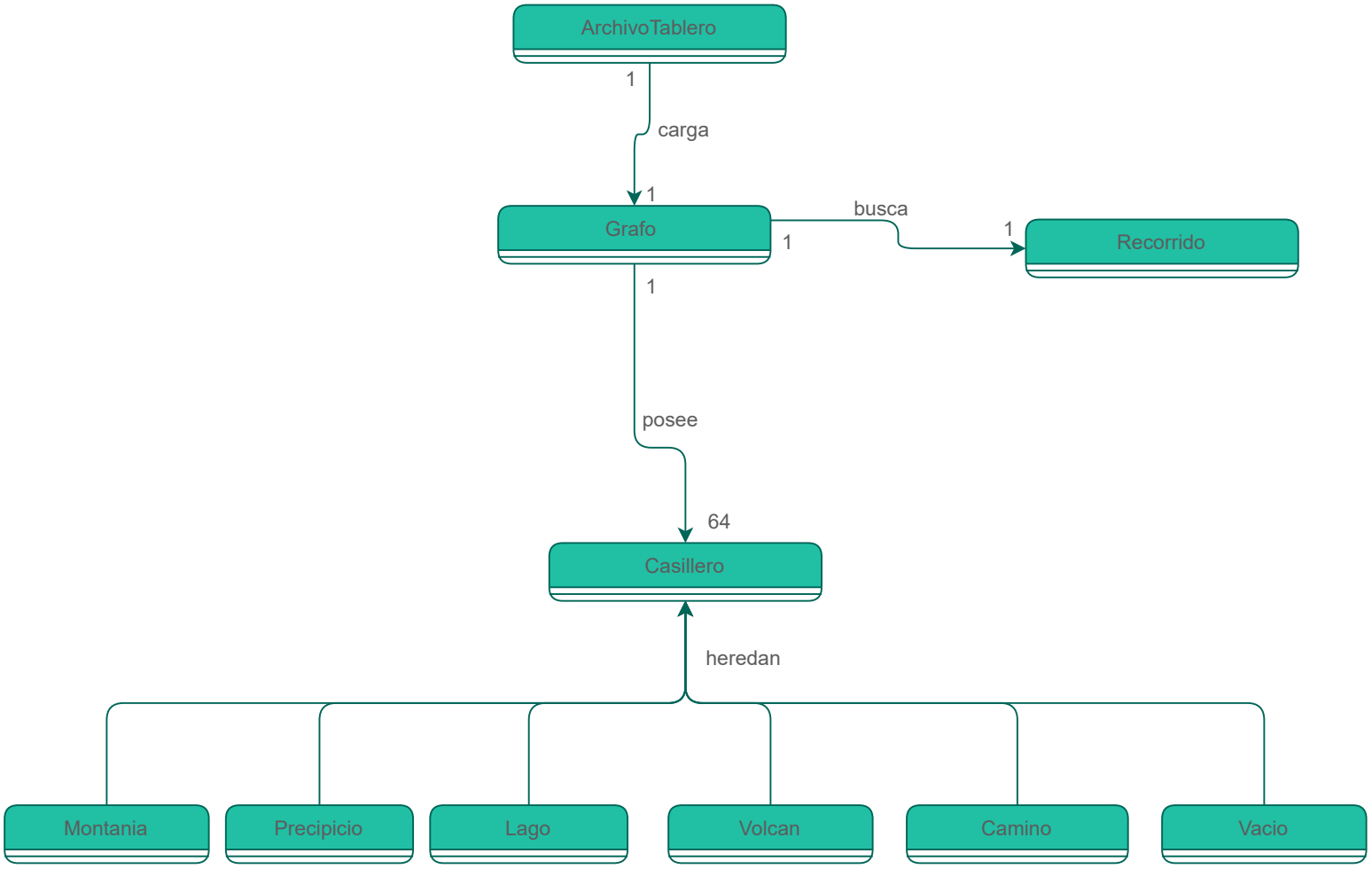
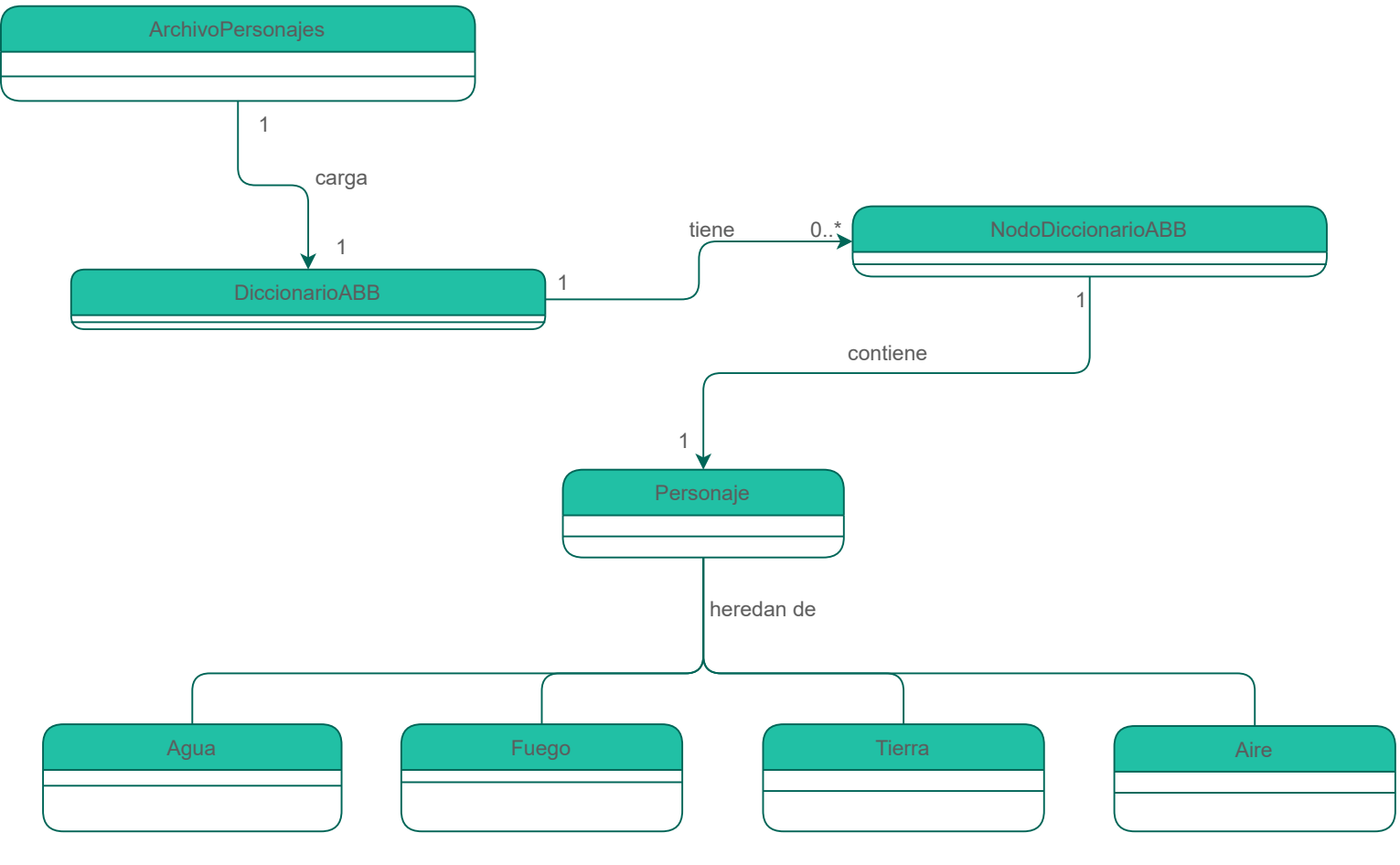
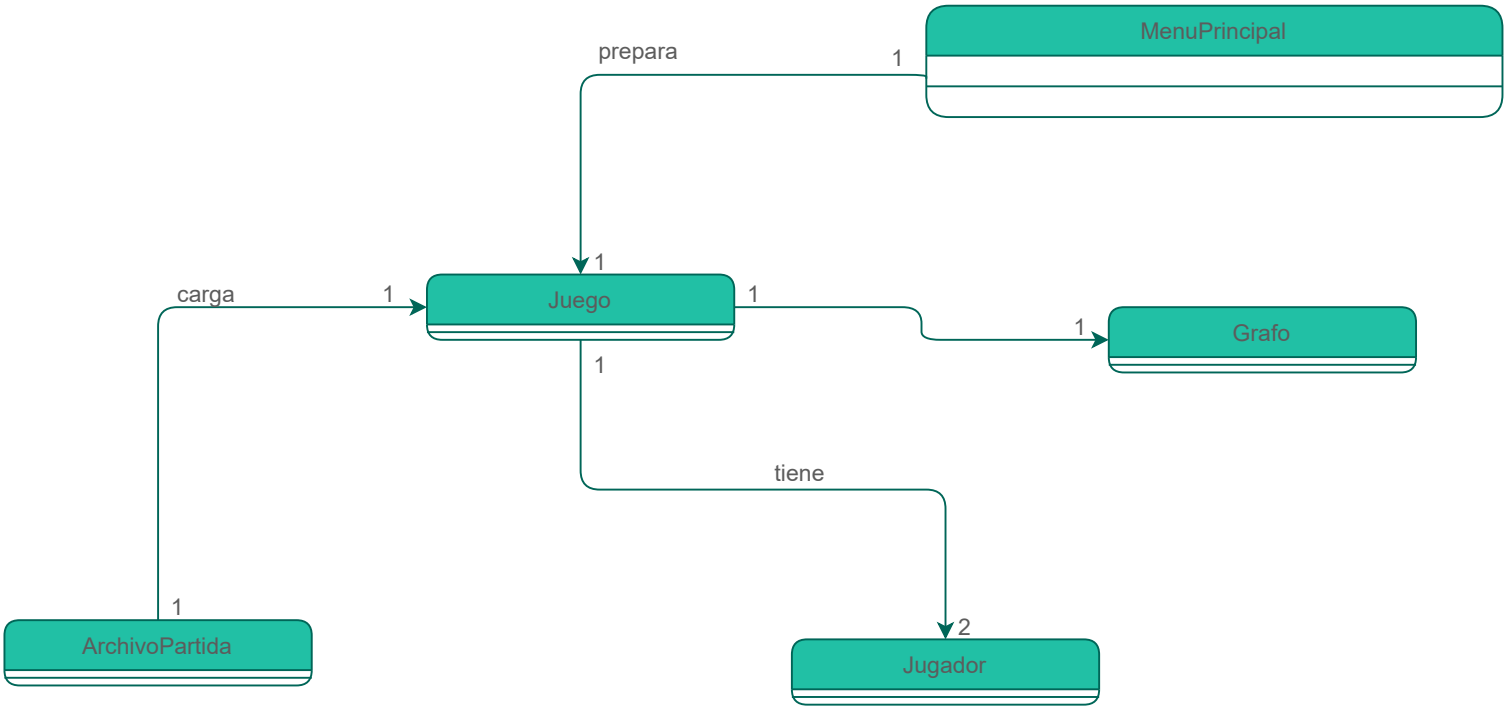

ArchivoTablero
1
carga
1
Grafo
busca
1
1
Recorrido
1
posee
64
Casillero
heredan
Montania
Precipicio
Lago
Volcan
Camino
Vacio


ArchivoPersonajes
1
carga
1
DiccionarioABB
1
tiene
0..*
NodoDiccionarioABB
1
contiene
1
Personaje
heredan de
Agua
Fuego
Tierra
Aire


MenuPrincipal
prepara
1
1
Juego
carga
1
1
1
Grafo
1
tiene
1
ArchivoPartida
2
Jugador

<< *clase* >>
Agua

+ vecesAlimentado : int

+ Agua () string : nombre
+ string obtenerElemento () : override;
+ bool defensaActivada () : override;
+ void desactivarDefensa () : override;
+ void mostrar () : override;
+ void alimentar () : averride;
+ void atacar(array<Personaje* , MAX_PERSONAJES> enemigos) override;
+ void defender(array<Personaje* , MAX_PERSONAJES> enemigos) override
- void aumentarEnergia ();
- void curarAliado (Personaje* aliado);
- void mostrarMsDefensa () : int nuevoVidaPropia;

<< *clase* >>
Aire

+ Aire () string : nombre
+ string obtenerElemento () : override;
+ bool defensaActivada () override;
+ void desactivarDefensa () override;
+ void mostrar () : override;
+ void alimentar () : override;
+ void atacar (array<Personaje*, MAX_PERSONAJES> enemigos) override;
+ void defender (array<Personaje*, MAX_PERSONAJES>aliados) override;
+ int calcularAtkEntrante(Personaje* enemigo);
+ void aumentarEnergia ();

<< *clase* >>
Fuego

+ Fuego() string : nombre
+ string obtenerElemento () : override;
+ bool defensaActivada () override;
+ void desactivarDefensa () override;
+ void mostrar () : override;
+ void alimentar () : override;
+ void atacar (array<Personaje*, MAX_PERSONAJES> enemigos) override
+ void defender (array<Personaje*, MAX_PERSONAJES>aliados) override
+int calcularAtkEntrante(Personaje* enemigo);
- void aumentarEnergia ();
- void aumentarVida ();
- bool vidaMaxima ();

<< *clase* >>
Tierra

+ Tierra() string : nombre
+ string obtenerElemento () : override;
+ void mostrar () : override;
+ void alimentar () : override;
+ void atacar (array<Personaje*, MAX_PERSONAJES> enemigos)
+ void defender (array<Personaje*, MAX_PERSONAJES>aliados, Grafo&tablero)
+ int calcularAtkEntrante(Personaje* enemigo);
- void aumentarEnergia ();
- bool energiaMaxima ();

<< *clase* >>
ArchivoPartida

- ifstream archivoEntrada;
- ofstream archivoSalida;

+ ArchivoPartida()
+ bool hayPartidaGuardada ();
+ void cargarPartida (); Juego & juego;
+ void guardarPartida () Juego* juego, int turno;
+ void eliminarArchivo ();
+ ~ArhcivoPartida ();
- void cargarPersonajes (Grafo& tablero, Jugador &jugador)
- Personaje* crearPersonaje () string, string, string, string, string, array<int,2> posicion
- void guardarPersonajes() array<Personaje* MAX_PERSONAJES> personajes, int cantPersonajes

<< *clase* >>
ArchivoPersonaje

- ifstream archivo;

+ ArchivoPersonajes ();
+ bool estaAbierto ();
+ void procesarArchivo (DiccionarioABB<string, Personaje*>& diccionario)
+ ~ArchivoPersonaje ();
- Personaje* crearPersonaje () string, string, string, string);

<< *clase* >>
ArchivoTablero

- ifstream archivo;

+ ArchivoTablero ();
+ bool estaAbierto ();
+ void cargarTablero (Grafo tablero);
+ ~ArchivoTablero ();
- Casillero* crearCasillero (char tipo);

<< *clase* >>
Menu principal

- opcion: int

+ void iniciar()
+ void pausar();
+ void interfazPrincipal(Juego& juego, string, Personaje*>&diccionario)
- void limpiarPantalla();
- void mostrarOpciones();
- void pedirOpcion ();
- void AgregarPersonaje (DiccionarioABB<string, Personaje*> diccionario);
- void eliminarPersonaje (DiccionarioABB<string, Personaje*> diccionario);
- void mostrarPersonaje (DiccionarioABB<string, Personaje*> diccionario);
- void detallePersonaje (DiccionarioABB<string, Personaje*> diccionario);
- void pedirElemento (); string elemento;
- void pedirNomre (); string nombre;;
- void validarElemento (); string elemento;
- void pasarAMinuscula () string : cadena;
- Personaje* crearPersonaje(string, string);
- void cargarPersonajes (jugador1. jugador2, DiccionarioABB<string, Personaje*>diccionario);
- void menuJuego (jugador1. jugador2, DiccionarioABB<string, Personaje*>diccionario);
- void mostrarJuego ();
- void seleccionarPersonaje (jugador1, jugador2, DiccionarioABB<string, Personaje*>diccionario);
- void asignarCodigo (Personaje* personaje, int numPersonaje)

<< *clase* >>
Grafo

- agua int [CANT_VERTICE][CANT_VERTICE]
- aire int [CANT_VERTICE][CANT_VERTICE]
- fuego int [CANT_VERTICE][CANT_VERTICE]
- tierra int [CANT_VERTICE][CANT_VERTICE]
- vertices array<Casillero*, CANT_VERTICES>

+ void inicializarMatrices ();
+ void asignarVertice () Casillero* vertice, array <int , 2 > posicion
+ Personaje* obtenerPersoanje (matriz < int , 2 > posicion);
+ bool estaVacio () matriz <int, 2> posicion
+ void moverPersonaje (Personaje * personaje, arreglo < int , 2 > posInicial, arreglo < int , 2 > posFinal)
+ void eliminarPersonaje (array < int , 2> posicion)
+ void mostrar ()
- void inicializarMatriz () matrizPesos int [CANT][CANT], & elemento string
- void rellenarMatriz () matriz int [CANT][CANT]
- Recorrido dijkstra (int origen, int destino, int matrizPesos [CANT_VERTICES] [CANT_VERTICES])
- int distanciaMinima (arreglo <int, CANT> distancias, arreglo <bool , CANT > visitados)
- int calcularIndice (matriz < int , 2 > posicion);

<< *clase* >>
Personaje

nombre : string
escudo : int
vida : int
energia : int
array <int, 2> posicion
char codigo

+ Personaje (stirng nombre)
+ void asignarEscudo (int escudo)
+ void asignarVida (int vida)
+ void asignarEnergia (int energia)
+ void asignarPosicion (array <int , 2> posicion)
+ void asignarCodigo (char codigo)
+ string obtenerNombre ()
+ int obtenerEscudo ()
+ int obtenerVida ()
+ int obtenerEnergia ()
+ array <int , 2> obtenerPosicion ()
+ char obtenerCodigo ()
+ array <int , 2> pedirCoordenada ()
+ bool energiaSuficiente (int energiaNecesaria)
+ void vidaCeroEnergia ()
+ virtual string obtenerElemento ()
+ virtual void mostrar ()
+ virtual void alimentar ()
+ virtual void aumentarEnergia ()
+ virtual void defensaActivada ()
+ virtual void desactivarDefensa ()
+ virtual void atacar (array <Personaje*, MAX_PERSONAJES> enemigos)
+ virtual void defender(array <Personaje*, MAX_PERSONAJES> enemigos)
+ virtual int calcularAtkEntrante (Personaje* atacante)
+ virtual ~ Personaje ()
# void mostrarAtributos ()
# bool esValida (string & fila, string & columna)
# void restarVida (Personaje*& enemigo)
# bool energiaMaxima ()
# int calcularAtkEntranteTierra (Personaje* enemigo)

<< *clase* >>
Jugador

- array <Personaje*, MAX_PERSONAJES> personajes
- cantidadPersonajes : int
- opcion : int

+ jugador ()
+ bool peronajesCargados ()
+ void asignarPersonaje (Personaje* personaje)
+ array <Personaje*, MAX_PERSONAJES> obtenerPersonaje()
+ obtenerCantPersonajes () : int
+ void posicionamientoPersonaje (Grafo*& tablero, int i)
+ bool quiereSalir ()
+ void chequearBajas (Grafo*& tablero)
+ void jugar (Grafo*& tablero, array<Personaje*, MAX_PERSONAJES> enemigos)
+ bool todoMuertos ()
- void alimentarMover (Personaje*& personaje, Grafo*& tablero)
- void mostrarOpcionesAM ()
- void pedirOpcion ()
- void fmoverPersonaje (Personaje*& personaje, grafo*& tablero)
- void validarMovimiento(Grafo*& tablero, array<int,2> posInicial, array<int,2>& posFinal, Personaje* personaje, int &energiNecesaria)
- void defenderAtacar (Personajes*& personaje, Grafo*& tablero, array<Personaje*, MAX_PERSONAJES>enemigos)
- void mostarOpcionesDA ()
- void defensaAire (Personaje*& personaje, Grafo*& tablero)
- void eliminarPersonaje (int i , Grafo*& tablero)

<< *clase* >>
Recorrido

- energiaGastada : int
- vector<int> caminoTomado

+ void asignarEnergiaGastada (int energiaGastada)
+ int obtenerEnergiaGastada ()
+ void agregarPosicion (int indice)
+ void mostrar ()

<< *clase* >>
Juego

- Jugador jugador1, jugador2
- Grafo tablero
- turno int corto
- salir bool

+ Juego ();
+ void iniciarPartida ()
+ void reanudarPartida ()
+ void asignarTurno () breve int turno
+ Jugador obtenerJugador1 ()
+ Jugador obtenerJugador2 ()
+ Grafo obtenerTablero ()
- vacio pausar ()
- void limpiarPantalla ()
- void posicionarPersonajes ()
- vacio cicloPrincipal ()
- bool terminado ()
- void mostrarGanador ()

<< *clase* >>
Casillero

- Personaje* personaje

+ void asignarPersonaje (Personaje* personaje)
+ Personaje* obtenerPersonaje ()
+ bool estaVacio ()
+ virtual int obtenerCosto () elemento : string
+ virtual char obtenerCodigo ()
+ virtual ~Casillero ()