**FIUBA - 75.07**

**Algoritmos y programación III**

*Trabajo práctico 2: Dragon AlgoBall*

1er cuatrimestre, 2017

(trabajo grupal de 4 integrantes)

Alumnos:

|  |  |  |
| --- | --- | --- |
| **Nombre** | **Padrón** | **Mail** |
| Santos, Ramiro | 99772 | ram.santos@hotmail.com |
| Cordeu, Thomas | 99288 | tcordeu@gmail.com |
| Kakazu, Mariano | 98178 | kakazu.mariano@gmail.com |
| Aparicio, Rodrigo | 98967 | Rodriver999@hotmail.com |

***Fecha de entrega final***: 22 de junio

***Tutor***: Carlos Fontela

***Comentarios***:

**Supuestos**

**Modelo de dominio**

A la hora de encarar el Trabajo Práctico empezamos a identificar las diferentes identidades presentes en el problema. Por un lado, está el Juego, con sus respectivos Jugadores, cada uno con un Equipo (que a la vez estos tienen 3 Personajes cada uno). A su vez hay un Tablero, donde se dispondrán no solo los Personajes, sino también los Consumibles. Este Tablero puede ser organizado con un Organizador del Juego, a quien Juego le delega varias responsabilidades, como ubicar los Personajes de cada Equipo en el campo de batalla.  
En cuanto a los modos de cada Personaje, estos tienen cada uno los 3 Estados correspondientes aunque obviamente solo uno a la vez pueden utilizar como indica el reglamento del juego.  
Por último, para modelar los enfrentamientos entre los Personajes de los diferentes Equipos se usa un Desarrollador de Ataque o Batalla, al cual se le delegan las responsabilidades de lo que sucede en una batalla, por ejemplo, como se ven modificados los Personajes en ella.

**Diagramas de clases**

**Diagramas de secuencia**

**Diagrama de paquetes**

**Diagramas de estado**

**Detalles de implementación**

**Excepciones**

Hasta ahora se han hecho seis excepciones. La primera llamada ErrorConsumibleInstantaneo que salta cuando se le quiere reducir la cantidad de usos restantes a un consumible que es instantáneo y solo se puede usar una vez; la segunda llamada ErrorNoHayUsosRestantes que avisa que se quiso reducir la cantidad de usos de un consumible que ya se usó todas las veces posibles, es decir que ya pasaron todos los turnos en los que funcionaba. La excepción ErrorCasilleroYaOcupado avisa que no se puede meter más nada en el casillero porque está ocupado, y al revés de esta, la excepción ErrorNoHayObjeto avisa que se quiere obtener un objeto (ya sea un personaje o un consumible) de un casillero que no tiene. Por último se implementó la excepción ErrorMovimientoInvalido que justamente salta cuando se quiso realizar un momivimiento erróneo.

# 