

tivo

Original

«Abstract»
FabricaSprite
String
ta(): Sprite
dor(): Sprite
onioRojo(): Sprite
amarillo(): Sprite
lDeFuego(): Sprite
oaza(): Sprite
asma(): Sprite
): Sprite
): Sprite
(): Sprite
(): Sprite
leras(): Sprite
des(): Sprite
desDestructibles(): Sprite
Resbaladizo(): Sprite
paPinchos(): Sprite
ica(): Sprite
es(): Sprite
radiza(): Sprite
era(): Sprite
akichi(): Sprite

FabricaEntidades
- fabrica: FabricaSprite
+ getSilueta(): Sprite
+ getJugador(x: int, y: int): Jugador
+ getDemonioRojo(x: int, y: int): DemonioRojo
+ getTrollAmarillo(x: int, y: int): TrollAmarillo
+ getRanaDeFuego(x: int, y: int): RanaDeFuego
+ getCalabaza(x: int, y: int): Calabaza
+ getFantasma(x: int, y: int): Fantasma
+ getAzul(x: int, y: int): BotellaAzul
+ getRoja(x: int, y: int): BotellaRoja
+ getVerde(x: int, y: int): BotellaVerde
+ getFruta(x: int, y: int): Fruta
+ getVida(x: int, y: int): VidaExtra
+ getEscaleras(x: int, y: int): Escaleras
+ getParedes(x: int, y: int): Paredes
+ getParedesDestructibles(x: int, y: int): ParedDestructible
+ getSueloResbaladizo(x: int, y: int): SueloResbaladizo
+ getTrampaPinchos(x: int, y: int): TrampaPinchos
+ getPlataforma(x: int, y: int): Plataforma
+ getMoviles(x: int, y: int): PlataformaMovil
+ getQuebradiza(x: int, y: int): PlataformaQuebradiza
+ getMoghera(x: int, y: int): Moghera
+ getKamakichi(x: int, y: int): Kamakichi

ParserNivel
- fabrica: FabricaEntidades
+ generarNivel(modo:int): Nivel
- cargarNivel(archivo:String, modo:int): Nivel
+ obtenerArchivoNivel(modo:int):String

int, alto: int): void

Rosa: incor
Azul: faltan
Verde: dud
Naranja: si



Preguntar por los niveles y los sprites

nsistencias entre los 2 diagramas(o revisar posible problema)
ordenar
as
stema de orden

+ transparentarBoton(JButton boton)

**<<Interface>>
PositionList**

```
+ size: int
+ isEmpty():boolean
+ first(): Position<E>
+ last: Position<E>
+ next(p: Position<E>): Position<E>
+ post(p: Position<E>): Position<E>
+ addFirst(p: Position<E>): void
+ addLast(p: Position<E>): void
+ addAfter(p: Position<E>, e: E): void
+ addBefore(p: Position<E>, e: E): void
+ remove(p: Position<E>): E
+ set(e: Position<E>, e: E): E
+ iterator(): Iterator<E>
+ position(): Iterable<Position<E>>
```



ListaDC

```
# size: int
# head: Node<E>
# tail: Node<E>
```



**<<Interface>>
Entry**

```
+ getKey(): K
+ getValue(): V
```



**<<Interface>>
Position**

```
+ element(): E
```



Entrada

```
# key: K
# value: #V
+ getKey(): K
+ getValue(): V
```

Node

```
+ element: E
+ element(): E
```



```
+ size: int
+ isEmpty():boolean
+ first(): Position<E>
+ last: Position<E>
+ next(p: Position<E>): Position<E>
+ post(p: Position<E>): Position<E>
+ addFirst(p: Position<E>): void
+ addLast(p: Position<E>): void
+ addAfter(p: Position<E>, e: E): void
+ addBefore(p: Position<E>, e: E): void
+ remove(p: Position<E>): E
+ set(e: Position<E>, e: E): E
+ iterator(): Iterator<E>
+ position(): Iterable<Position<E>>
```

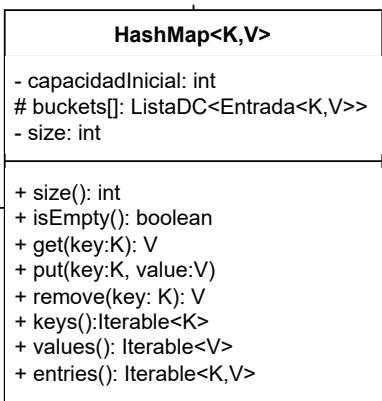
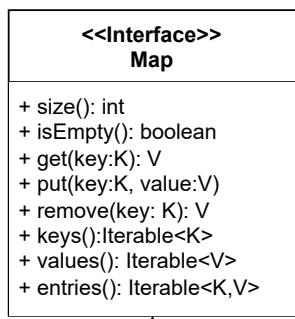
```
+ registrarOyenteBotonClasico()
+ registrarOyenteBotonContraTimer()
+ registrarOyenteBotonSupervivencia()
+ transparentarBoton(jButton)
```

```
+ actualizarLabelsPuntaje(j: Jugador)
+ actualizarLabelsTiempo(j: Jugador)
+ actualizarScrollHaciaJugador(j: Jugador)
+ centrarEnJugador(posXJugador: int)
+ agregarPanelJuegoConFondoYScroll()
+ agregarLabelsEditablesTiempo()
+ agregarLabelsEditablesPuntaje()
+ agregarLabelsEditablesVidas()
+ decorarLabelsTiempo()
+ decorarLabelsPuntaje()
+ decorarLabelsVidas()
```

```
+ agregarBotonInvadir()
+ registrarOyenteBotonReiniciar()
+ registrarOyenteBotonSiguiente()
+ registrarOyenteBotonNuevo()
+ transparentarBoton(jButton)
```

```
n<E>
n<E>
d
d
): void
:E): void

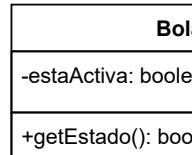
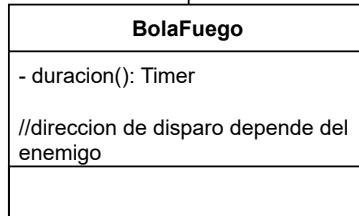
E>>
```



```
ion<E>
ion<E>
oid
oid
:E): void
e:E): void
```

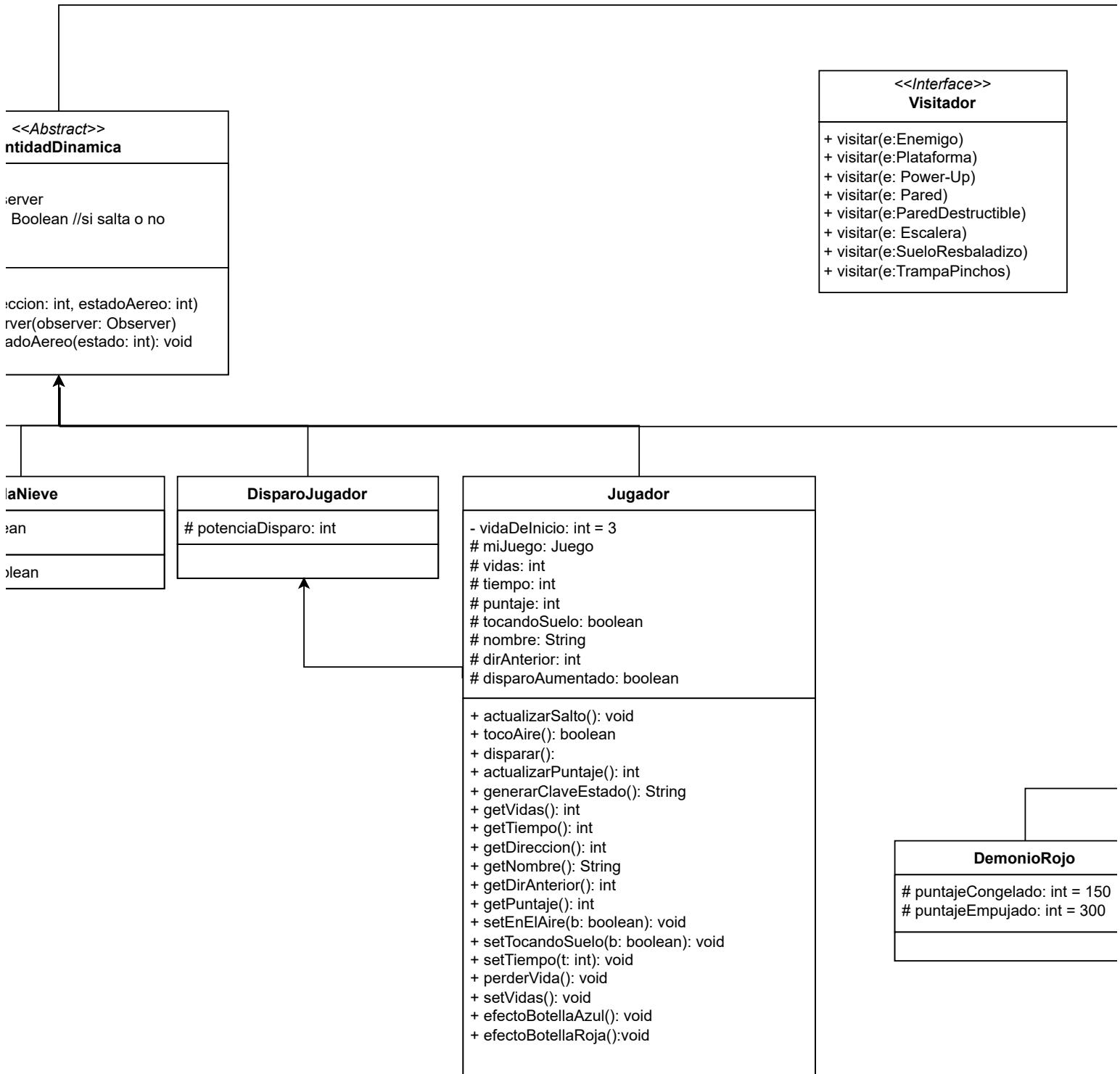
:

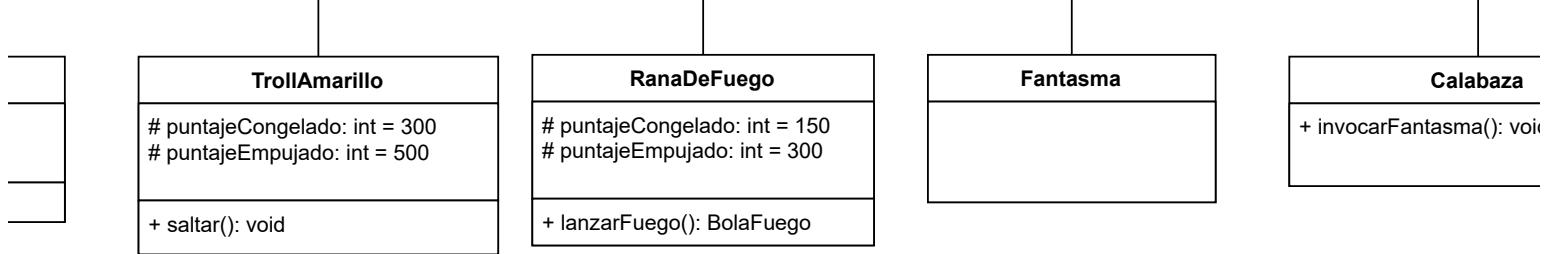
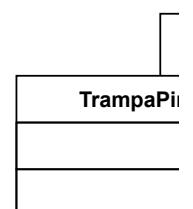
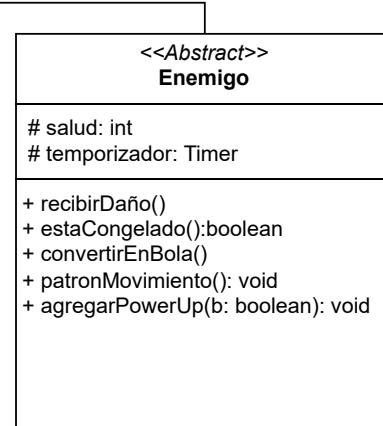
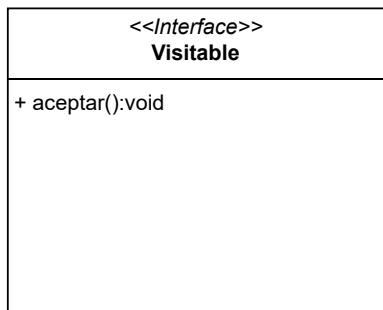
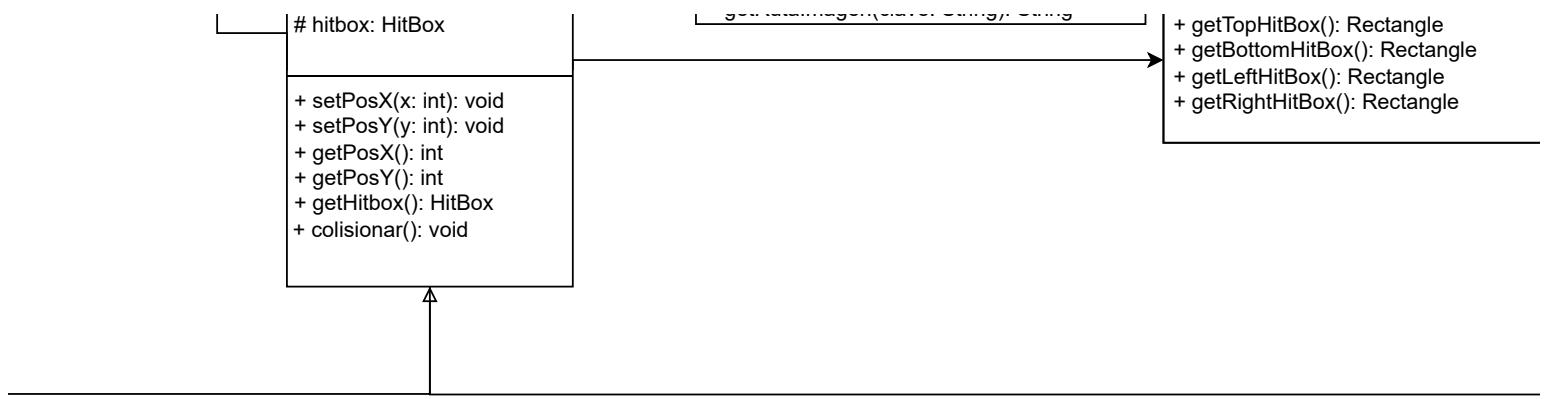
i<E>>

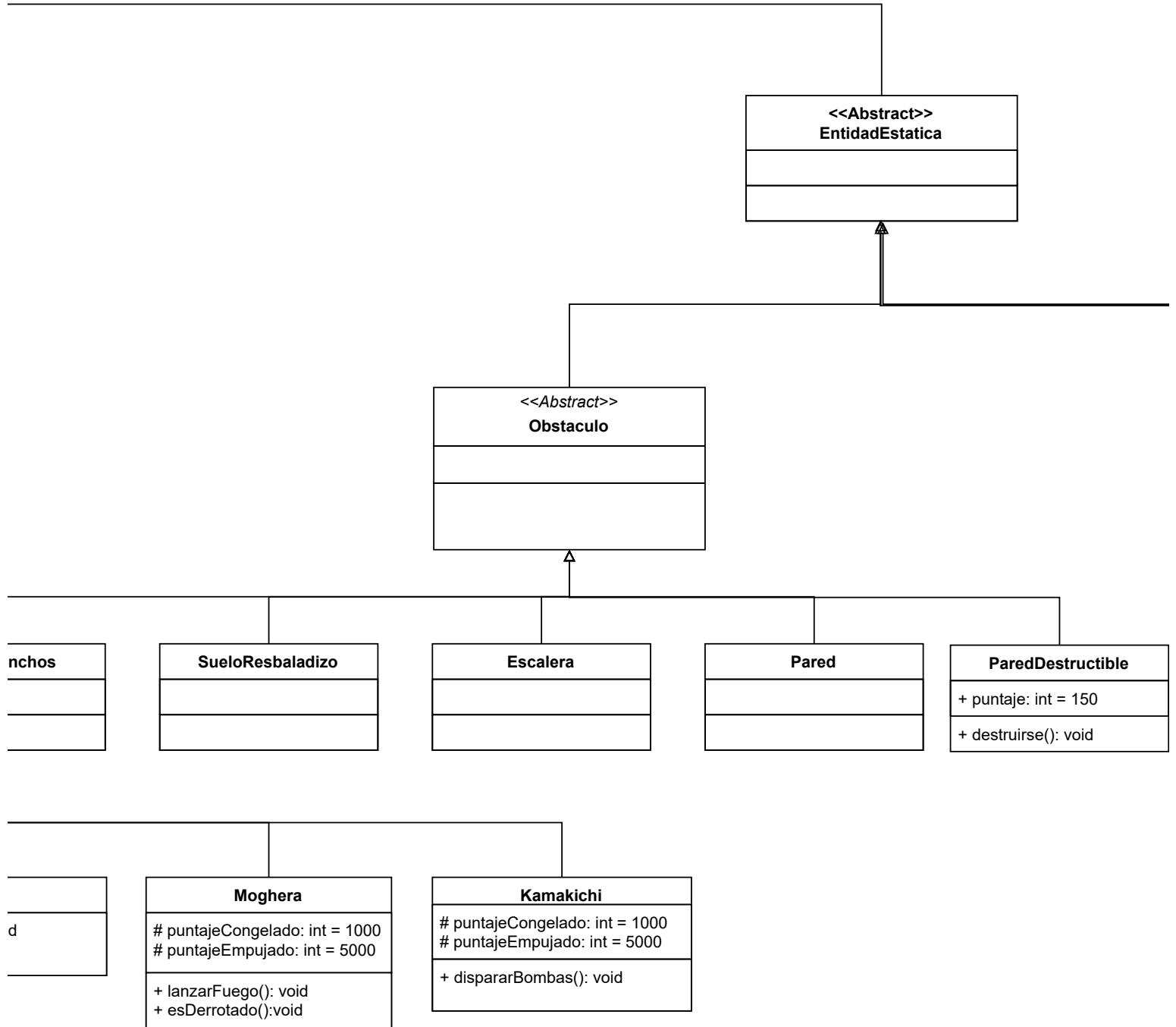


```
    inicio()  
    alir()  
    uevaPartida()  
    on: JButton)
```

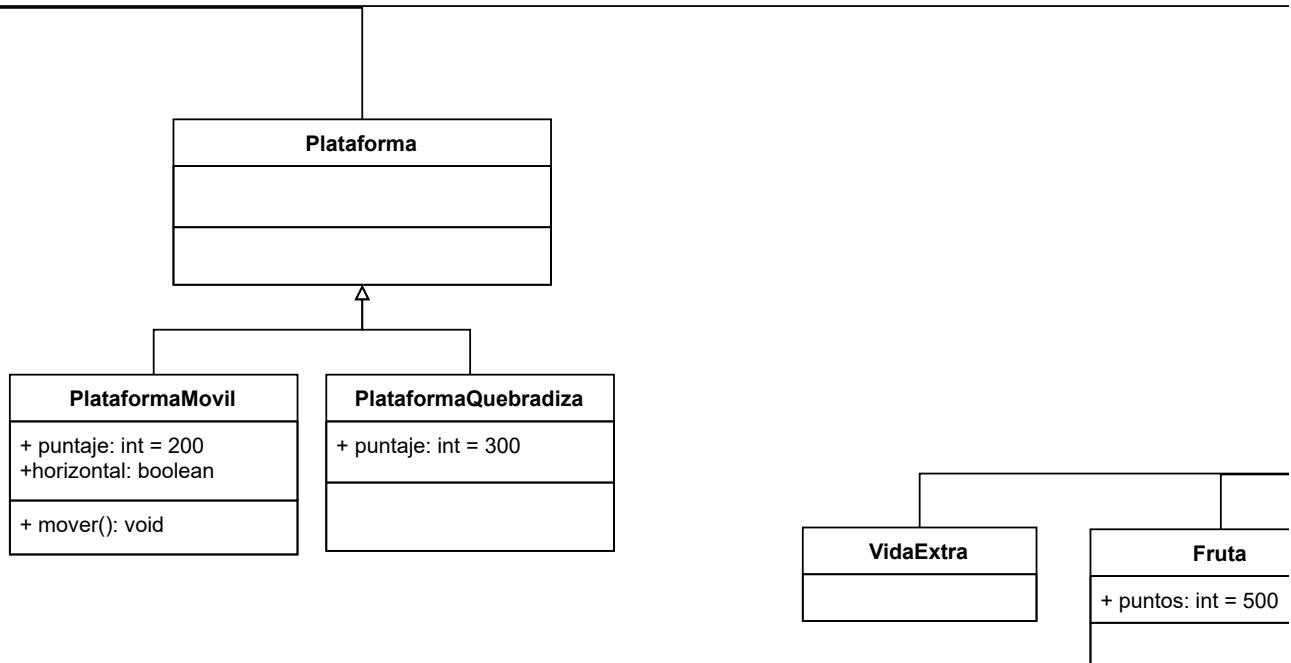
Pendiente: ver como ingresar usuario para el ranking

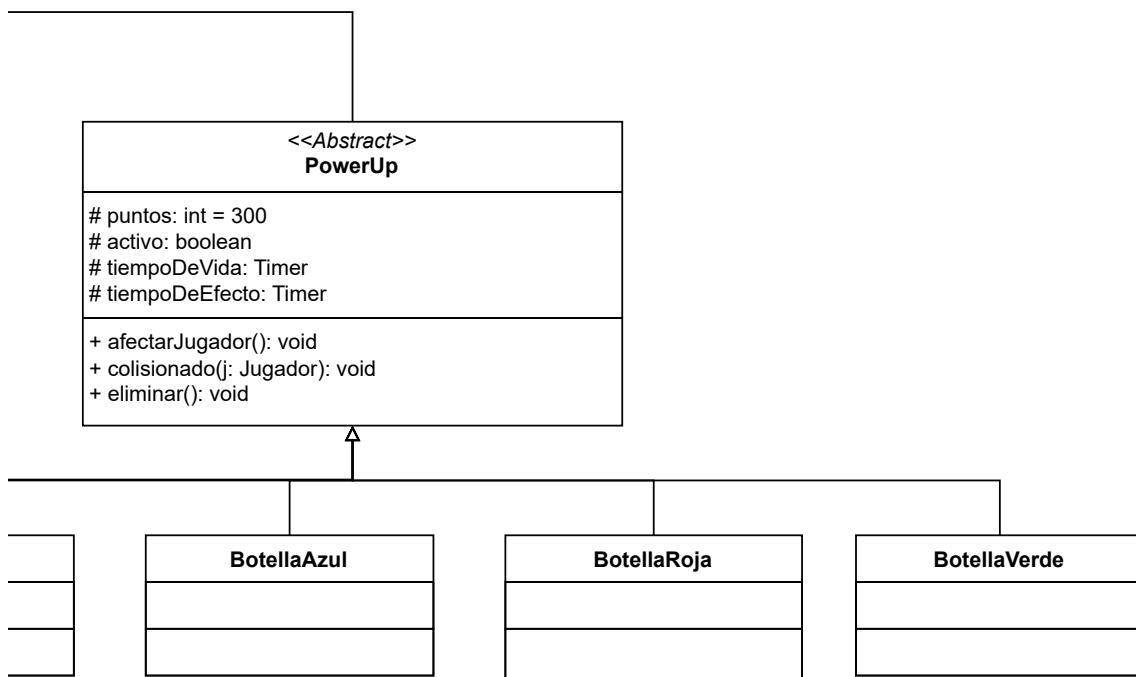






sistema de orden
faltan cosas(aun definiendola)
da si borrar
o del proyecto
ntes: faltan diseñar





<<Interface>> Observer
+ actualizar(): void + actualizar(direccion: int, estadoAereo: int): void

ObserverGrafico
entidadObservada: Entidad
+ actualizar(): void # actualizarImagen(): void # actualizarPosicionTamano(): void

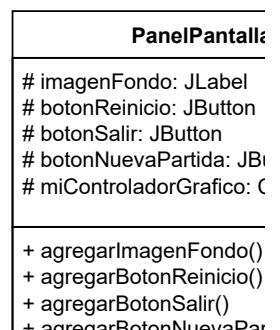
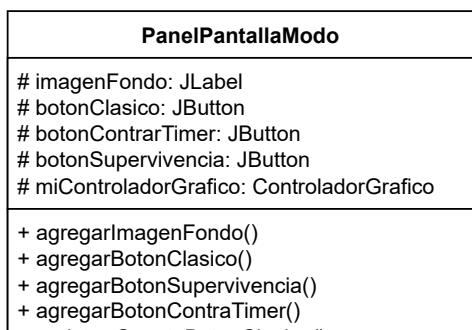
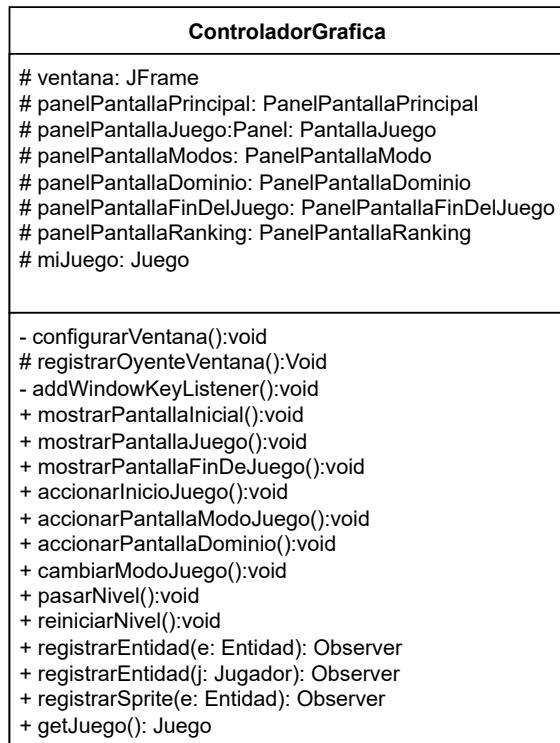
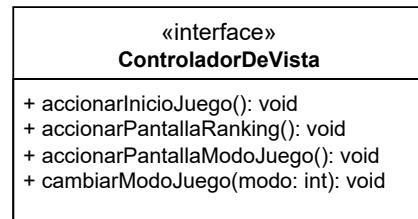
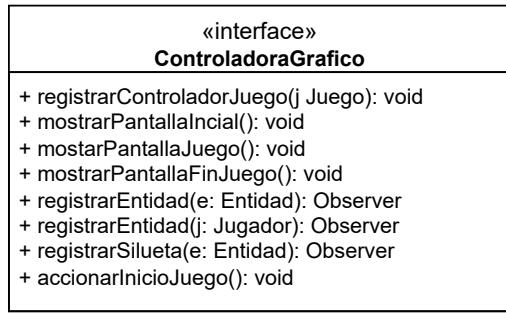
ObserverEntidades
+ actualizar(direccion: int, estadoAereo: int): void

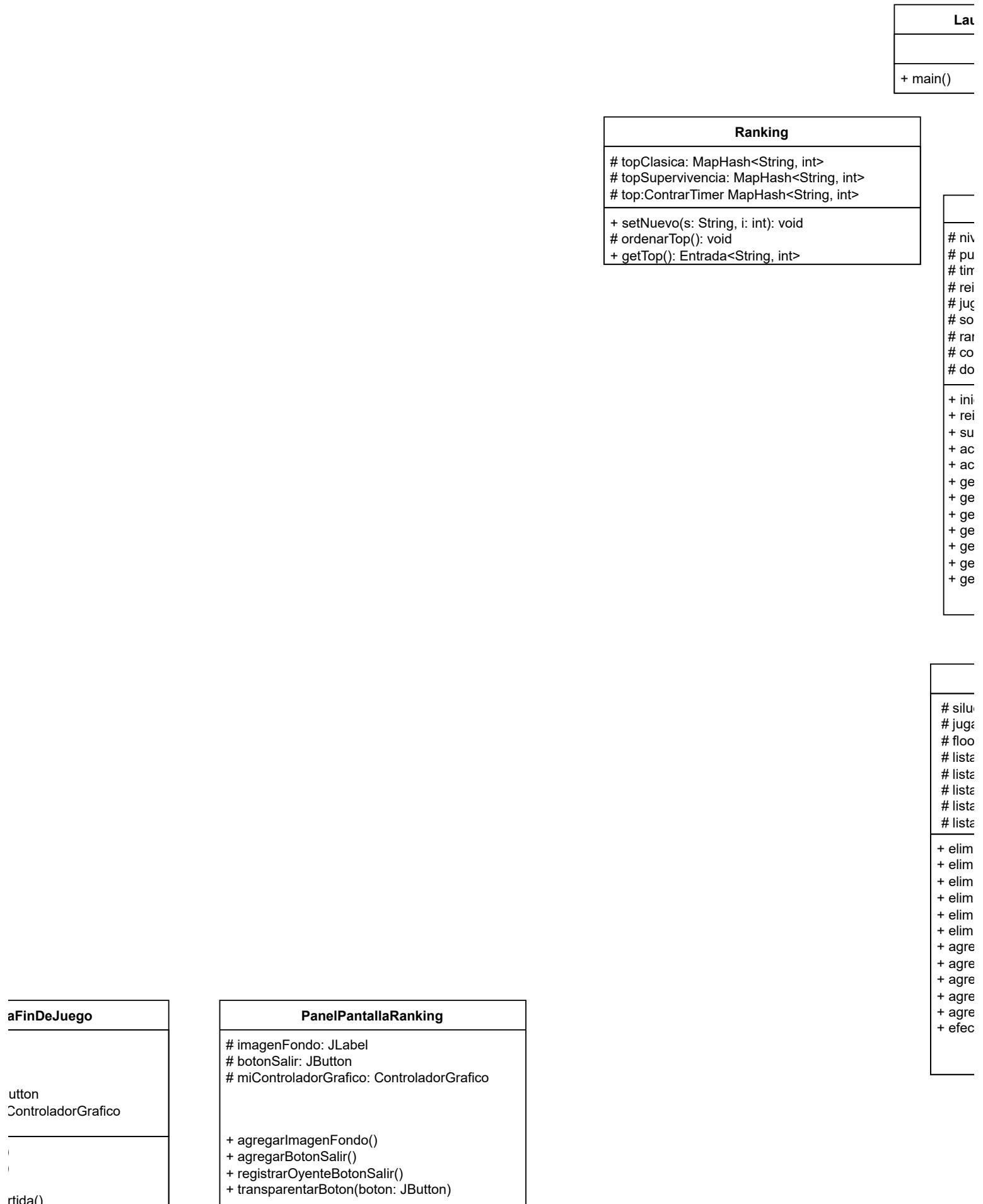
ObserverJugador
panelPantallaJuego: PanelPantallaJuego # jugadorObservado: Jugador
+ actualizar(): void + actualizar(direccion: int, estadoAereo: int): void

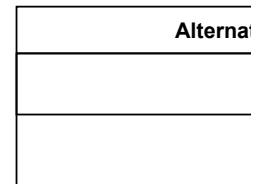
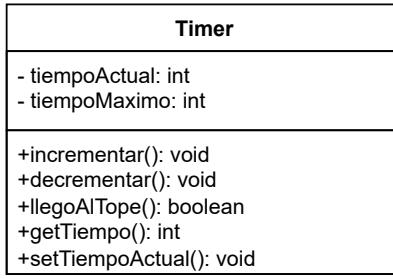
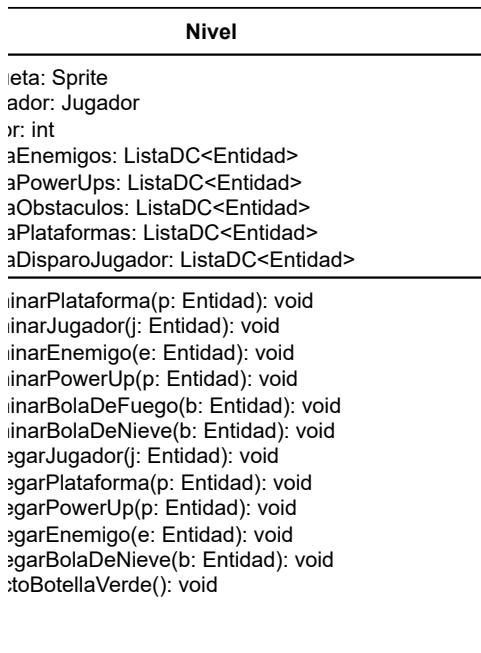
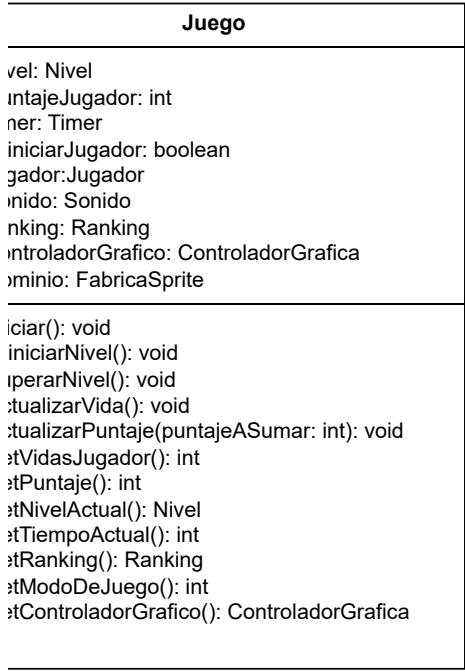
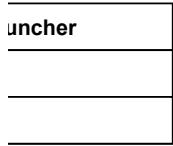
PanelPantallaPrincipal
imagenFondo: JLabel # botonIniciar: JButton # botonRanking: JButton # miControladorGrafico: ControladorGrafico
+ agregarImagenFondo() + agregarBotonIniciar() + agregarBotonRanking() + registrarOyenteBotonIniciar() + registrarOyenteBotonRanking()

PanelPantallaDominio
imagenFondo: JLabel # botonOriginal: JButton # botonAlternativo: JButton # miControladorGrafico: ControladorGrafico

+ agregarImagenFondo() + agregarBotonOriginal() + agregarBotonAlternativo() + registrarOyenteBotonOriginal() + registrarOyenteBotonAlternativo()
+ transparentarBoton(JButton boton)





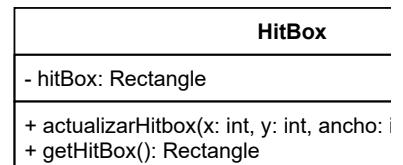
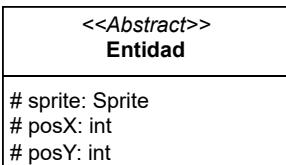


```

# rutaACarpeta:

+ getSpriteSiluet
+ getSpriteJugac
+ getSpriteDemc
+ getSpriteTrollA
+ getSpriteRana
+ getSpriteCalak
+ getSpriteFanta
+ getSpriteAzul(
+ getSpriteRoja(
+ getSpriteVerde
+ getSpriteFruta(
+ getSpriteVida(
+ getSpriteEscal
+ getSpriteParec
+ getSpriteParec
+ getSpriteSuelo
+ getSpriteTram|
+ getSpriteEstati
+ getSpriteMovil
+ getSpriteQueb
+ getSpriteMogh
+ getSpriteKama

```



tivo

Original

«Abstract»
FabricaSprite
String
ta(): Sprite
dor(): Sprite
onioRojo(): Sprite
amarillo(): Sprite
lDeFuego(): Sprite
oaza(): Sprite
asma(): Sprite
): Sprite
): Sprite
(): Sprite
(): Sprite
leras(): Sprite
des(): Sprite
desDestructibles(): Sprite
oResbaladizo(): Sprite
paPinchos(): Sprite
ica(): Sprite
es(): Sprite
radiza(): Sprite
era(): Sprite
akichi(): Sprite

FabricaEntidades
- fabrica: FabricaSprite
+ getSilueta(): Sprite
+ getJugador(x: int, y: int): Jugador
+ getDemonioRojo(x: int, y: int): DemonioRojo
+ getTrollAmarillo(x: int, y: int): TrollAmarillo
+ getRanaDeFuego(x: int, y: int): RanaDeFuego
+ getCalabaza(x: int, y: int): Calabaza
+ getFantasma(x: int, y: int): Fantasma
+ getAzul(x: int, y: int): BotellaAzul
+ getRoja(x: int, y: int): BotellaRoja
+ getVerde(x: int, y: int): BotellaVerde
+ getFruta(x: int, y: int): Fruta
+ getVida(x: int, y: int): VidaExtra
+ getEscaleras(x: int, y: int): Escaleras
+ getParedes(x: int, y: int): Paredes
+ getParedesDestructibles(x: int, y: int): ParedDestructible
+ getSueloResbaladizo(x: int, y: int): SueloResbaladizo
+ getTrampaPinchos(x: int, y: int): TrampaPinchos
+ getPlataforma(x: int, y: int): Plataforma
+ getMoviles(x: int, y: int): PlataformaMovil
+ getQuebradiza(x: int, y: int): PlataformaQuebradiza
+ getMoghera(x: int, y: int): Moghera
+ getKamakichi(x: int, y: int): Kamakichi

ParserNivel
- fabrica: FabricaEntidades
+ generarNivel(modo:int): Nivel
- cargarNivel(archivo:String, modo:int): Nivel
+ obtenerArchivoNivel(modo:int): String

int, alto: int): void

+ transparentarBoton(JButton boton)

**<<Interface>>
PositionList**

+ size: int
+ isEmpty():boolean
+ first(): Position<E>
+ last: Position<E>
+ next(p: Position<E>): Position<E>
+ post(p: Position<E>): Position<E>
+ addFirst(p: Position<E>): void
+ addLast(p: Position<E>): void
+ addAfter(p: Position<E>, e: E): void
+ addBefore(p: Position<E>, e: E): void
+ remove(p: Position<E>): E
+ set(e: Position<E>, e: E): E
+ iterator(): Iterator<E>
+ position(): Iterable<Position<E>>

**<<Interface>>
Entry**

+ getKey(): K
+ getValue(): V

**<<Interface>>
Position**

+ element(): E

ListaDC

size: int
head: Node<E>
tail: Node<E>

Entrada

key: K
value: #V

+ getKey(): K
+ getValue(): V

Node

+ element: E

+ element(): E

+ size: int
+ isEmpty():boolean
+ first(): Position<E>
+ last: Position<E>
+ next(p: Position<E>): Position<E>
+ post(p: Position<E>): Position<E>
+ addFirst(p: Position<E>): void
+ addLast(p: Position<E>): void
+ addAfter(p: Position<E>, e: E): void
+ addBefore(p: Position<E>, e: E): void
+ remove(p: Position<E>): E
+ set(e: Position<E>, e: E): E
+ iterator(): Iterator<E>
+ position(): Iterable<Position<E>>

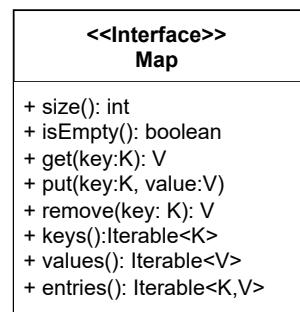
```
+ registrarOyenteBotonClasico()
+ registrarOyenteBotonContraTimer()
+ registrarOyenteBotonSupervivencia()
+ transparentarBoton(jButton: JButton)
```

```
+ actualizarLabelsPuntaje(j: Jugador)
+ actualizarLabelsTiempo(j: Jugador)
+ actualizarScrollHaciaJugador(j: Jugador)
+ centrarEnJugador(posXJugador: int)
+ agregarPanelJuegoConFondoYScroll()
+ agregarLabelsEditablesTiempo()
+ agregarLabelsEditablesPuntaje()
+ agregarLabelsEditablesVidas()
+ decorarLabelsTiempo()
+ decorarLabelsPuntaje()
+ decorarLabelsVidas()
```

```
+ agregarBotonInvadir()
+ registrarOyenteBotonRe()
+ registrarOyenteBotonSe()
+ registrarOyenteBotonNi()
+ transparentarBoton(bot)
```

```
n<E>
n<E>
d
d
): void
:E): void
```

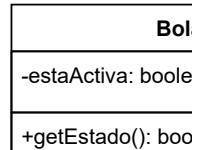
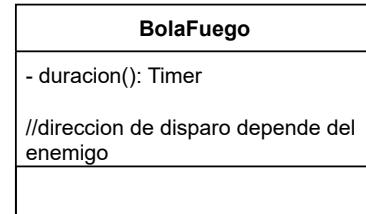
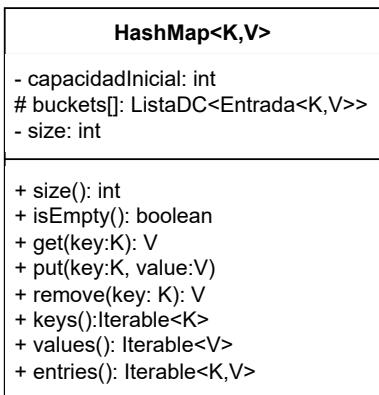
E>>



```
ion<E>
ion<E>
oid
oid
:E): void
e:E): void
```

:>

i<E>>



```

    inicio()
    alir()
    uevaPartida()
    on: JButton

```

<<Abstract>>
ntidadDinamica

server
Boolean //si salta o no

ccion: int, estadoAereo: int)
ver(observer: Observer)
adoAereo(estado: int): void

<<Interface>>
Visitador

- + visitar(e:Enemigo)
- + visitar(e:Plataforma)
- + visitar(e: Power-Up)
- + visitar(e: Pared)
- + visitar(e:ParedDestructible)
- + visitar(e: Escalera)
- + visitar(e:SueloResbaladizo)
- + visitar(e:TrampaPinchos)

laNieve

an

olean

DisparoJugador

potenciaDisparo: int

Jugador

- vidaDelInicio: int = 3
- # miJuego: Juego
- # vidas: int
- # tiempo: int
- # puntaje: int
- # tocandoSuelo: boolean
- # nombre: String
- # dirAnterior: int
- # disparoAumentado: boolean

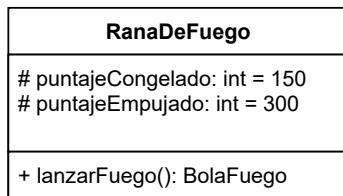
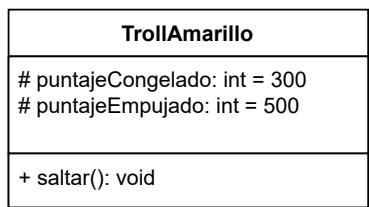
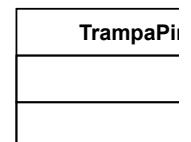
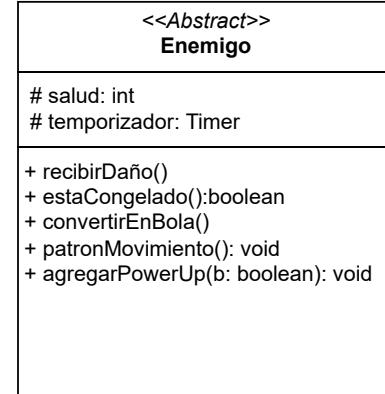
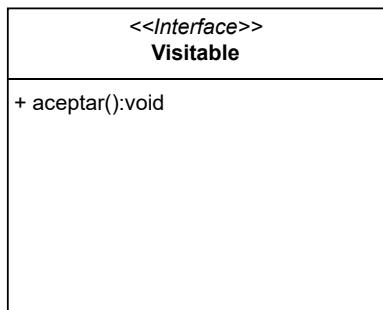
- + actualizarSalto(): void
- + tocoAire(): boolean
- + disparar():
- + actualizarPuntaje(): int
- + generarClaveEstado(): String
- + getVidas(): int
- + getTiempo(): int
- + getDireccion(): int
- + getNombre(): String
- + getDirAnterior(): int
- + getPuntaje(): int
- + setEnElAire(b: boolean): void
- + setTocandoSuelo(b: boolean): void
- + setTiempo(t: int): void
- + perderVida(): void
- + setVidas(): void
- + efectoBotellaAzul(): void
- + efectoBotellaRoja():void

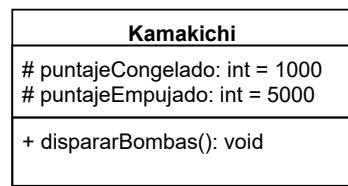
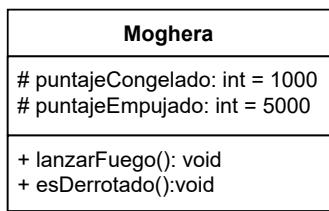
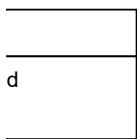
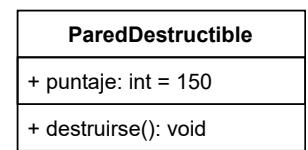
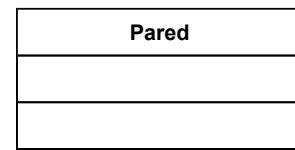
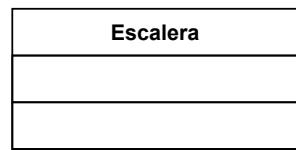
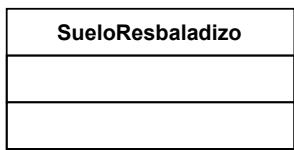
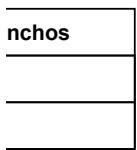
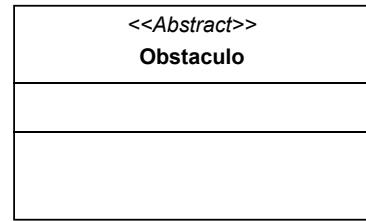
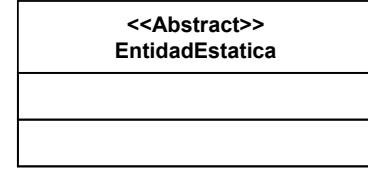
DemonioRojo

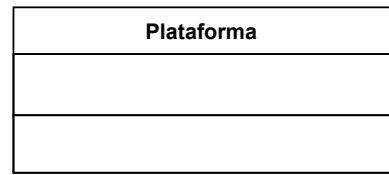
puntajeCongelado: int = 150
puntajeEmpujado: int = 300

```
# hitbox: HitBox
+ setPosX(x: int): void
+ setPosY(y: int): void
+ getPosX(): int
+ getPosY(): int
+ getHitbox(): HitBox
+ colisionar(): void
```

```
+ getTopHitBox(): Rectangle
+ getBottomHitBox(): Rectangle
+ getLeftHitBox(): Rectangle
+ getRightHitBox(): Rectangle
```







PlataformaMovil
+ puntaje: int = 200
+horizontal: boolean
+ mover(): void

PlataformaQuebradiza
+ puntaje: int = 300

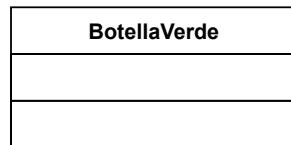
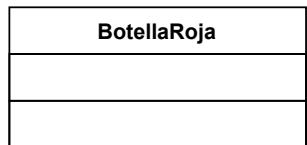
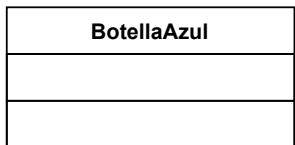
VidaExtra

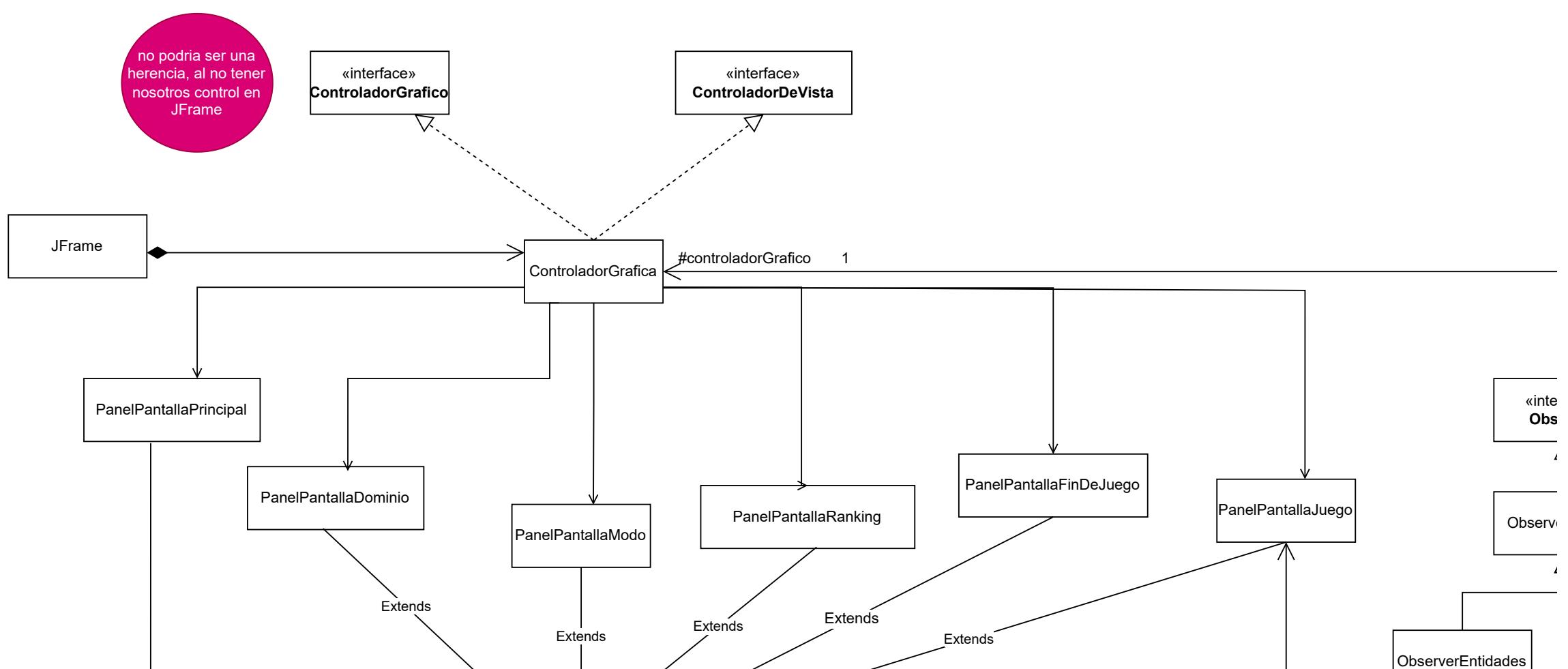
Fruta
+ puntos: int = 500

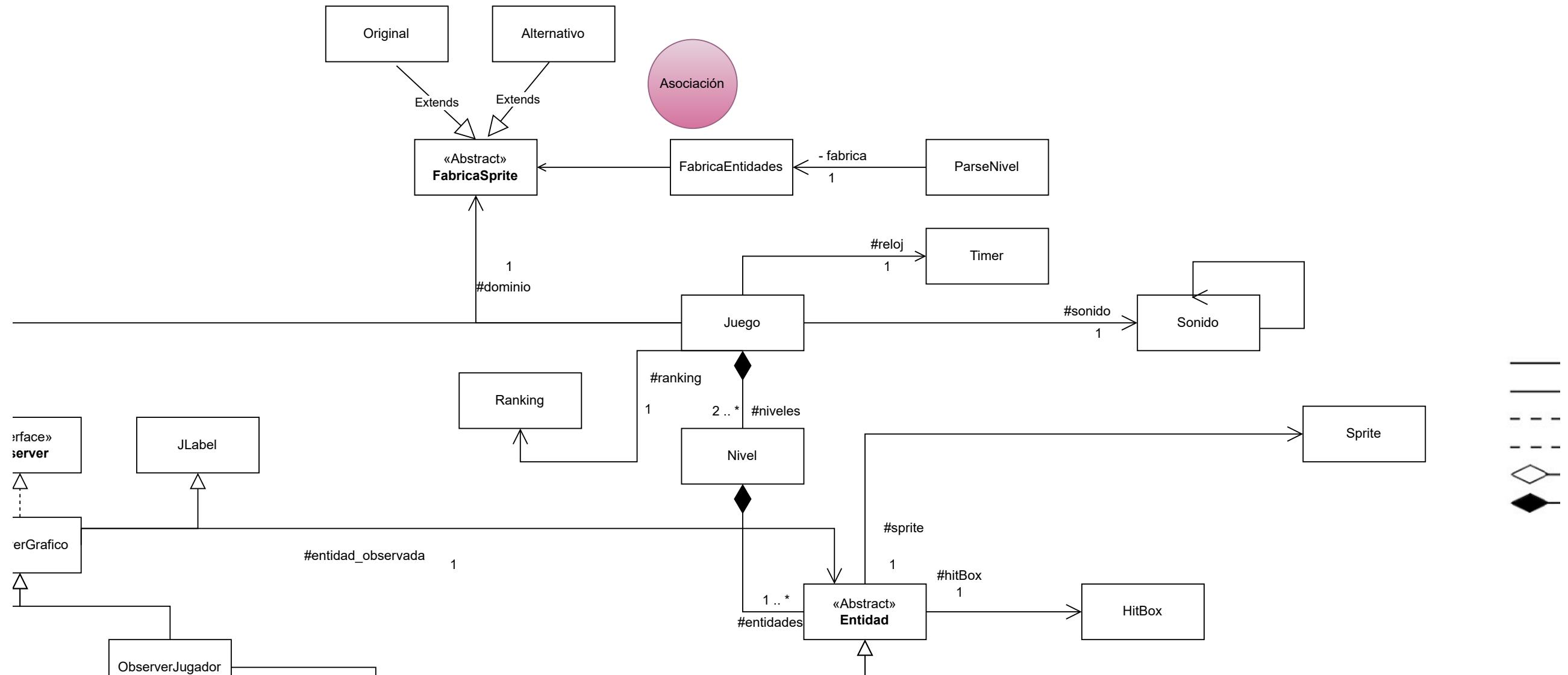
```
<<Abstract>>
PowerUp

# puntos: int = 300
# activo: boolean
# tiempoDeVida: Timer
# tiempoDeEfecto: Timer

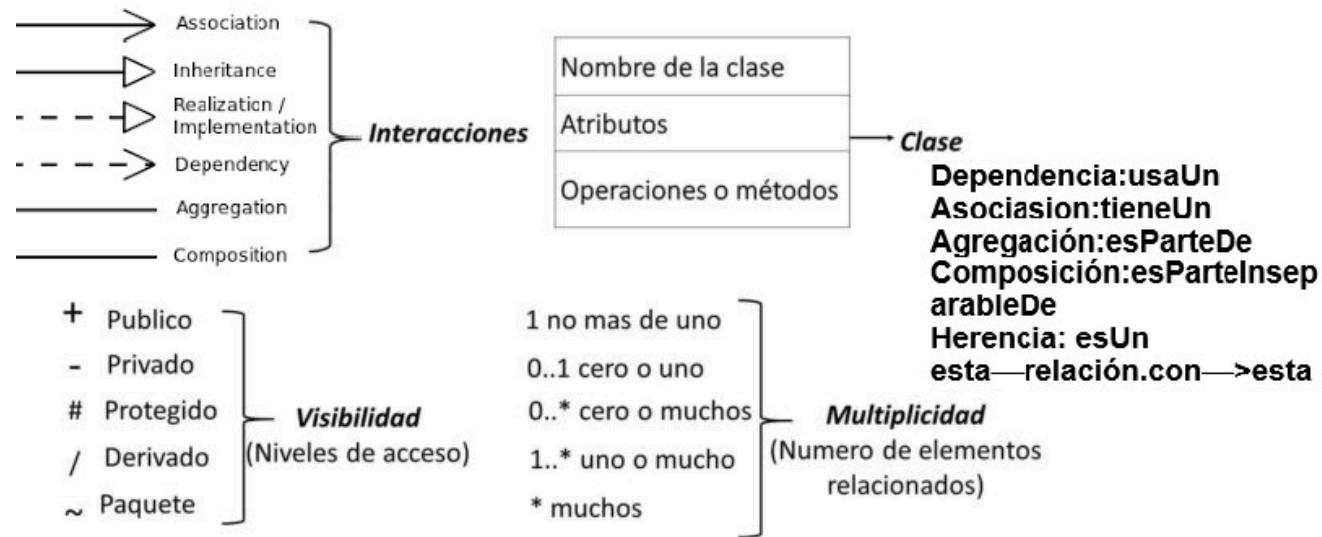
+ afectarJugador(): void
+ colisionado(j: Jugador): void
+ eliminar(): void
```

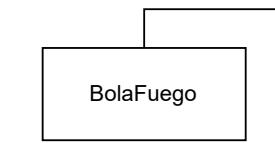
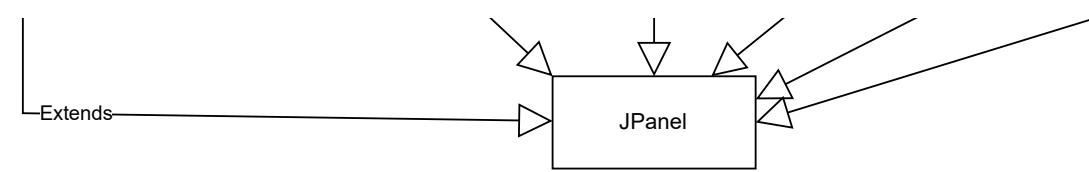


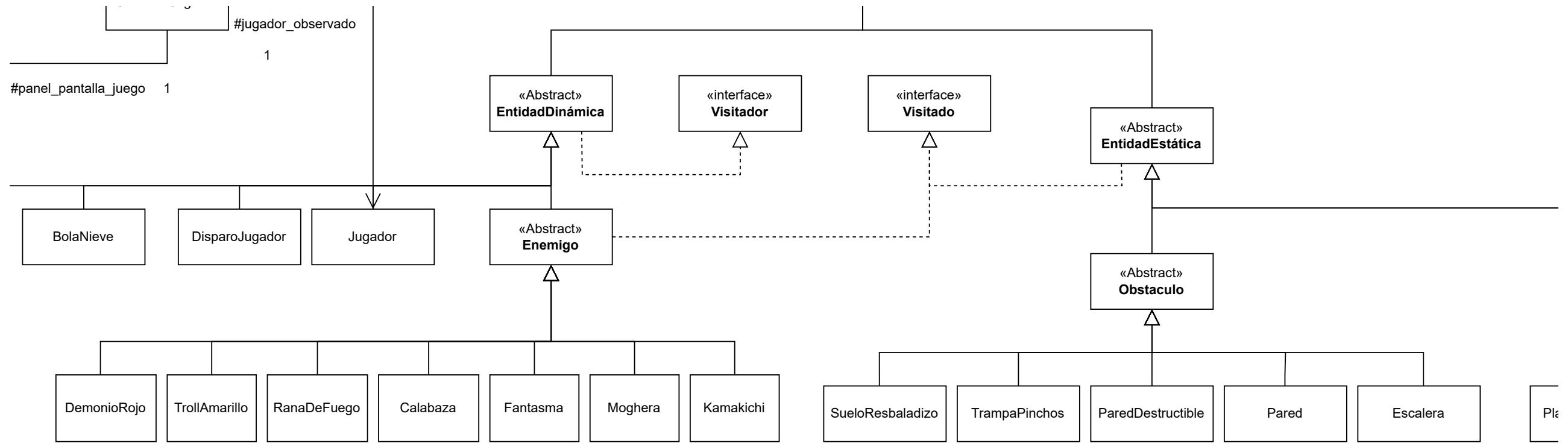


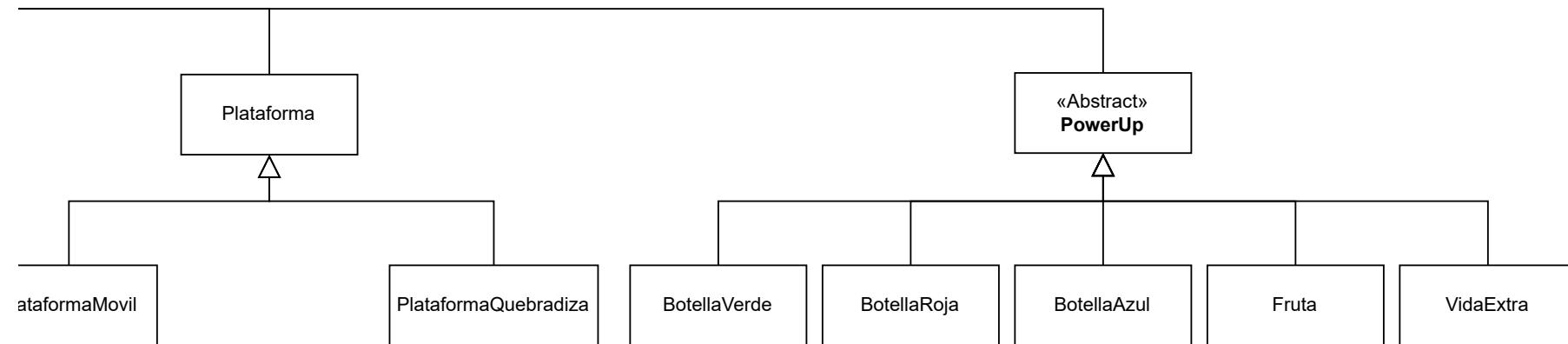


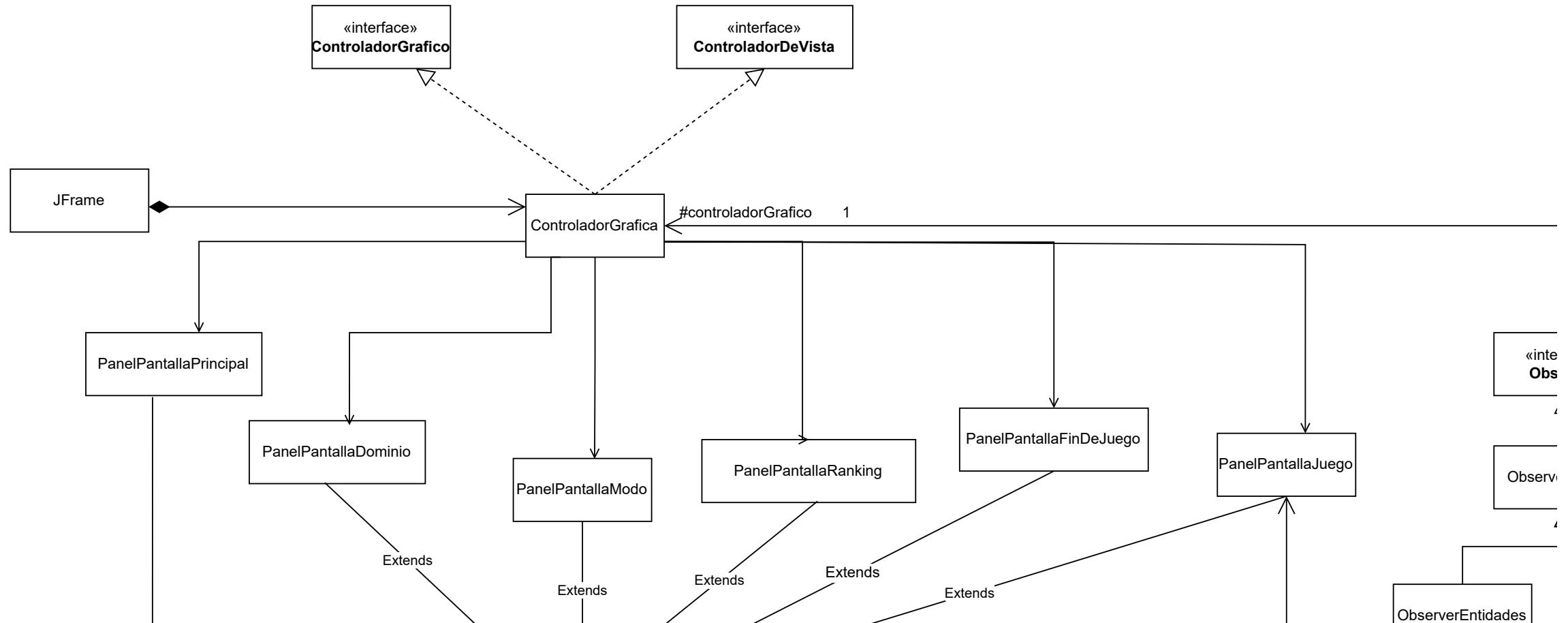
Elementos y símbolos en los diagramas de clases UML

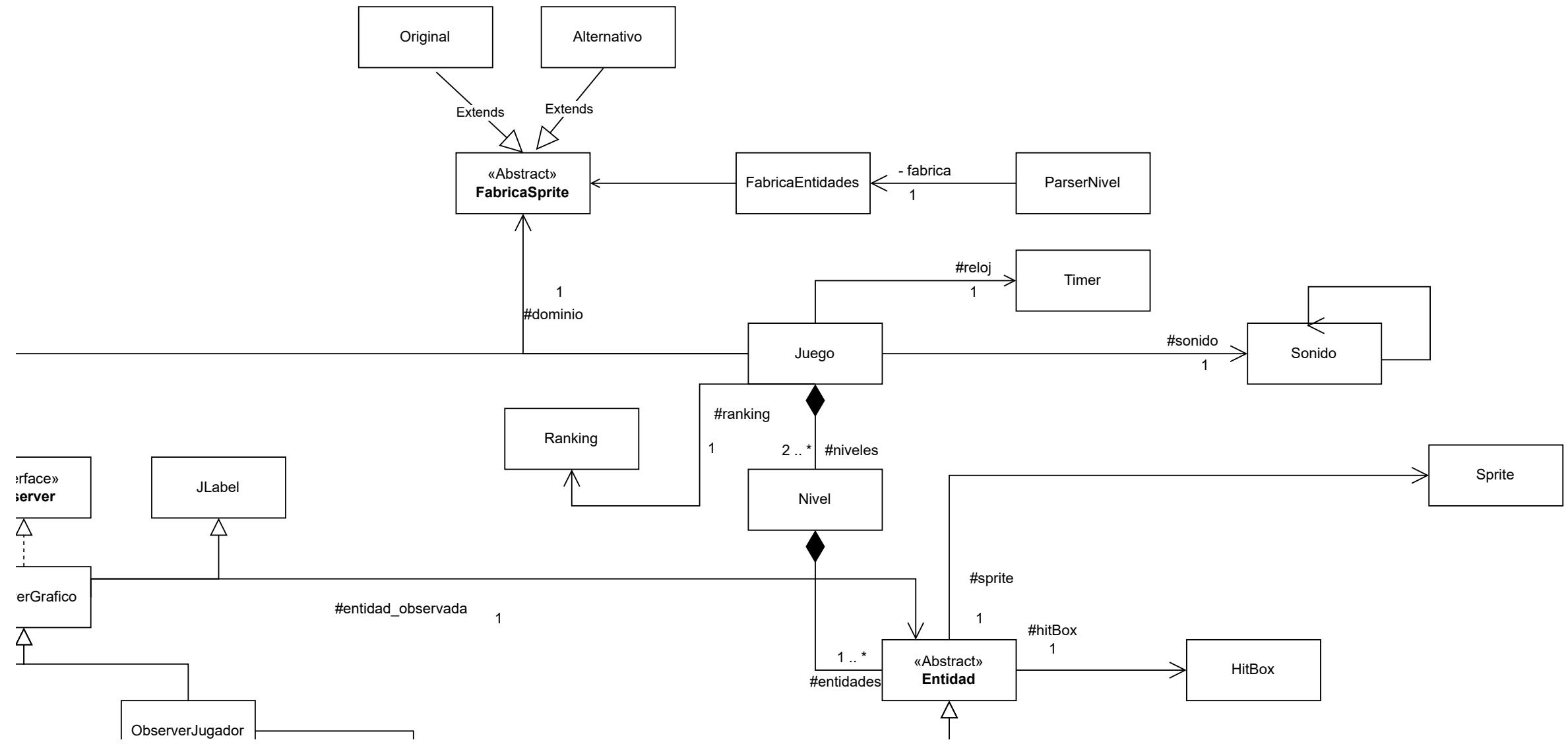


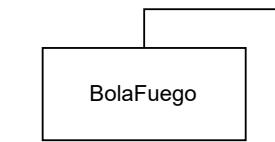
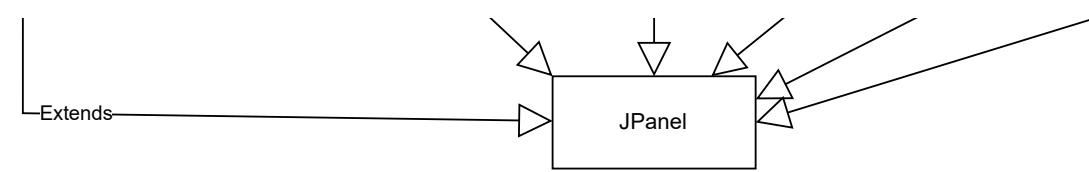


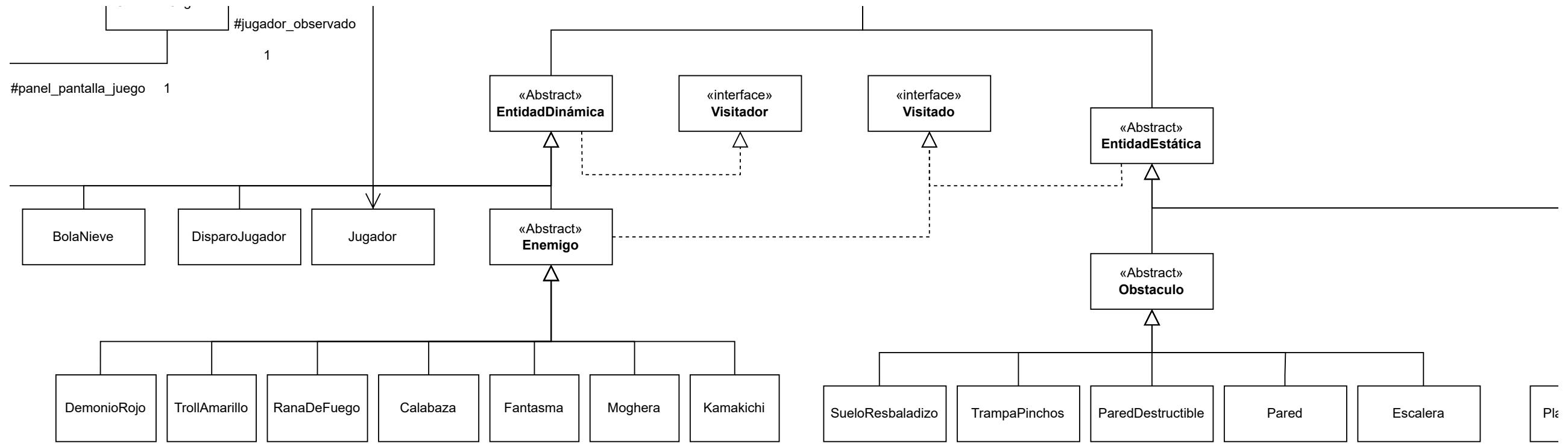


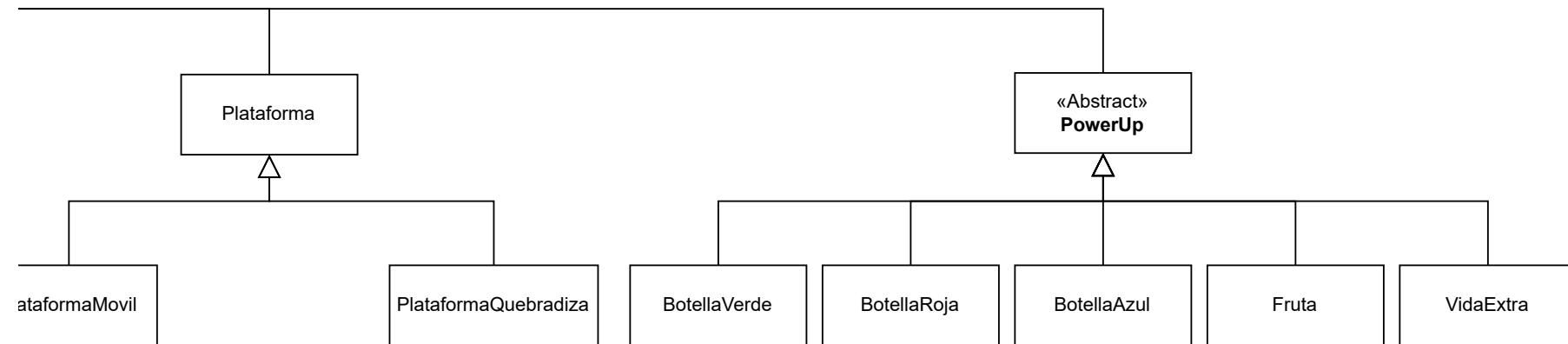


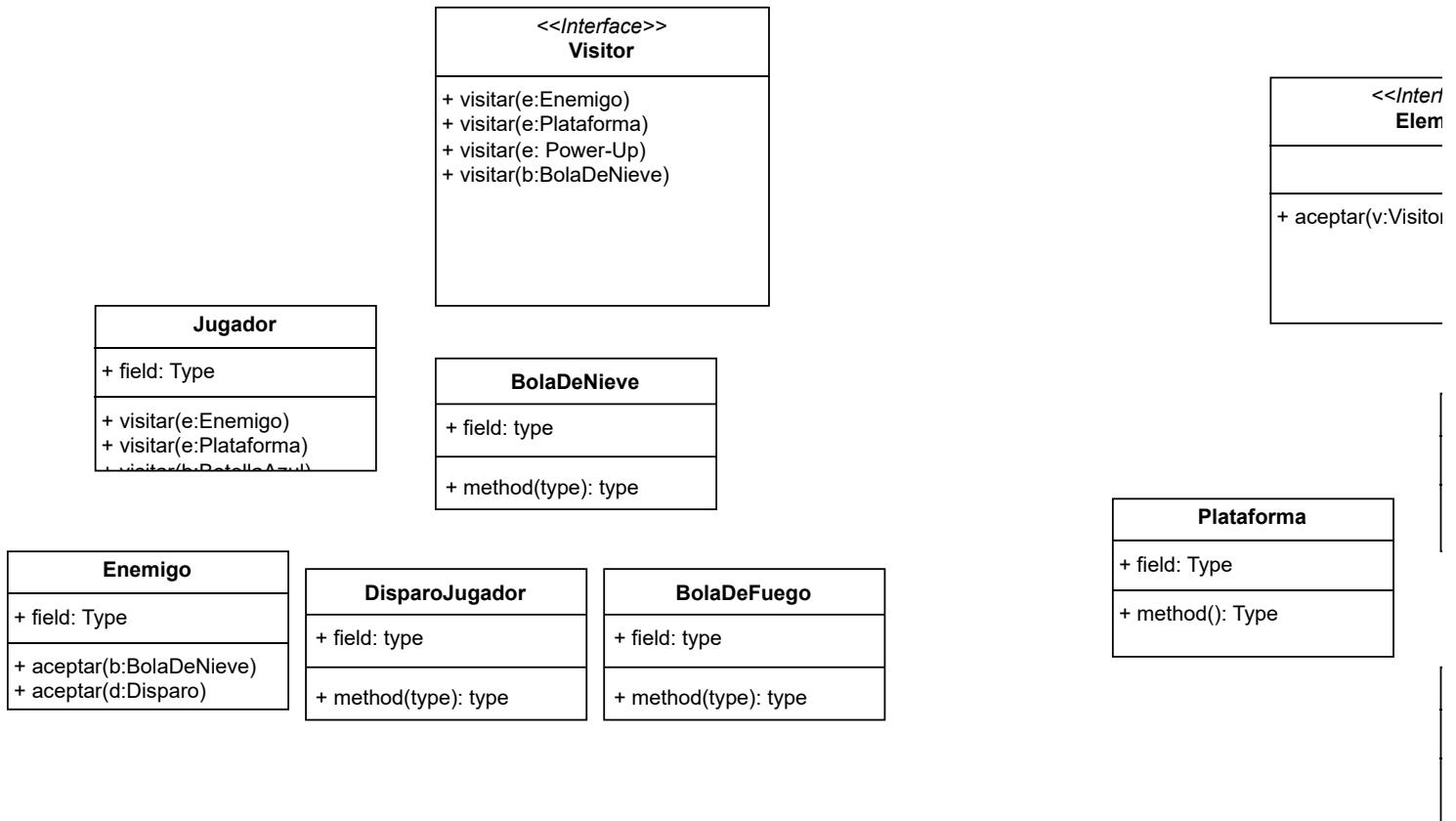










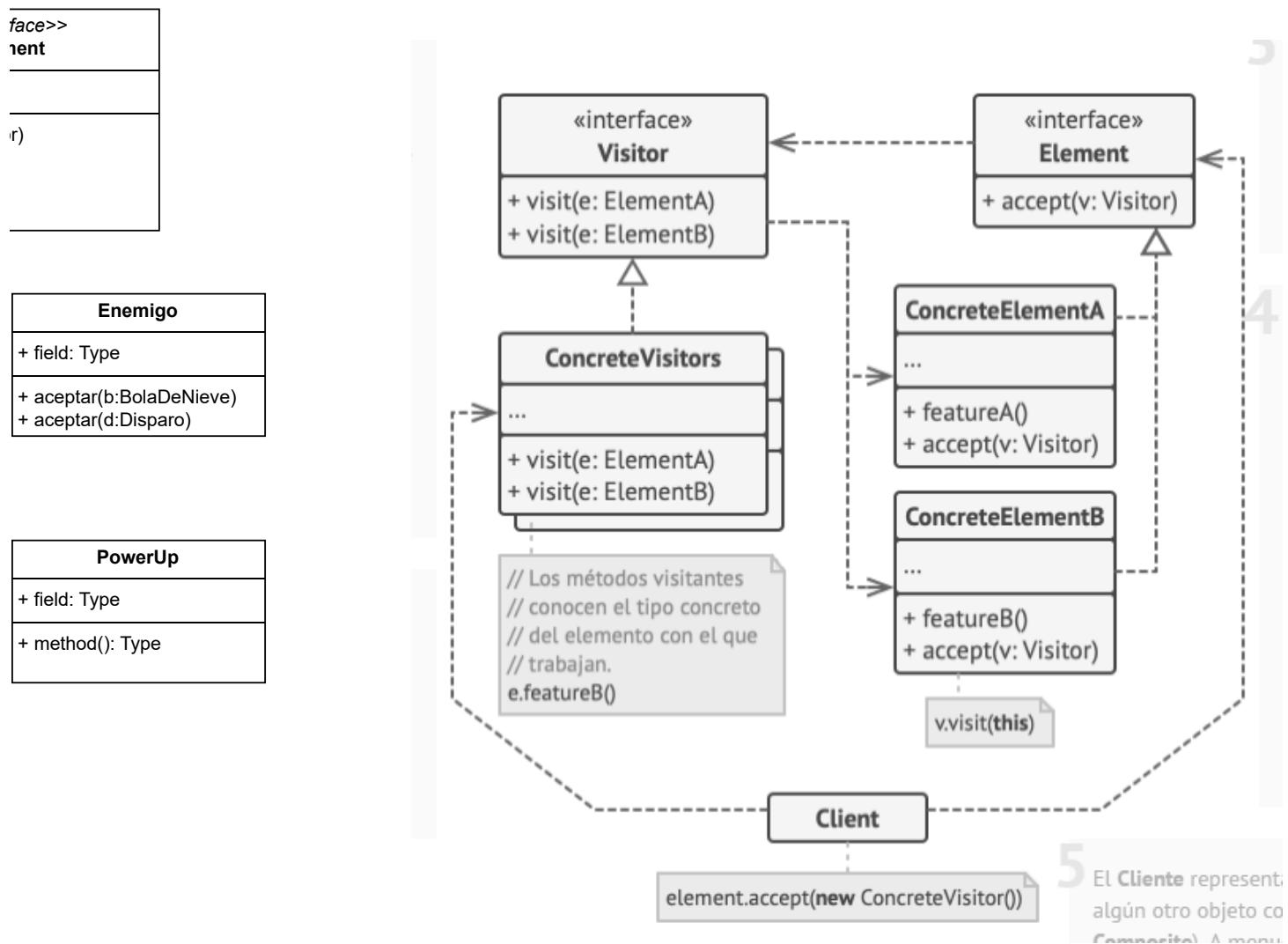


```
class Plataforma extends Entidad {
```

```
void aceptar(Visitor v) {
    // Lógica propia de la Plataforma
    this.reproducirSonidoColision();
```

```
// Delego en el visitante
```

```
v.visitarPlataforma(this);
```



```
class ExportVisitor implements Visitor is
    method doForCity(City c) { ... }
    method doForIndustry(Industry f) { ... }
    method doForSightSeeing(SightSeeing ss) { ... }
    // ...
}
```

```
// Código cliente
foreach (Node node in graph)
    node.accept(exportVisitor)

// Ciudad
class City is
    method accept(Visitor v) is
        v.doForCity(this)
    // ...

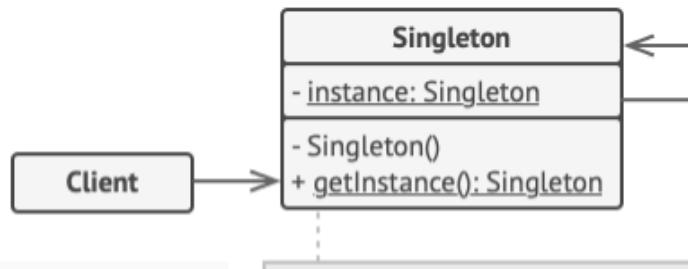
// Industria
class Industry is
    method accept(Visitor v) is
```

```
// Lógica adicional si quiero  
this.marcarComoUsada();  
}  
}
```

```
--> v.doForIndustry(this)  
// ...
```

Sonido
- instancia:Sonido
- mapeo: Map <String,Clip>
-Sonido() //Builder privado
+ getInstancia(): Sonido (static)
+ reproducir(s: String): void
- cargarSonido(r:String) :Clip
+-----+ Musica +-----+
- stream : SourceDataLine
- archivo : String
+ reproducir(loop:boolean) : void
+ detener() : void
+-----+

Sonido
+-----+ Sonido <>Singleton>< +-----+
- instance : Sonido [static]
- sonidos : Map<String, Clip> // efectos precargados
- musicaActual : Musica // pista en curso
+ getInstance() : Sonido [static]
+ reproducirSonido(ruta:String) : void
+ reproducirMusica(ruta:String, loop:boolean) : void
+ detenerMusica() : void
- cargarSonido(ruta:String) : Clip
+-----+



1 La clase **Singleton** declara el método estático `obtenerInstancia` que devuelve la misma instancia de su propia clase.

El constructor del Singleton debe ocultarse del código cliente. La llamada al método `obtenerInstancia` debe ser la única manera de obtener el objeto de Singleton.

```

if (instance == null) {
    // Nota: si estás creando una aplicación
    // que soporte el multihilo, debes
    // colocar un bloqueo de hilo aquí.
    instance = new Singleton()
}
return instance

```