Menu

-personajes: Diccionario
-tablero: Tablero
-jugador1: str[ ]
-jugador2: str[ ]

+mostrar_menu()
+ingresar_opcion(): int
+elegir_opcion()
+procesar_opcion(int,bool)
+pedir_datos_personaje(str, str)
+agregar_nuevo_personaje()
+eliminar_personaje(str)
+mostrar_nombres_personajes()
+mostrar_datos_personaje(str)
+alimentar_personaje(str)
+esta_en_el_diccionario(str): bool
+comenzar_juego()
+mostrar_menu_2()
+seleccionarPersonajes()
+comienzoJuego(int)
+guardarPartida(str[ ], int)
+leer_partida()
+pedirNombre(): str
+validar_elemento(str)
+validar_datos(str)
+validar_opcion(int, int, int): bool
+existePartida(): bool
+chequeaRepetido(str[ ]): str
+esta_repetido(str, str[ ]): bool
+contadorVida(str[ ]): int
+imprimirTurno(int)
+primerasOpcInternas(str)
+segundasOpcInternas(str, str[ ])
+posicionarPersonajes(int)
+ponerPersonaje(str, int)
+atacar(str)
+defenderse(str, str[ ])
-bienvenida()
+opcionGuardar(): str
+procesarLinea(str): str
+modificarDatos(str, int, int, int, int)
-asignarTurno(): int

Diccionario

-raiz: *NodoABB

-leer_archivo()
-procesar_linea_archivo(str)
-cargar_atributos_personaje(Personaje,int,str,int,str)
+getRaiz(): NodoABB
+agregarNodo(NodoABB,str,Personaje): NodoABB
+agregarPersonaje(str,Personaje)
+agregarPersonaje(Personaje,int,str,int,str)
+buscar(NodoABB,str): NodoABB
+buscar(str): bool
+traer(str): Personaje
+mostrarOrdenados(NodoABB)
+mostrarOrdenados()
+encuentraMinimo(NodoABB): NodoABB
+quitarRaiz()
+quitarNodo(NodoABB)
+quitarNodo(str)
+quitarHoja(NodoABB)
+quitarNodoConHijoDerecho(NodoABB)
+quitarNodoConHijoIzquierdo(NodoABB)
+quitarNodoConDosHijos(NodoABB)
+limpiar()
+vacio(): bool

Tablero

#TABLERO: [int][int]

+cargar_casillero(int, int, int)
+cargar_tablero()
+poner_personaje(int, str str)
+consulta_disponible(int, int): bool
+consulta_personaje(int, int): str
+cambiarDisponible(int, int)
+consulta_equipo(int, int): int
+guardar_equipo(int, int, int)
+sacar_personaje(int, int): str
+mostrar_tablero()
+espacio()
+colorIndice(char): str
+inicioColor(char, char): str
+print(str, char, char)
+elegir_color_fondo(str): char
+mostrar_casillero(char, char, str)
+mostrar_vacio(int)
+mostrar_indice_fila()
+mostrar_movimiento(Lista)

NodoABB

-key: str
-value: Personaje
-izquierdo: NodoABB
-derecho: NodoABB
-padre: NodoABB

+setKey(str)
+setValue(Personaje)
+setDerecho(NodoABB)
+setIzquierdo(NodoABB)
+setPadre(NodoABB)
+setHijoNuevo(NodoABB)
+setHijoNuevo(NodoABB,NodoABB)
+quitarPadre(NodoABB)
+getKey(): str
+getValue(): Personaje
+getDerecho(): NodoABB
+getIzquierdo(): NodoABB
+getPadre(): NodoABB
+HijoIzquierdoUnico(): bool
+HijoDerechoUnico(): bool
+DosHijos(): bool
+esHoja(): bool

Grafo

-raiz: Lista
-listaCasilleros: Lista
-largo: int
-matrizPesoAire: int
-matrizCaminoAire: int
-matrizPesoFuego: int
-matrizCaminoFuego: int
-matrizPesoAgua: int
-matrizCaminoAgua: int
-matrizPesoTierra: int
-matrizCaminoTierra: int

-iniciarMatrices(int**, int**)
-leer_archivo(): Lista
-cargarGrafo()
-buscarSublista(Vertice): Lista
-agregarVertice(Vertice, Vertice)
-calcularPesoAire(Casillero): int
-calcularPesoTierra(Casillero): int
-calcularPesoFuego(Casillero): int
-calcularPesoAgua(Casillero): int
-calcularPesoElemento(Casillero, str): int
-calcularPeso(Casillero, Casillero, str): int
-matrizCamino(str): int
-matrizEnergia(str): int
-caminosMinimos(Lista, str, int**, int**)
-algoritmoFloyd(int**, int**)
-encontrarCamino(int, int, int **)
-obtenerCaminoMinimo(int, int, int**, Lista)
+mostrar()
+mostrarMatriz(int**)
+mostrarCamino(Lista)
+energiaNecesaria(int, int, str): int
+obtenerCaminoMinimo(int, int, str): Lista

Lista

-primero: Dato
-largo: int

+alta(Dato)
+obtenerLargo(): int
+buscarElemento(str): Dato
+baja(int)
+listaVacia(): bool
+mostrarLista()
+buscarNodo(int): Nodo
+buscarNodo(Dato): Nodo

Nodo

-elemento: Dato
-siguiente: Nodo

+obtenerDato(): Dato
+asignarSiguiente(Nodo)
+obtenerSiguiente(): Nodo

Casillero

-posX: int
-posY: int
-tipoTerreno: str
-disponible: bool
-numero: int
-equipo: int
-personaje: str

+setX(int)
+setY(int)
+setTerreno(str)
+cambiarDisponible()
+ponerPersonaje(str)
+setEquipo(int)
+getTerreno(): str
+getDisponible(): bool
+getX(): int
+getY(): int
+getPersonaje(): str
+getEquipo(int)
+getNumero(): int
+distancia(Casillero): int
+mostrar()

Vertice

-Value: T1

+getValue(): T1
+setValue(T1)

Personaje

-nombre: str
-tipo: str
-escudo: uns int
-vida: uns int
-energía: uns int

+obtenerNombre(): str
+obtenerTipo(): str
+obtenerVida(): int
+obtenerEscudo(): uns int
+obtenerEnergia(): uns int
+cambiarEnergia(int)
+cambiarVida(int)
+*alimentarse()*
+mostrarDatos()

Agua

-cantidad_alimentado: int
-COSTO_ENERGIA_ATAQUE: uns int
-COSTO_ENERGIA_DEFENSA: int
-DEBILIDAD_TIERRA: int
-FORTALEZA_FUEGO: int
-ATAQUE: int
-DEFENSA_PERSONAL: int
-DEFENSA_EQUIPO: int
-elemento_debilidad: str
-elemento_fortaleza: str

+alimentarse()
+validarEnergiaAtaque(): bool
+validarEnergiaDefensa(): bool
+objetivos(Tablero, str[ ])
+calculo_ataque(int, str, int)
+defenderse(Tablero)

Fuego

-COSTO_ENERGIA_ATAQUE: int
-COSTO_ENERGIA_DEFENSA:
-DEBILIDAD: int
-FORTALEZA: int
-ATAQUE: int
-DEFENSA_PERSONAL: int
-elemento_debilidad: str
-elemento_fortaleza: str

+alimentarse()
+validarEnergiaAtaque(): bool
+validarEnergiaDefensa(): bool
+objetivos(Tablero)
+calculoAtaque(int, str, int)
+defenderse(Tablero)

Tierra

-COSTO_ENERGIA_ATAQUE: int
-COSTO_ENERGIA_DEFENSA: int
-DEBILIDAD: int
-FORTALEZA: int
-ATAQUE_CERCANO: int
-ATAQUE_MEDIANO: int
-ATAQUE_LEJANO: int
-ESCUDO_EXTRA: int
-DEFENSA_EQUIPO: int
-elemento_debilidad: str
-elemento_fortaleza: str

+alimentarse()
+validarEnergiaAtaque(): bool
+validarEnergiaDefensa(): bool
+objetivos(Tablero, str[ ])
+calculo_ataque(int, str, int): int
+defenderse(Tablero)

Aire

-COSTO_ENERGIA_ATAQUE: int
-COSTO_ENERGIA_DEFENSA: int
-DEBILIDAD: int
-FORTALEZA: int
-ATAQUE: int
-ESCUDO_EXTRA: int
-DEFENSA_EQUIPO: int
-elemento_debilidad: str
-elemento_fortaleza: str

+alimentarse()
+validarEnergiaAtaque(): bool
+validarEnergiaDefensa(): bool
+objetivos(Tablero, str[ ])
+calculo_ataque(int, str, int)
+defenderse(Tablero)

Relationship labels: 1, 1, 1, 1, 1, 1, 1, 1, 1..*, 1, 1, 64, 6, *