**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**

Se implementó el método cambiarAEstado1() (estado original de un personaje) pero sin embargo se omitió esta opción a la hora de realizar la vista, ya que no conlleva ninguna ventaja (sí se permitó transformar a un personaje tanto al Estado 2 o el 3 cuantas veces quiera, siempre que pueda obviamente), esto quiere decir que como primer supuesto se puede mencionar el hecho de que no se permite transformar un personaje a su estado original.

Como segundo supuesto se tomó que el efecto de la Semilla del Ermitaño es instantáneo, es decir que al ser agarrada por un personaje el efecto se ejecuta en el momento, impidiendo que el jugador la aplique en su personaje en el momento que quiera.

Otro supuesto considerado fue que al atacar un personaje con una Esfera del Dragón encima, esta aumenta su ataque en un 25% pero este aumento no se lo tiene en cuenta a la hora de comprar el daño del poder de pelea del personaje atacante con el personaje atacado. Por lo tanto, el que la esfera aumente el poder de ataque no significa que en comparación el que ataca tenga mayor poder de pelea.

También no se consideró que MajinBoo al inhabilitar un personaje con su ataque Conviértete en Chocolate le impide transformarse, de manera que un personaje con 3 turnos inhabilitados no significa que el personaje no puede transformarse en alguno de esos 3 turnos. En cuanto a esto, también se consideró que los 3 turnos de inhabilitación también incluyen los turnos del oponente, es decir, de quien efectuó el ataque Conviértete en Chocolate.

En cuanto a la transformación de Gohan en SuperSayajin fase 2 se consideró que si los aliados de Gohan están muertos, por más que sus vidas sean menos del 30% del total (son 0), este no puede transformarse.

Por último, algo quizás obvio pero no está demás aclararlo, se asume que las Esferas del Dragón se acumulan en cada equipo más allá de que hayan perdido su efecto de ataque.

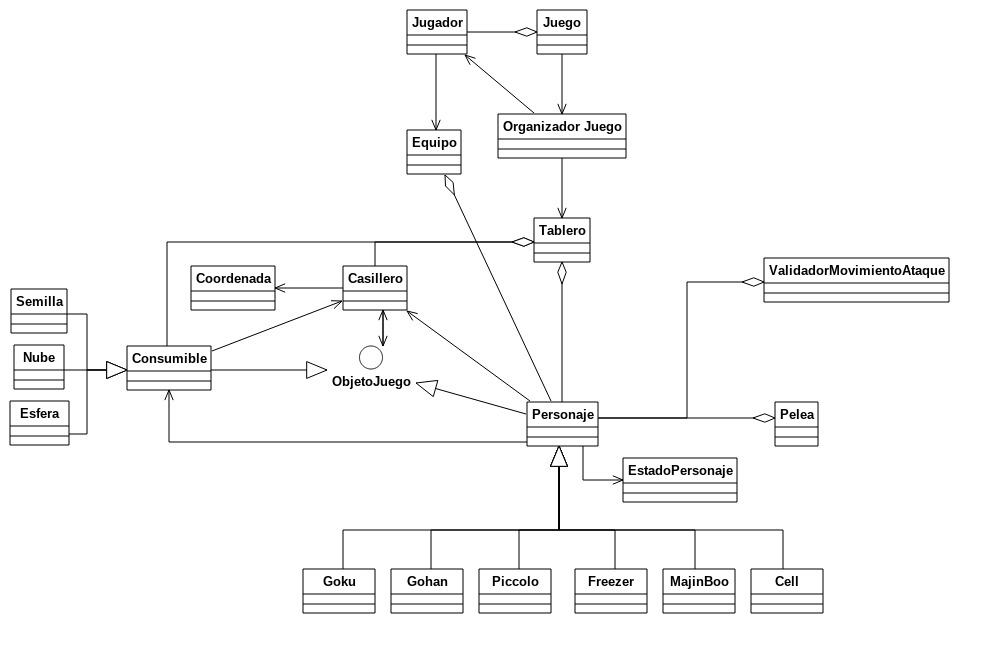
**Modelo de dominio**

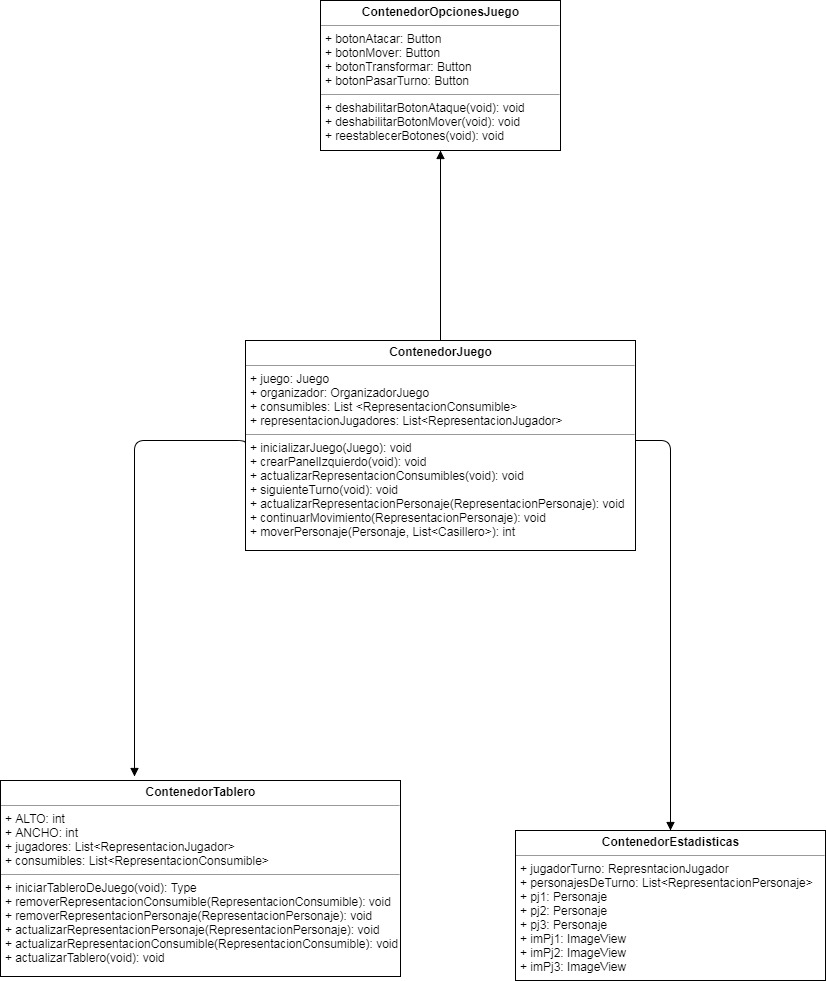
A la hora de encarar el Trabajo Práctico se empezó a identificar las diferentes identidades presentes en el problema. Por un lado, está el Juego, con sus respectivos Jugadores, cada uno con un Equipo (que a la vez estos tienen 3 Personajes cada uno). A su vez hay un Tablero, donde se dispondrán no solo 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 se puede utilizar como indica el reglamento del juego.

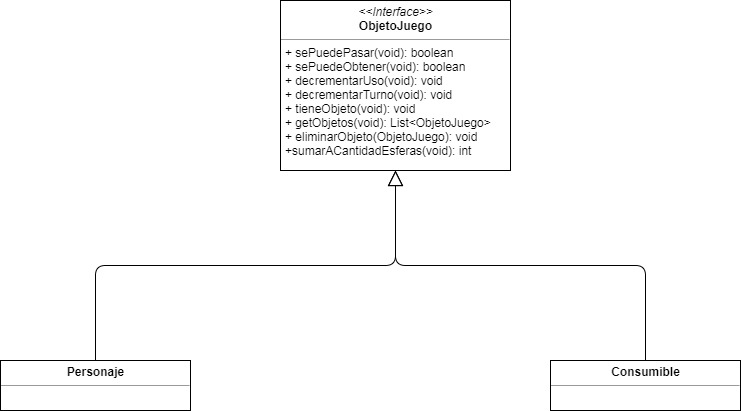
Por último, para modelar los enfrentamientos entre los Personajes de los diferentes Equipos se usa una clase Pelea, a la cual se le delegan las responsabilidades de lo que sucede en una batalla, por ejemplo, como se ven modificados los Personajes en ella.

**Diagramas de clases**

*****Diagrama I: Diagrama de clases del modelo en su totalidad.*

**

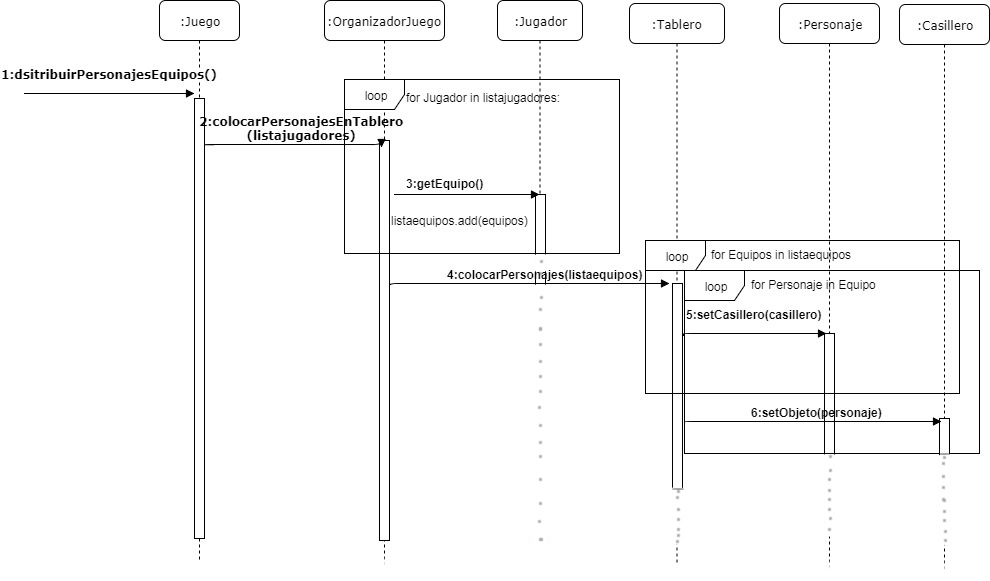
*Diagrama II: Diagrama de los contenedores principales de la vista (aparecen siempre durante el desarrollo de una partida). Para mayor claridad se omitieron algunos atributos poco relevantes y métodos como getters y setters.*

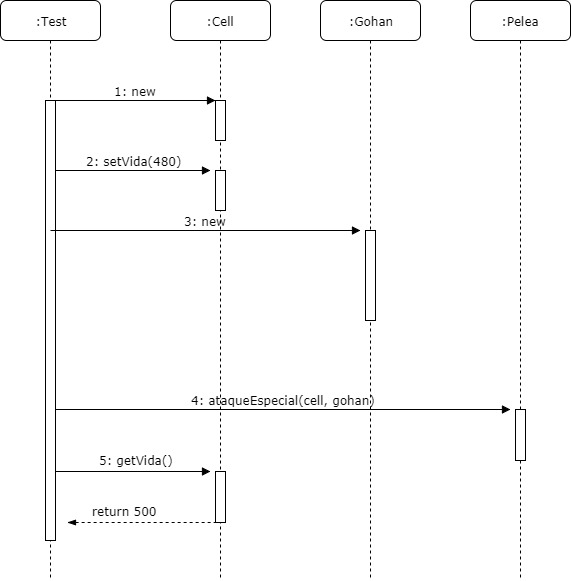
****

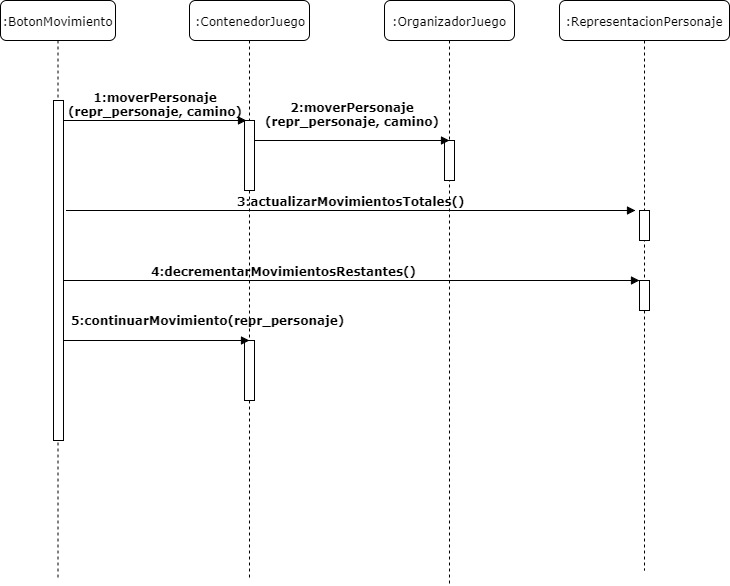
*Diagrama III: diagrama de la clase ObjetoJuego. Al igual que en el anterior diagrama se evitaron escribir algunos métodos para mayor claridad.*

Si bien este último diagrama es más simple que los anteriores pareció adecuado agregarlo teniendo en cuenta que es la única clase de todo el proyecto que es una interfaz, y siendo la interfaz un concepto importante visto en clase.

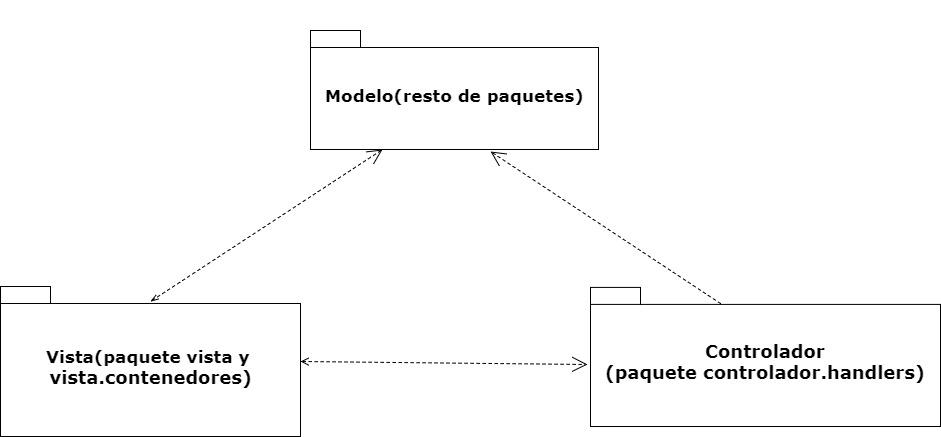
**Diagramas de secuencia**

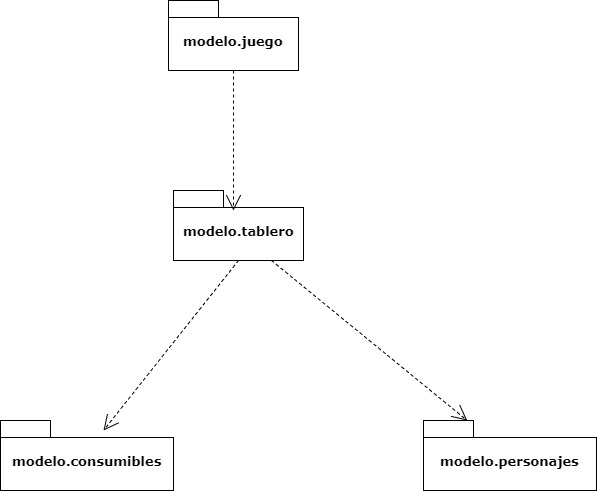
*Diagrama IV: diagrama de secuencia que muestra la comunicación entre clases a la hora de distribuir los personajes en el tablero al inicio del juego.*

*****Diagrama V: diagrama de secuencia que muestra el caso paritcular en que Cell aplica su ataque especial (Absrober) sobre Gohan.*

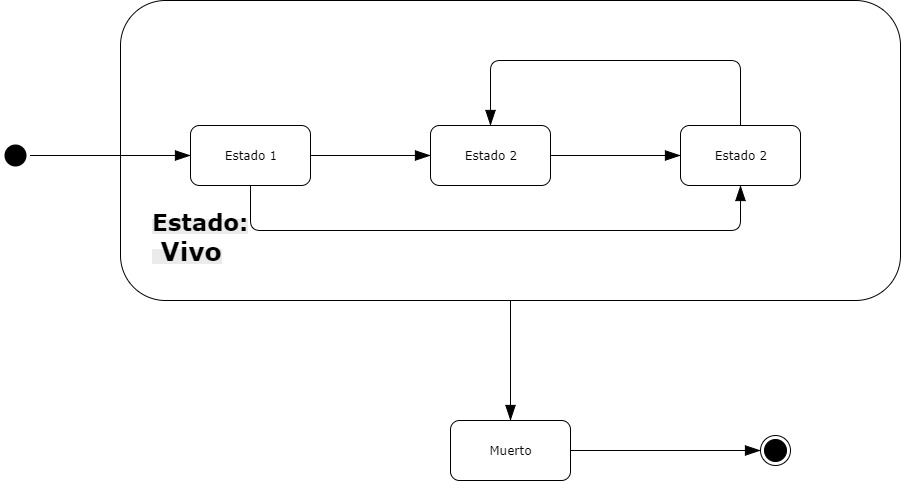
*****Diagrama VI: diagrama de secuencia que muestra la interacción entre clases de la vista al presionar el botón para mover un personaje en especial (también se ve como la vista interactúa con el modelo).*

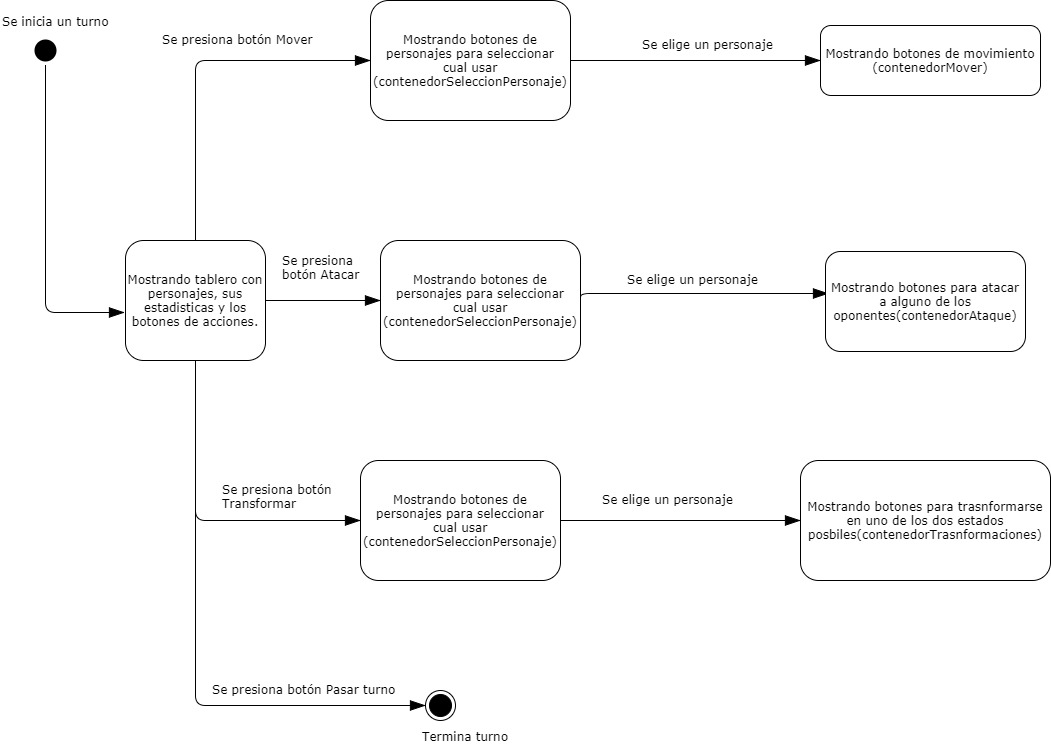
**Diagrama de paquetes**

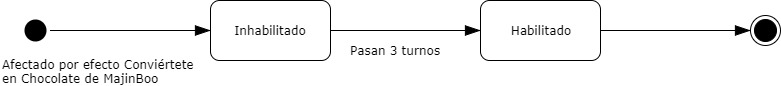
*****Diagrama VII: diagrama de paquetes que muestra la interacción entre los paquetes del modelo, los de la vista y el del controlador.*

*****Diagrama VIII: diagrama de paquetes que muestra la interacción entre los paquetes principales del modelo.*

**Diagramas de estado**

*****Diagrama IX: diagrama de estado que muestra todos los estados (a grandes rasgos) por los que puede pasar un Personaje.*

*****Diagrama X: diagrama de estado que muestra los estados por los que pasa la vista principal (donde se desarrolla la partida) al seleccionar los botones que ofrece durante un turno.*

***Diagrama XI: diagrama de estado que muestra los estados por los que pasa un personaje al ser afectado por el ataque Conviértete en Chocolate de MajinBoo.*

Si bien este último diagrama es bastante simple, pareció adecuado agregarlo, ya que aun así aporta información.

**Detalles de implementación**

Durante el desarrollo del modelo primero se pensó en implementar los estados de los personajes por separado, es decir, una clase por cada estado posible del personaje. Luego, al notar que esta era una manera ineficiente de resolver el problema se utilizó el patrón de diseño Double Dispatch(DD), evitando tener a los estados en clases separadas, los cuales no tenían comportamiento diferente más que algunos cambios de valores de los atributos de la clase. Con DD lo que se hizo fue que la clase EstadoPersonaje (clase que define algunos atributos sobre el estado particular del personaje) tenga el comportamiento de todos los estados posibles de todos los personajes. Dicho de otra manera, las 18 clases de estados, pasaron a ser 18 métodos de la clase EstadoPersonaje. De esta forma, cuando se quiere cambiar el estado del personaje, directamente la instancia del estado en el personaje cambia al estado deseado.

Por otro lado, al notar que tanto los Consumibles como los Personajes compartían varios comportamientos se decidió hacer una interfaz ObjetoJuego de modo que los englobase y permitiese ahorrar código y mejorar la legibilidad del mismo.

Otra cosa mencionable es la clase Pelea, la cual se creó para encargarse de todo lo que tenga que ver con lo que pasa durante una batalla (cambios de estatus en cada personaje según el ataque y el consumible que tiene el atacante).

Para hacer la vista se separó lo que eran los contenedores donde aparecían tanto el tablero como los botones en cada escena de las representaciones (clases encargadas de tener y actualizar las imágenes de cada personaje y cada consumible).

Si bien se intentó utilizar el patrón MVC, no se pudo cumplir con todos los requisitos ya que la vista terminó interactuando con el modelo.

**Excepciones**

* Por un lado están las excepciones que no aparecen durante el desarrollo del juego (no se muestran al usuario):

-ErrorNoTieneCantidadUsos: esta excepción aparece cuando se quiere trabajar con los usos de una Nube o una Semilla del Ermitaño, ya que estas se las cuenta por turnos y no por usos.

-ErrorHayTurnosDeDuracion: misma excepción que la anterior pero salta cuando se quiere trabajar con turnos con la Esfera.

-ErrorNoHayTurnosDisponibles: salta cuando a un consumible como la Semilla del Ermitaño o la Nube se le quiere decrementar turnos estando ya en 0 su cantidad de turnos disponibles.

-ErrorNoHayUsosRestantes: lo mismo que la anterior pero para la esfera y con cantidad de usos.

Cabe aclarar que al que se le pueda llamar a la Esfera con métodos que tienen que ver con turnos y no con usos se debe a que se usó polimorfismo para trabajar con todos los consumibles por igual y no tener que preguntar cada vez qué tipo de consumible es y en base a eso ejecutar un método en especial.

-ErrorNoCompatibilidad: esta excepción salta cada vez que se utiliza un método de la interfaz ObjetoJuego con un Personaje cuando el método es solo aplicable para un Consumible (que es el otro ObjetoJuego).

-ErrorNoHayObjeto: salta cuando se le quiere pedir a un casillero un objeto y este no tiene.

-ErrorNoHayMasExtremos: excepción que avisa, cuando se está buscando los casilleros extremos del tablero, que no hay más extremos.

* Por otro lado están las excepciones que sí aparecen durante una partida y se muestran por pantalla los errores que tiran:

-ErrorNoHayKi: esta excepción salta cuando se quiere realizar un ataque especial o una transformación pero no se tiene el suficiente ki para ejectuarlo.

-ErrorNoSePuedeRealizarAtaqueEspecial:

-ErrorNoPuedeCambiarEstado: esta excepción salta cuando el personaje no puede transformarse, pero no debido a la falta de ki, sino a otras condiciones, como es el caso de Gohan, que no puede transformarse si sus aliados no tienen menos del 30% de su vida.

-ErrorAtaqueInvalido: salta cuando un personaje quiere atacar a otro que supera su alcance de ataque en cuanto a distancia.

-ErrorCasilleroYaOcupado: esta excepción aparece cuando se quiere ir por un camino el cual está bloqueado por otro personaje, osea, cuando se intenta avanzar a un casillero ocupado.

-ErrorPersonajeInutilizado: aparece cuando se quiere accionar con un personaje y este se encuentra inhabilitado por el ataque Conviertete en Chocolate de MajinBoo. Esta excepción no incluye transformarse (como se indico en la sección supuestos el estar inhábil no incluye no poder transformarse).

-ErrorMovimientoInvalido: salta cuando se quiere ir más allá de los límites del tablero con un personaje.