocado}$$

Mecánica del Combate

Al final de los enfrentamientos deberá quedar un único personaje en pie. Este será declarado el ganador y será merecedor del **Trono de Hierro**. Haga los honores correspondientes mostrando sus datos por pantalla y un mensaje destacado. Además, agréguelo al **ranking histórico de ganadores**.






Salvando la partida

Tenga en cuenta que tiene que subir los cambios al repositorio remoto una vez logrados los avances actuales.

In case of fire



-  1. `git commit`
-  2. `git push`
-  3. `exit building`

GAME
OVER

¿Play again?