MANUAL TÉCNICO

IDE utilizada: Visual Studio Code

Sistema operativo objetivo: Windows, Linux, Macintosh, OS X

Lenguaje de programación utilizado: Java

Diagrama de clases

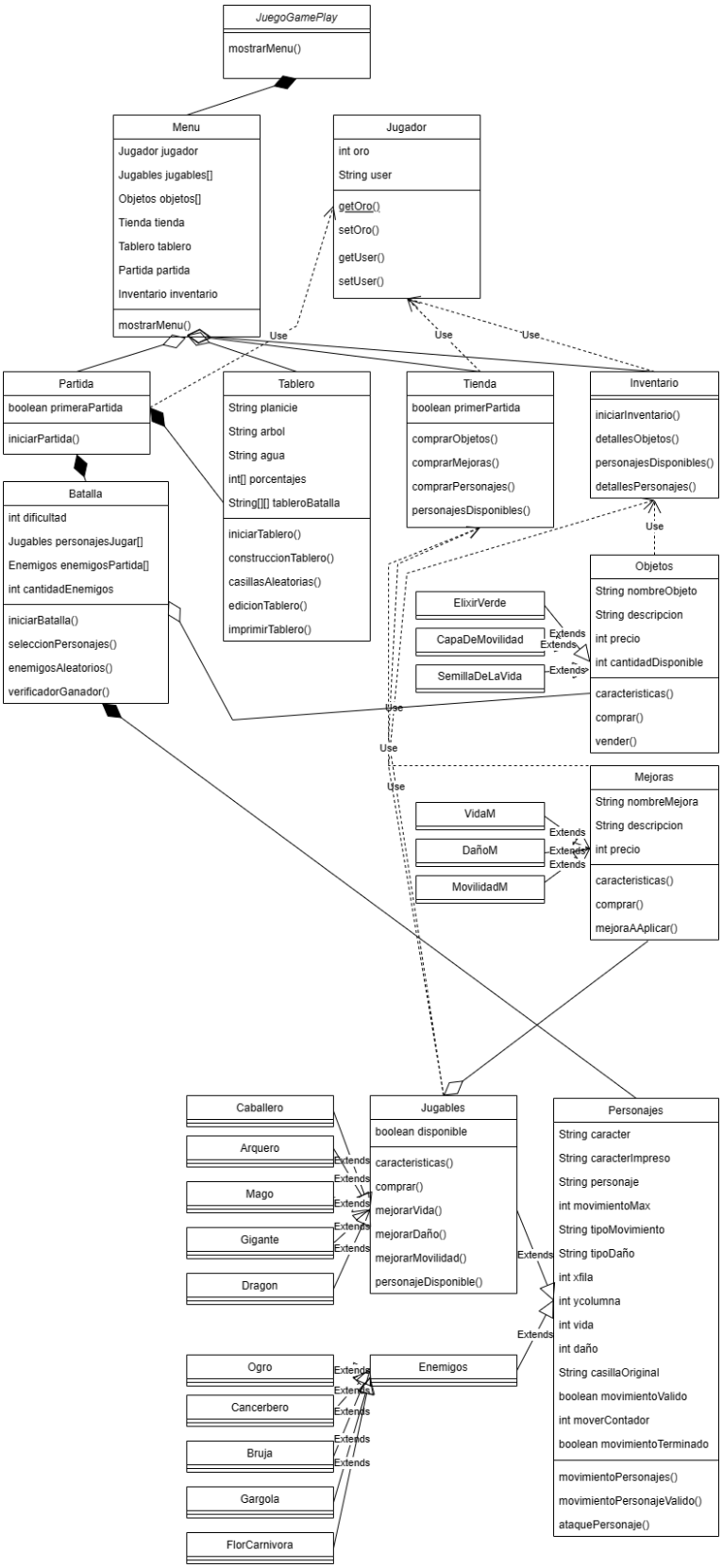
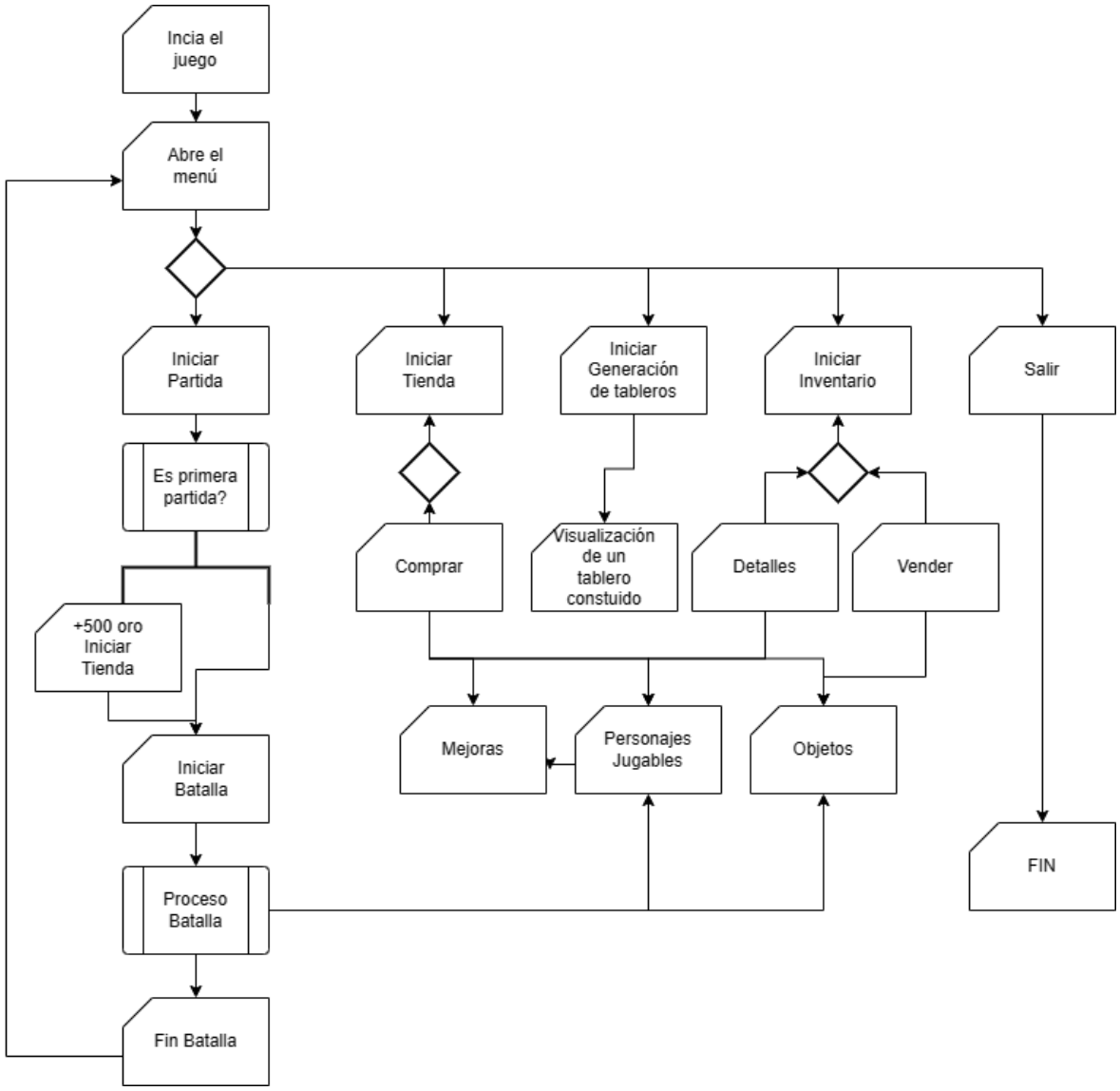


Diagrama de flujo general



Métodos importantes

mostrarMenu()

Se verifica si es la primera partida

Si es la primera partida

    Mostrar que ingrese a iniciar partida para comenzar

Sino

    Unicamente mostrar la selección de opciones

Imprimir el menú, junto a las opciones y guardar el dato de opción

Si opción es 1, entonces se llama a iniciarPartida()

Si opción es 2, entonces se llama a iniciarTienda()

Si opción es 3, entonces se llama a iniciarTablero()

Si opción es 4, entonces se llama a iniciarInventario()

Si opción es 5, se cierra el juego, terminando el ciclo

Si opción es difente, entonces se repite la solicitud de la opción

iniciarPartida()

Se verifica si es su primera partida

Si sí es la primera partida

    Mostrar una bienvenida

    Solicitar el nombre

    Acreditar 500 de oro

    Se abre la Tienda

Si no

    Se abre la batalla

iniciarTablero()

Se abre la selección de tableros y se solicita que ingrese una

Si opción es 1, entonces se usa un mapa 8x8 y los porcentajes de casillas predeterminados

Si opción es 2, entonces se usa un mapa con longitud que el usuario ingresa, además de utilizar porcentajes de casillas personalizados por el usuario

Si porcentajes suman 100

    Avanza a la construcción del tablero

Si porcentajes no suman 100

    Repetir la solicitud de parámetros para un mapa personalizado

construccionTablero()

Se crea el tablero con las longitudes declaradas

Se construye en la fila 0? Se construye en la columna 0?

Si sí, entonces se construye cada valor del tablero dando a cada casilla un valor de coordenada, de esta forma -- | 1 | -- y así hasta llegar a la longitud

Si se construye en la columna 0, entonces se dan los valores de las coordenadas

Si no, entonces se construye cada valor del tablero dando a cada casilla un valor vacío, de esta forma -- | | -- y así hasta llegar a la longitud

casillasAleatorias()

Tomando los valores de porcentaje de creación, empezamos con crear las casillas tipo árbol, luego tipo agua, luego tipo lava y finalmente tipo planicie.

Calculamos la cantidad de casillas a crear, calculando las casillas totales (longitud*longitud) dividido el porcentaje en decimal (porcentaje/100.0)

Se verifica que el contador de casillas sea menor o igual a la cantidad de casillas

    Sí, entonces continua

    No, entonces termina el proceso

Se verifica si la casilla a crear es planicie?

Sí, entonces crear casillas hasta que no exista ninguna con valor vacío y se marca como terminado

No, entonces continuamos

Se construye por valores que sí dando uno de ellos sí cree la casilla

Valor aleatorio es igual a n

Sí, entonces crear casilla

Casilla es diferente a agua?

Sí, entonces se crea la casilla

No, entonces crea la casilla, pero solo si está en las orillas

No, entonces saltar a la siguiente casilla

Si el tablero no se ha terminado, entonces repite el proceso

edicionTablero()

Obtenemos el valor de fila y columna destino y el carácter para usar

Se crea el valor en tablero[fila][columna] es igual a carácter

imprimirTablero()

Se recorre cada casilla del tablero y se muestra en pantalla una por una hasta imprimir el tablero

movimientoPersonajes()

Creamos un ciclo que se repita si el valor ingresado es diferente a los movimientos básicos AWSD

Si la opción es A, W, S o D

Llama al método movimientoPersonajeValido, dando los parámetros de la ubicación destino

Este método se repite si el contador de movimientos es menor a los movimientos máximos y que el movimiento no se haya terminado

Si es diferente, o el movimiento no es válido se repite el ciclo

movimientoPersonajeValido

Se verifica que la casilla destino, con valores xfila y ycolumna, sean válidos, por ejemplo para los de tierra sea tipo planicie o lava, y los que vuelan sea tipo planicie, lava, agua, árbol

Sí, entonces la casilla donde se ubica el personaje se restablece, dandode el valor de planicie u otra casilla original, luego se cambia el valor de casilla original a la

nueva casilla destino, para almacenar a donde se posicione el personaje. Luego se cambia el valor de la casilla destino por la impresión del personaje.

Si se quiere mover más veces, se verifica si el valor de movimientos máximos sea mayor a 1

Sí, entonces aumenta en 1 el contador de movimientos y repite el proceso.

No, entonces marca el movimiento como terminado

No, la casilla es diferente a las permitidas, entonces marca el movimiento como No Valido

movimientoPersonajes en enemigos

Se muestra la posición y nombre del enemigo activo

Cambiamos el ingreso del movimiento AWSD por un número aleatorio

numeroAleatorio = aleatorio entre 4 y 1)

Si es 1 equivale a A

Si es 2 equivale a W

Si es 3 equivale a S

Si es 4 equivale a D

De la misma manera, si es movimiento no es válido, o el movimiento no ha terminado se repite el ciclo en la misma dirección

Se muestra la nueva posición y nombre del enemigo

movimientoPersonajeValido en enemigos

Usa el mismo procedimiento que antes, pero se debe usar un número aleatorio para saber si se mueve una casilla más o no hasta su máximo de movimientos

numeroAleatorio = aleatorio entre 2 y 1,

Si es 1, entonces verifica si su contador de movimientos es menor a movimientos máximos

Sí, entonces aumenta el contador en 1

No, entonces marca el movimiento como terminado

Si es 2, marca el movimiento como terminado

caracteristicas() similar en varias clases

Mostrar la vida del personaje

Mostrar el daño del personaje

Mostrar el tipo de movimiento

Mostrar el movimiento máximo

Mostrar el tipo de ataque

comprar() similiar en varias clases

Mostrar el oro disponible

Verificar si ya se compró el personaje

    Sí, entonces mostrar que ya lo adquirió

    No, entonces continuar

Mostrar ¿Desea comprar al personaje por precio de oro?

Verificar la respuesta

    Sí, entonces verificar la cantidad de oro

    Es mayor?

        Sí, entonces descontar el precio y marcar como comprado

        No, entonces mostrar que no tiene suficiente oro

personajeDisponible()

Si disponible es falso

    Entonces mandar un texto vacío “”

Si disponible es verdadero

    Entonces mandar un texto con el nombre del personaje

Retornar el texto

personajeEstado()

El personaje está sobre lava?

Sí, entonces vida es mayor a 5?

Sí, entonces descontar el 0.05*vida

No, entonces descontar 1

No, entonces no hace nada

Mostrar vida