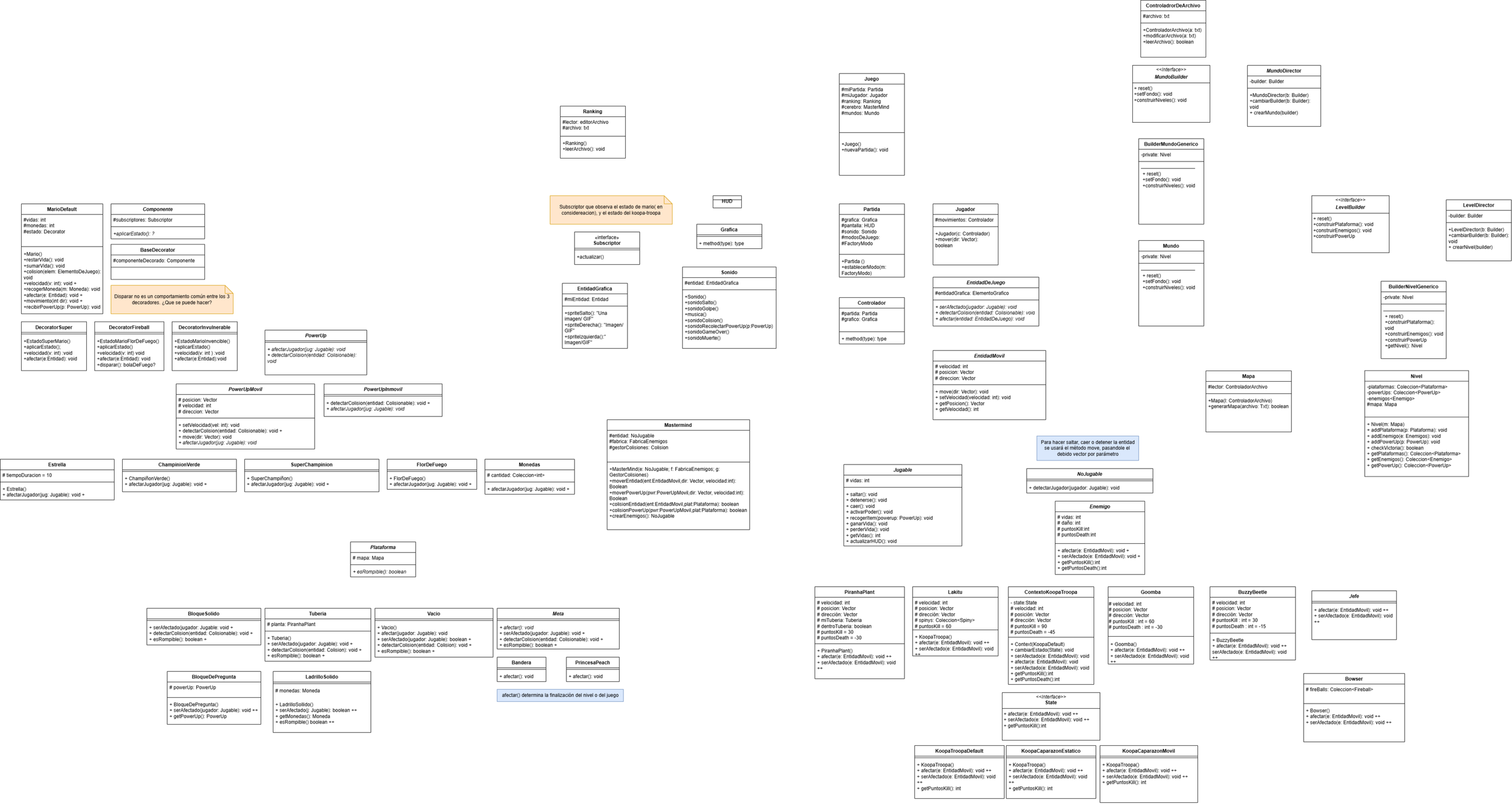
ControladrorDeArchivo
#archivo: txt
+ControladorArchivo(a: txt)
+modificarArchivo(a: txt)
+leerArchivo(): boolean
Juego
#miPartida: Partida
#miJugador: Jugador
#ranking: Ranking
#cerebro: MasterMind
#mundos: Mundo
+Juego()
+nuevaPartida(): void
<<Interface>>
MundoBuilder
+ reset()
+setFondo(): void
+construirNiveles(): void
MundoDirector
-builder: Builder
+MundoDirector(b: Builder)
+cambiarBuilder(b: Builder): void
+ crearMundo(builder)
Ranking
#lector: editorArchivo
#archivo: txt
+Ranking()
+leerArchivo(): void
BuilderMundoGenerico
-private: Nivel
+ reset()
+setFondo(): void
+construirNiveles(): void
MarioDefault
#vidas: int
#monedas: int
#estado: Decorator
+Mario()
+restarVida(): void
+sumarVida(): void
+colision(elem: ElementoDeJuego): void
+velocidad(v: int): void +
+recogerMoneda(m: Moneda): void
+afectar(e: Entidad): void +
+movimiento(int dir): void +
+recibirPowerUp(p: PowerUp): void
Componente
#subscriptores: Subscriptor
+aplicarEstado(): ?
BaseDecorator
#componenteDecorado: Componente
Disparar no es un comportamiento común entre los 3 decoradores. ¿Que se puede hacer?
Subscriptor que observa el estado de mario(en considereacion), y el estado del koopa-troopa
«interface»
Subscriptor
+actualizar()
HUD
Grafica
+ method(type): type
Partida
#grafica: Grafica
#pantalla: HUD
#sonido: Sonido
#modosDeJuego:
#FactoryModo
+Partida ()
+establecerModo(m: FactoryModo)
Jugador
#movimientos: Controlador
+Jugador(c: Controlador)
+mover(dir: Vector): boolean
Mundo
-private: Nivel
+ reset()
+setFondo(): void
+construirNiveles(): void
<<Interface>>
LevelBuilder
+ reset()
+construirPlataforma(): void
+construirEnemigos(): void
+construirPowerUp
LevelDirector
-builder: Builder
+LevelDirector(b: Builder)
+cambiarBuilder(b: Builder): void
+ crearNivel(builder)
Sonido
#entidad: EntidadGrafica
+Sonido()
+sonidoSalto()
+sonidoGolpe()
+musica()
+sonidoColision()
+sonidoRecolectarPowerUp(p:PowerUp)
+sonidoGameOver()
+sonidoMuerte()
EntidadGrafica
#miEntidad: Entidad
+spriteSalto(): "Una imagen/ GIF"
+spriteDerecha(): "Imagen/ GIF"
+spriteIzquierda():" Imagen/GIF"
Controlador
#partida: Partida
#grafico: Grafica
+ method(type): type
EntidadDeJuego
#entidadGrafica: ElementoGrafico
+ serAfectado(jugador: Jugable): void
+ detectarColision(entidad: Colisionable): void
+ afectar(entidad: EntidadDeJuego): void
BuilderNivelGenerico
-private: Nivel
+ reset()
+construirPlataforma(): void
+construirEnemigos(): void
+construirPowerUp
+getNivel(): Nivel
DecoratorSuper
+EstadoSuperMario()
+aplicarEstado();
+velocidad(v: int): void
+afectar(e:Entidad): void
DecoratorFireball
+EstadoMarioFlorDeFuego()
+aplicarEstado()
+velocidad(v: int) void
+afectar(e:Entidad): void
+disparar(): bolaDeFuego?
DecoratorInvulnerable
+EstadoMarioInvencible()
+aplicarEstado()
+velocidad(v: int): void
+afectar(e:Entidad):void
PowerUp
+ afectarJugador(jug: Jugable): void
+ detectarColision(entidad: Colisionable): void
PowerUpInmovil
+ detectarColision(entidad: Colisionable): void +
+ afectarJugador(jug: Jugable): void
PowerUpMovil
posicion: Vector
velocidad: int
direccion: Vector
+ setVelocidad(vel: int): void
+ detectarColision(entidad: Colisionable): void +
+ move(dir: Vector): void
+ afectarJugador(jug: Jugable): void
EntidadMovil
velocidad: int
posicion: Vector
direccion: Vector
+ move(dir: Vector): void
+ setVelocidad(velocidad: int): void
+ getPosicion(): Vector
+ getVelocidad(): int
Mapa
#lector: ControladorArchivo
+Mapa(l: ControladorArchivo)
+generarMapa(archivo: Txt): boolean
Nivel
-plataformas: Coleccion<Plataforma>
-powerUps: Coleccion<PowerUp>
-enemigos<Enemigo>
#mapa: Mapa
+ Nivel(m: Mapa)
+ addPlataforma(p: Plataforma): void
+ addEnemigo(e: Enemigos): void
+ addPowerUp(p: PowerUp): void
+ checkVictoria(): boolean
+ getPlataformas(): Coleccion<Plataforma>
+ getEnemigos(): Coleccion<Enemigo>
+ getPowerUp(): Coleccion<PowerUp>
Mastermind
#entidad: NoJugable
#fabrica: FabricaEnemigos
#gestorColisiones: Colision
+MasterMind(e: NoJugable, f: FabricaEnemigos, g: GestorColisiones)
+moverEntidad(ent:EntidadMovil,dir: Vector, velocidad:int): Boolean
+moverPowerUp(pwr:PowerUpMovil,dir: Vector, velocidad:int): Boolean
+colisionEntidad(ent:EntidadMovil,plat:Plataforma): boolean
+colisionPowerUp(pwr:PowerUpMovil,plat:Plataforma): boolean
+crearEnemigos(): NoJugable
Para hacer saltar, caer o detener la entidad se usará el método move, pasandole el debido vector por parámetro
Jugable
vidas: int
+ saltar(): void
+ detenerse(): void
+ caer(): void
+ activarPoder(): void
+ recogerItem(powerup: PowerUp): void
+ ganarVida(): void
+ perderVida(): void
+ getVidas(): int
+ actualizarHUD(): void
NoJugable
+ detectarJugador(jugador: Jugable): void
Enemigo
vidas: int
daño: int
puntosKill:int
puntosDeath:int
+ afectar(e: EntidadMovil): void +
+ serAfectado(e: EntidadMovil): void +
+ getPuntosKill():int
+ getPuntosDeath():int
Estrella
tiempoDuracion = 10
+ Estrella()
+ afectarJugador(jug: Jugable): void +
ChampinionVerde
+ ChampiñonVerde()
+ afectarJugador(jug: Jugable): void +
SuperChampinion
+ SuperChampiñon()
+ afectarJugador(jug: Jugable): void +
FlorDeFuego
+ FlorDeFuego()
+ afectarJugador(jug: Jugable): void +
Monedas
cantidad: Coleccion<int>
+ afectarJugador(jug: Jugable): void +
Plataforma
mapa: Mapa
+ esRompible(): boolean
PiranhaPlant
velocidad: int
posicion: Vector
dirección: Vector
miTuberia: Tuberia
dentroTuberia: boolean
puntosKill = 30
puntosDeath = -30
+ PiranhaPlant()
+ afectar(e: EntidadMovil): void ++
+ serAfectado(e: EntidadMovil): void ++
Lakitu
velocidad: int
posicion: Vector
dirección: Vector
spinys: Coleccion<Spiny>
puntosKill = 60
+ KoopaTroopa()
+ afectar(e: EntidadMovil): void ++
+ serAfectado(e: EntidadMovil): void ++
ContextoKoopaTroopa
- state:State
velocidad: int
posicion: Vector
dirección: Vector
puntosKill = 90
puntosDeath = -45
+ Context(KoopaDefault)
+ cambiarEstado(State): void
+ serAfectado(e: EntidadMovil): void
+ afectar(e: EntidadMovil): void
+ serAfectado(e: EntidadMovil): void
+ getPuntosKill():int
+ getPuntosDeath():int
Goomba
velocidad: int
posicion: Vector
dirección: Vector
puntosKill : int = 60
puntosDeath : int = -30
+ Goomba()
+ afectar(e: EntidadMovil): void ++
+ serAfectado(e: EntidadMovil): void ++
BuzzyBeetle
velocidad: int
posicion: Vector
dirección: Vector
puntosKill : int = 30
puntosDeath : int = -15
+ BuzzyBeetle
+ afectar(e: EntidadMovil): void ++
serAfectado(e: EntidadMovil): void ++
Jefe
+ afectar(e: EntidadMovil): void ++
+ serAfectado(e: EntidadMovil): void ++
Bowser
fireBalls: Coleccion<Fireball>
+ Bowser()
+ afectar(e: EntidadMovil): void ++
+ serAfectado(e: EntidadMovil): void ++
<<Interface>>
State
+ afectar(e: EntidadMovil): void ++
+ serAfectado(e: EntidadMovil): void ++
+ getPuntosKill():int
KoopaTroopaDefault
+ KoopaTroopa()
+ afectar(e: EntidadMovil): void ++
+ serAfectado(e: EntidadMovil): void ++
+ getPuntosKill(): int
KoopaCaparazonEstatico
+ KoopaTroopa()
+ afectar(e: EntidadMovil): void ++
+ serAfectado(e: EntidadMovil): void ++
+ getPuntosKill(): int
KoopaCaparazonMovil
+ KoopaTroopa()
+ afectar(e: EntidadMovil): void ++
+ serAfectado(e: EntidadMovil): void ++
+ getPuntosKill(): int
BloqueSolido
+ serAfectado(jugador: Jugable: void +
+ detectarColision(entidad: Colisionable): void +
+ esRompible(): boolean +
Tuberia
planta: PiranhaPlant
+ Tuberia()
+ serAfectado(jugador: Jugable): void +
+ detectarColision(entidad: Colision): void +
+ esRompible(): boolean +
Vacio
+ Vacio()
+ afectar(jugador: Jugable): void
+ serAfectado(jugador: Jugable): boolean +
+ detectarColision(entidad: Colision): void +
+ esRompible(): boolean +
Meta
+ afectar(): void
+ serAfectado(jugador: Jugable): void +
+ detectarColision(entidad: Colisionable): void +
+ esRompible(): boolean +
Bandera
+ afectar(): void
PrincesaPeach
+ afectar(): void
afectar() determina la finalización del nivel o del juego
BloqueDePregunta
powerUp: PowerUp
+ BloqueDePregunta()
+ serAfectado(jugador: Jugable): void ++
+ getPowerUp(): PowerUp
LadrilloSolido
monedas: Moneda
+ LadrilloSollido()
+ serAfectado(j: Jugable): boolean ++
+ getMonedas(): Moneda
+ esRompible() boolean ++