FACULTAD DE INGENIERIA
Universidad de Buenos Aires

Trabajo Práctico 2

[7507/9502] Algoritmos y Programación III
Curso 1
Segundo cuatrimestre del 2019

102349	BAZANO SAMMARTINO, Marina	marinabazano@gmail.com
102358	GIAMPIERI, Leonardo	lgiampieri62@gmail.com
102573	FEDELE EDO, Cecilia	fedelececilia@gmail.com
102264	HOJMAN, Joaquín	hojmanjoaquin@gmail.com

Índice

1. Introducción

El presente informe reune la documentación de la solución del segundo trabajo práctico de la materia Algoritmos y Programación III que consiste en desarrollar un juego en Java utilizando los conceptos del paradigma de la orientación a objetos vistos en el curso.

Al ser un trabajo integrador de toda la materia utilizamos todo lo aprendido, en este trabajo pueden verse los conceptos de polimorfismo, herencia, encapsulamiento, Test-Drive Development, integración continua, Extreme Programming, etc.

2. Supuestos

Antes de comenzar el desarrollo de la aplicación creamos nuestro repositorio de github y la agrupación con Travis, además conectamos nuestros IDEs al repositorio. Una vez tuvimos nuestro control de versiones en funcionamiento empezamos con el planteo del trabajo.

Comenzamos leyendo el enunciado y analizando los sustantivos encontrados, como por ejemplo: Jugador, Tablero, Unidades, etc. Con estas posibles clases detalladas empezamos a plantear las relaciones entre sí, como se comunican y que contienen cada una. Por otro lado, dentro de Tablero, decidimos tener otra clase que detalle el comportamiento de cada casillero dentro del mismo, y a partir de eso ver si es necesaria la creación de clases para las ubicaciones o coordenadas.

Finalmente, para empezar a escribir el código, escribimos algunas pruebas del funcionamiento de cada clase que pensabamos agregar, para facilitar el armado y comprobación de funcionamiento de cada una.

Por el lado de la interfaz gráfica fue complejo ver la unión entre el modelo y su interfaz, por ello comenzamos con lo básico. Hicimos un diseño inicial en papel de cuales serían los layouts a desarrollar y como sería la movilidad y ataque de nuestras unidades.

Previo a esto tuvimos que instalar javaFX y comprender su sintaxis y librerías.

Más allá de lo antes mencionado, no tuvimos suposiciones que afectaran el normal desarrollo del juego esperado por la catedra. La claridad de la consigna nos permitió obtener una aplicación que cumple lo pedido sin grandes cambios. Sin embargo, algunas suposiciones para el flujo del programa fueron: En la Selección de unidades la selección se realiza por turnos, es decir primero un jugador, luego el otro, y así hasta que ambos gastan todos sus puntos disponibles. Si un jugador se queda sin puntos antes que el otro, debe esperar a que este último finalice antes de comenzar el juego.

Cabe mencionar que mientras un jugador elige unidades se bloquean los botones de unidades ajenas y la parte del tablero enemigo.

Luego, en la pantalla de juego podemos apreciar el tablero con todas las unidades colocadas. Para jugar, tan solo hace falta que el usuario de turno escoja la unidad que va a utilizar, luego puede apretar un casillero vacío para moverla (siempre que esta pueda moverse y este a una distancia de tan solo un casillero) o bien tocar a una unidad enemiga y atacarla (o curarla, dependiendo de la unidad). Ciertas acciones invalidas hacen que el jugador pierda su turno. Los jugadores tienen solo una acción por turno, ya sea mover, atacar o curar.

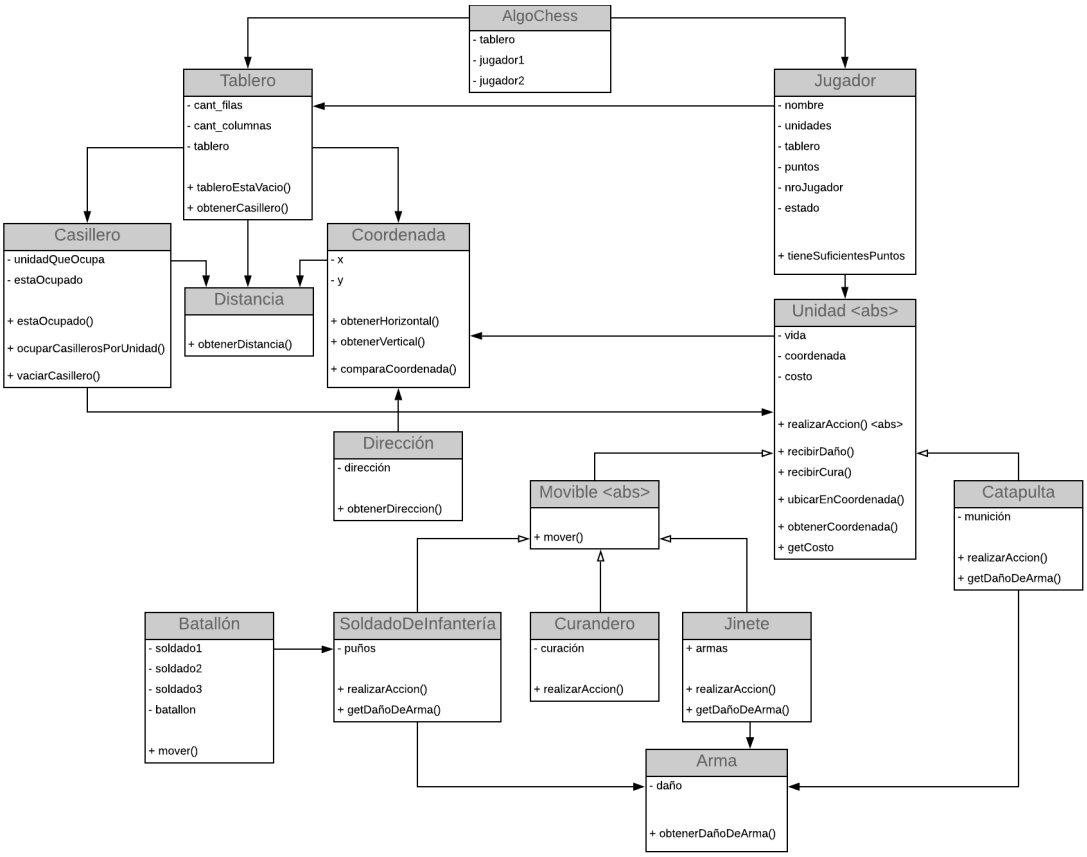
En el caso del batallón (tres soldados contiguos) para utilizarlo se debe apretar click derecho sobre cada uno de ellos, y luego hacer click derecho sobre el casillero a donde queremos mover el batallón. Recordamos que el batallón solo se mueve junto, no ataca.

Una vez eliminadas todas las unidades de uno de los jugadores pasamos a la pantalla final donde vemos el nombre del ganador y tenemos la opción de cerrar el juego o volver a jugar.

3. Diagramas de clase

En esta seccion podrá verse el diagrama de clases del programa AlgoChess.

En el mismo están consideradas las relaciones entre las clases del programa. Además, se muestran los métodos más importantes del programa.



4. Detalles de implementación

En esta sección del informe se especificara el comportamiento de las clases más importantes del modelo desarrollado para la codificación de la aplicación pedida.

4.1. Tablero

La clase Tablero fue planteada para tener ubicaciones basadas en coordenadas (x,y) señalando los objetos de la clase Casillero. El Tablero se crea vacío, con sus casilleros en un Diccionario, y al ubicar una Unidad esta se guarda dentro del objeto Casillero donde se la posicione.

El tablero es la base del juego. Las unidades son ubicadas aquí por los usuarios y luego sobre él se realizan los movimientos y ataques del juego. Cada casillero es ubicado en el tablero mediante una Coordenada, otra de las clases claves del programa. Entre coordenadas del tablero existe una Distancia.

En el caso en que se quiera ubicar una Unidad en un Casillero ya ocupado se lanza una excepción que desarrollamos llamada CasilleroOcupadoException.

Por otro lado cuando una Unidad se quiere mover a un casillero que no existe, se pasa en coordenadas, se lanza una excepción llamada CasilleroInválidoException.

4.2. Jugador

En el caso de la clase Jugador va a tener una identificación en su nombre, este puede ser 1 ó 2. A cada jugador se le va otorgar un lado del tablero, donde podrá ubicar sus piezas. Además, el jugador tiene una cantidad de puntos que puede gastar en comprar unidades para el juego, las cuales serán diferentes según el jugador, uno tendrá unidades Perro y otras unidades Gato

Los jugadores juegan moviendo sus piezas por el tablero y atacando a las unidades enemigas, en otras palabras, los jugadores controlan a las Unidades por el Tablero.

Cuando un jugador elimina todas las unidades del rival el juego concluye y lo proclama ganador.

4.3. Unidad

Unidad es una clase abstracta, utilizada para que sus clases hijas hereden sus métodos. Curandero, Jinete y SoldadoDeInfanteria a su vez son hijas de Movible, la clase también abstracta que alberga a las unidades que pueden moverse. Se detallará brevemente el comportamiento de cada unidad.

Soldado de infantería: El soldado puede moverse de a un casillero por vez y cuando tiene un enemigo a un casillero de distancia puede atacarlo. Si se juntan tres soldados de infantería se puede formar un batallón (especificada la clase en la próxima subsección)

Curandero: El curandero se mueve de a un casillero por turno y su acción consiste en curar a sus aliados. El curandero no realiza ataques.

Jinete: El jinete también puede moverse de a un casillero por vez y su ataque varía según el entorno en el que esté ubicado al momento de querer realizarlo. Según las piezas aliadas y enemigas que tenga a distancia cercana o mediana utiliza su espada o su arco y flecha para el ataque.

Catapulta: La catapulta es la única unidad del juego que no puede moverse. Pero su particularidad reside en dos cuestiones: la primera es que ataca enemigos a cualquier distancia, y la segunda es que todas las unidades contiguas al enemigo, y sus contiguas, y las contiguas de estas a su vez hasta cubrir todos los casilleros contiguos ocupados, reciben daño, incluso si son unidades aliadas.

4.4. Batallón

La clase batallón agrupa unidades del mismo tipo, en este caso Soldados, aunque podría albergar otras unidades para futuras implementaciones.

Comprueba que el batallón sea válido y mueve las tres unidades en conjunto.

Nuevamente es válido aclarar que los miembros del batallón solamente pueden moverse juntos, no realizar ninguna acción de ataque.

Si un miembro del batallón trata de moverse a un casillero ocupado este quedara atrás, mientras los otros soldados se mueven.

4.5. Dirección

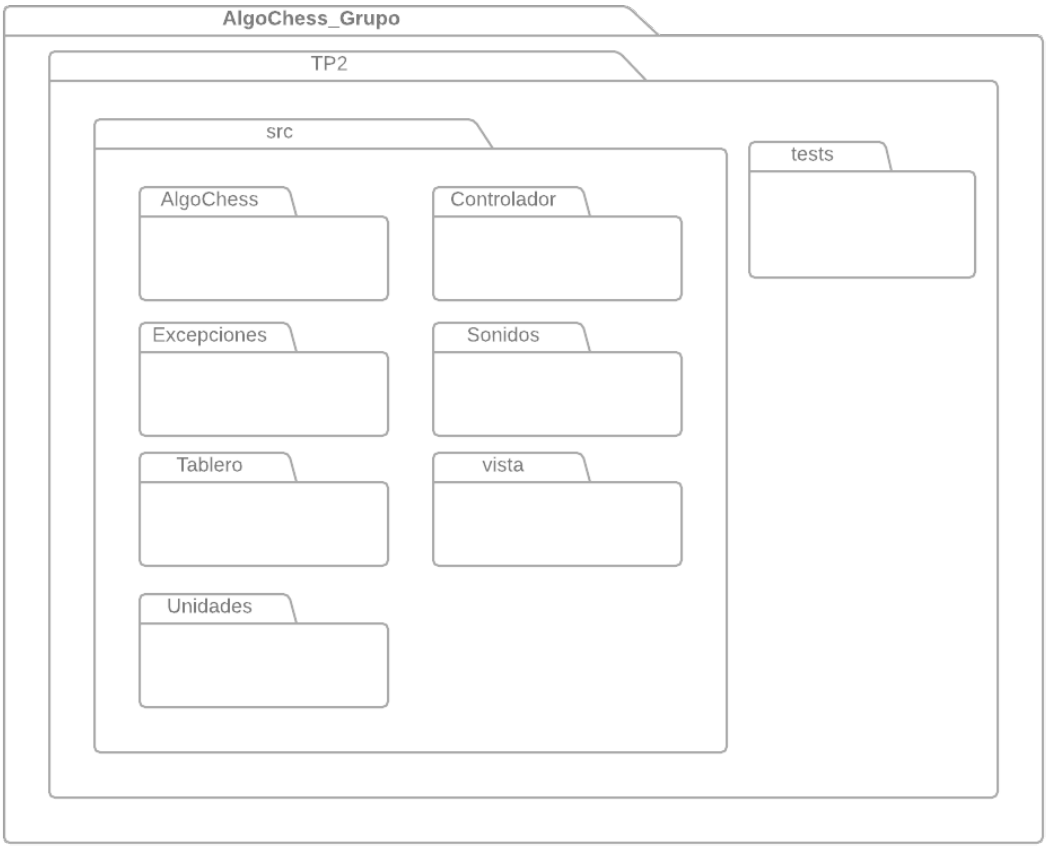
Dirección es una clase del tipo Enum de java, y su función es la siguiente:

Las unidades movibles solamente pueden moverse a un casillero de distancia, esta clase cumple la función de especificar hacia qué dirección se va a mover la Unidad.

Su función clave es recibir dos coordenadas y en base a eso retornar la dirección esperada.

5. Diagrama de Paquetes

En esta sección se podrá ver el diagrama de paquetes implementados en nuestro sistema.



6. Diagramas de secuencia

En esta sección podrán verse los diagramas de secuencia sobre los comportamientos considerados más importantes del programa AlgoChess.

El primer diagrama de secuencia presentado muestra como el jugador ubica una unidad en el tablero. En este caso el casillero elegido no esta ocupado.

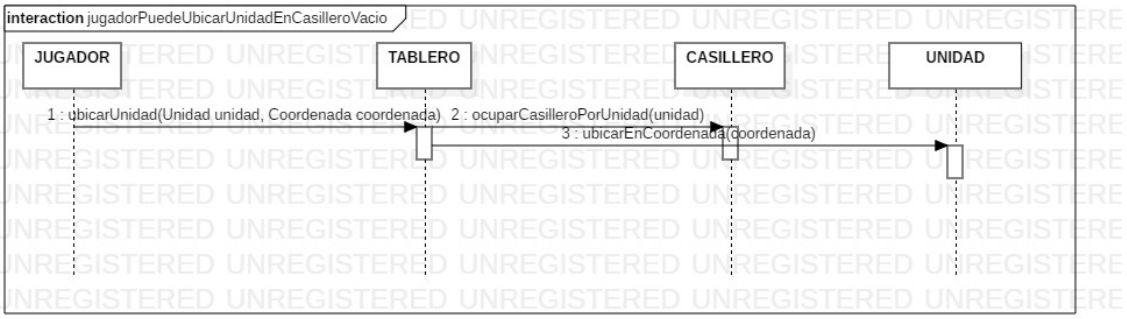


Figura 1:

El siguiente diagrama podria considerarse una ampliación del primero. Ademas de mostrar nuevamente como un jugador ubica una unidad agregar que el mismo puede mover dicha unidad. Cuando lo hace, el Casillero donde estaba ubicada la Unidad pasa a estar vacio, y el Casillero nuevo se llena con la Unidad. En el caso mostrado, el Casillero nuevo esta vacio.

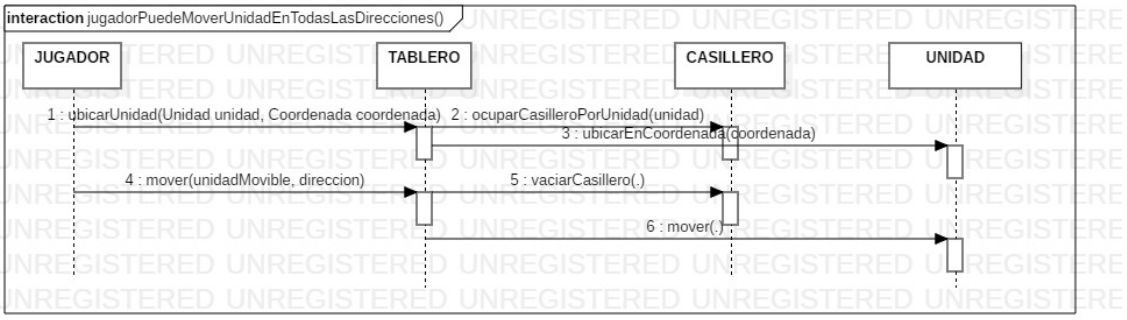


Figura 2:

El tercer diagrama de secuencia muestra como un Soldado ataca a una pieza enemiga (un Jinete), causandole daño que le resta vida.

Luego, el Curandero le aplica a dicho Jinete su realizarAccion, que es la de curar a la pieza seleccionada devoviendole una fracción de la vida perdida.

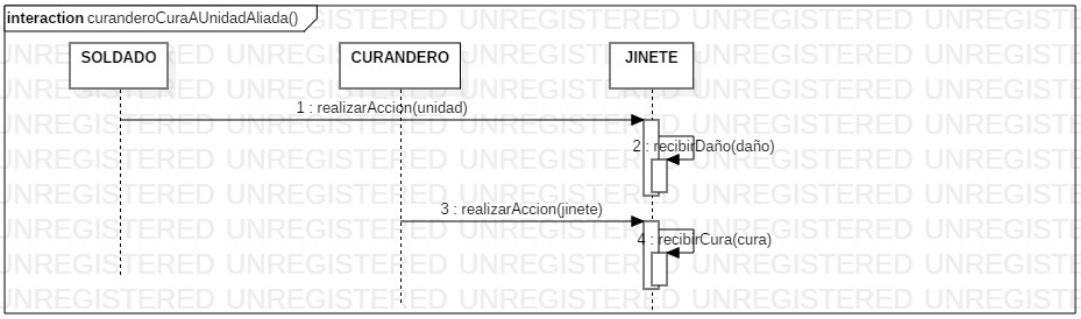


Figura 3:

Mediante el siguiente diagrama se pretende representar, de la forma mas fiel posible, el ataque de la catapulta. Una catapulta ataca a un enemigo, quien no es el unico que recibe el daño, si no que tambien lo reciben todas las unidades contiguas al mismo.

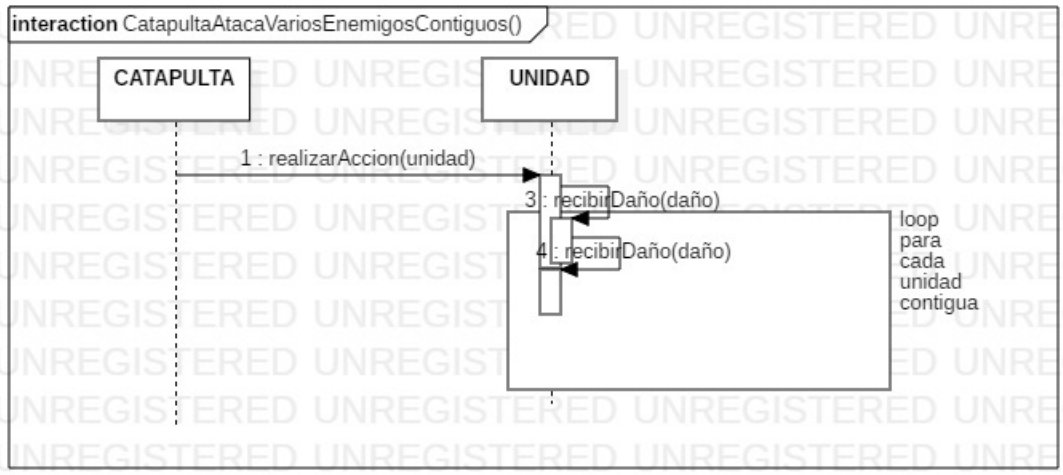


Figura 4:

El proximo diagrama presentado muestra el ataque de un Jinete a dos enemigos. Cuando el Jinete ataca al Soldado le resta una determinada vida porque la distancia es corta, y al Curandero le resta una cantidad diferente ya que la distancia es media.

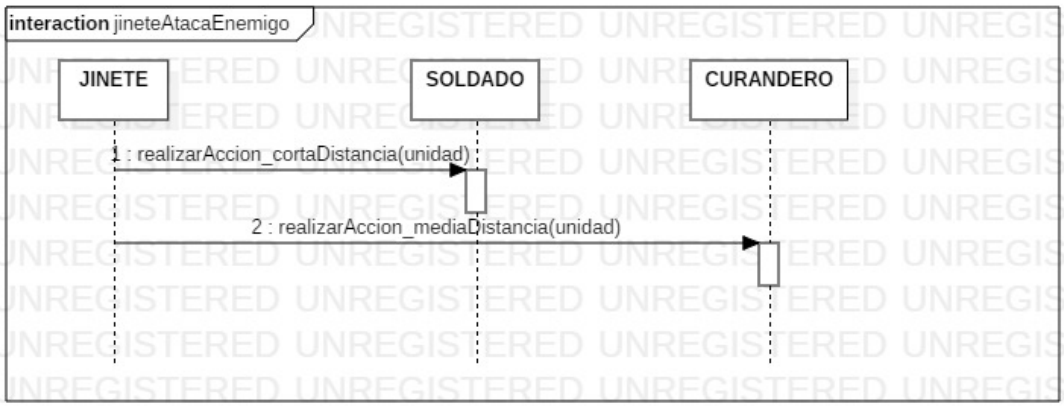


Figura 5:

El ultimo diagrama presentado sobre el flujo del juego es sobre la clase Batallon. Muestra al usuario seleccionando tres soldados, creando un batallon con ellos y luego moviendolos por el tablero.

La clase batallon se encarga de pedirle, un soldado a la vez, al Tablero que los mueva. Si algun soldado no puede moverse, permanece en su Casillero original.

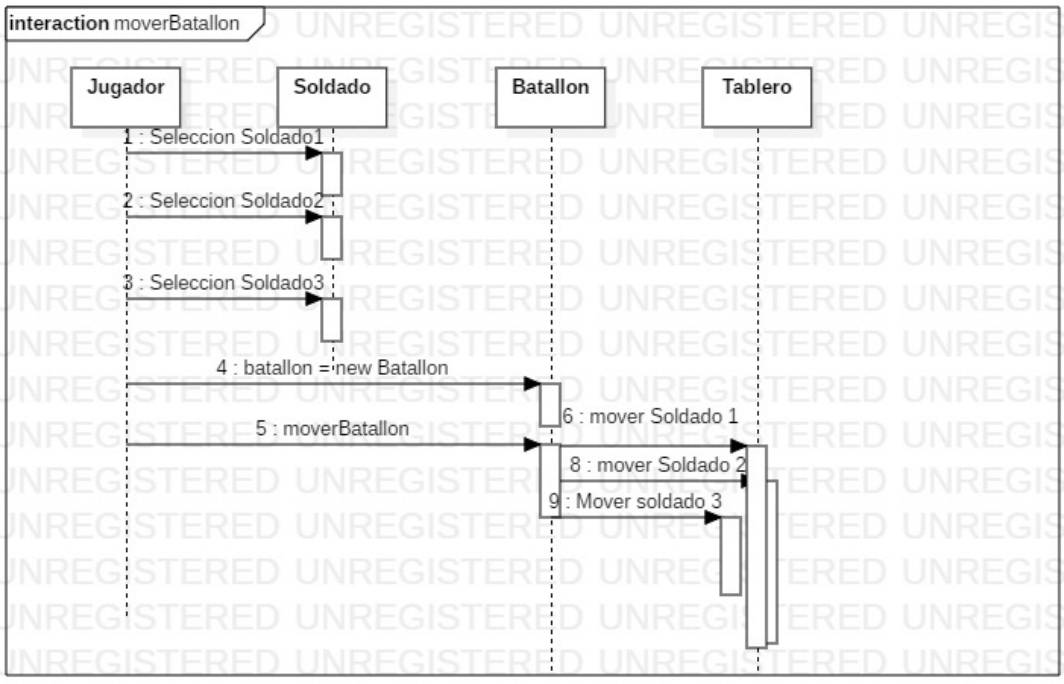


Figura 6:

7. Excepciones

Varias situaciones problemáticas del presente trabajo practico requirieron el lanzamiento de excepciones para controlarlas. Las mismas se encuentran más detallas en esta sección:

PuntosInsuficientesException: Esta excepción fue creada con el fin de lanzarla y controlar cuando un jugador quiere comprar una unidad pero no le quedan puntos suficientes para adquirirlo.

CasilleroOcupadoException: En este caso, vimos necesaria la creación de esta excepción para controlar que el jugador no pueda mover una unidad a un casillero del tablero que ya está ocupado con una unidad, ya sea propia o ajena.

CasilleroInvalidoException: Cuando un jugador intenta mover una pieza a un casillero que no existe, ya que está fuera del tablero, se lanza esta excepción.

BatallonInvalidoException: Sirve para controlar que el batallón seleccionado por el jugador sea un batallón válido.

AccionInvalidaException: Lanzamos esta excepción cuando el jugador quiere mover o atacar con su unidad y es un movimiento o ataque inválido.