Informe de aplicacion Proyecto Acceso a Datos Egoitz Perez de Arrilucea 3 Grado Superior Desarrollo de Aplicaciones Multiplataforma 2022-2023

Descripcion:
La aplicacion es un juego de combate mediante consola.
El usuario puede crear su propio jugador y luchar contra diferentes enemigos, con cada combate se sube de nivel que posteriormente permitira mejorar caracteristicas.
El jugador puede ser exportado a .xml y .dat, para en proximas partidas cargar el jugador de el .dat.
La partida tambien puede ser exportada a .xml.

Classes:

1. Main

   Funciones:

   1. Main()

      Primero se intenta importar el fichero de los enemigo y si no se puede lo creara.
      Despues se le mostrara al usuario las opciones de nuevo jugador: crear uno nuevo o importarlo, si decide crear uno nuevo se llamara a la funcion crearJugador() , si decide importar llamara a la funcion importarJugadorDAT(), si hay un error al importar mostrara un mensaje de error y las opciones de nuevo jugador de nuevo.
      Mostrara los datos de el jugador para informar al usuario.
      Se creara el objeto partida.
      Para los combates se entrara en un bucle, primero se llamara a la funcion seleccionarEnemigo(enemigos) para seleccionar un enemigo que se le pasara a la funcion combate(jugador, enemigoSeleccionado, partida) esta devolvera true o false para saber si el usuario ha ganado o perdido. Si ha perdido se registrara la derrota en el objeto partida y se terminara el bucle, si ha ganado se registrara el resultado en la partida y si el enemigo es el aleatorio añadira 100 puntos extra por la dificultad de este, ademas le subira el nivel a el jugador, despues comprobara si el nivel es un multiple de 5 entonces le dara una mejora al usuario, se puede elegir entre tener 10 mas de vida o mas 5 de ataque.
      Fuera de el bucle (Solo se puede salir perdiendo) se guardan los datos de la partida y se le muestran al usuario, se le da la opcion de exportar esa partida con la funcion exportarPartidaXML(partida), y despues se le da la opcion de exportar el jugador con las funciones Jugador.exportarJugadorDAT(jugador) y Jugador.exportarJugadorXML(jugador).

   2. Enemigo seleccionarEnemigo(Enemigo[])

      Recibe la lista de enemigos, los printea y pide al usuario que elja uno, al final devuelve el enemigo seleccionado. Ademas muestra un enemigo aleatorio que no esta en la lista, si el usuario selecciona este enemigo esta misma funcion crea un enemigo nuevo con variables aleatorias y lo devuelve.

   3. Boolean combate(Jugador, Enemigo, Partida)

      Recibe el jugador actual y el enemigo seleccionado por la funcion seleccionarEnemigo(), entra en un bucle hasta que se termine el combate, en el combate el primer turno sera de el jugador, tendra la opcion de atacar o curarse, una vez acabado su turno se realizara la operacion, sumar vida al jugador o restarle al enemigo y le sumara puntuacion por la operacion, despues se comprobara si al enemigo le queda vida, si no es asi el combate terminara, se saldra de el bucle y se retornara true indicando que ha ganado, si no sera el turno de el enemigo, este siempre atacara, se le resta la vida al jugador y se comprueba si esta llega a 0, si es asi el combate terminara y se devolvera false indicando que ha perdido y terminando el programa.

   4. Jugador crearJugador()

      Pide un nombre al usuario y crea un nuevo objeto jugador con ese nombre y los datos por defecto (Vida = 100, Ataque = 25), y preguntara al usuario si desea exportarlo. Al final devolvera el objeto jugador.

2. Partida

   Variables:

   1. LocalDateTime fecha
   2. int puntuacionTotal
   3. String[] combates
   4. String nombreJugador

   Funciones:

   5. void exportarPartidaXML(Partida)
      se crea el Xstream, se cambian las etiquetas, se escribira el objeto Partida recibido, Se crea el fichero Partida_fecha.xml y mostrara un mensaje para señalizar que ha acabado.

3. Jugador

   Variables:

   1. String nombre
   2. int vida
   3. int ataque
   4. int nivel

   Funciones:

   5. Jugador importarJugadorDAT()
      Se crea un objeto Jugador vacio, se busca el fichero Jugador.dat, si no existe se le informara al usuario, si existe se crearan los flujos, se le el jugador y muestra sus datos. Al final devolvera el objeto Jugador.

   6. void exportarJugadorDAT(Jugador)
      Busca el fichero Jugador.dat, si existe se le informara al usuario y se le mostrara el jugador guardado, seguido se le pregunta si desea sobreescrirlo o no, si decide que no la funcion terminara. Si en cambio decide sobreescribirlo o no hay un fichero previo existente se crearan los flujos, se escribira el objeto Jugador recibido y mostrara un mensaje para señalizar que ha acabado.

   7. void exportarJugadorXML(Jugador)
      Busca el fichero Jugador.xml, si existe se le informara al usuario, seguido se le pregunta si desea sobreescrirlo o no, si decide que no la funcion terminara. Si en cambio decide sobreescribirlo o no hay un fichero previo existente se crea el Xstream, se cambian las etiquetas, se escribira el objeto Jugador recibido y mostrara un mensaje para señalizar que ha acabado.

4. Enemigo
   Variables:
   1. String nombre
   2. int vida
   3. int ataque

   Funciones:
   4. Enemigo[] importarEnemigosDAT()
      Crea una lista de objetos Enemigo vacia, busca el fichero Enemigos.dat . Si el fichero existe creara los flujos y ira leyendo el fichero en la lista, si no existe mostrara un mensaje de error y dejara la lista vacia. Al final devolvera la lista.

   5. void exportarEnemigosDAT(Enemigo[])
      Crea el fichero Enemigos.dat, crea los fujos, va escribiendo los enemigos de la lista que ha recibido y muestra un mensaje cuando termina.

Ficheros:

1. Enemigos.dat
   Guarda la lista de objetos enemigo, se crea con el metodo exportarEnemigosDAT() y lee con el metodo importarEnemigosDAT(). El usuario no duede crear este fichero, viene creado con la applicacion ,al comienzo de la partida se importaran los enemigos guardados en el fichero, si el fichero no esta se creara uno nuevo automaticamente.

2. Jugador.dat
   Guarda el objeto jugador, se crea con el metodo exportarJugadorDAT() y lee con el metodo importarJugadorDAT(), El usuario decide si lo crea al crear el objeto y al final de la partida, tambien decide si en lugar de crear un objeto nuevo desea importarlo de el fichero al comienzo de la partida.

3. Jugador.xml
   Guarda el objeto jugador, se crea con el metodo exportarJugadorXML() usando Xstream, El usuario decide si lo crea al crear el objeto y al final de la partida.

4. Partida_fecha.xml
   Guarda el objeto partida , se crea con el metodo exportarPartidaXML() usando Xstream. El usuario decide si lo crea al final de la partida.