FACULTAD DE INGENIERIA
Universidad de Buenos Aires

Trabajo Práctico II — A.L.T.E.G.O

[7507/9502] Algoritmos y Programación III
Curso 2
Primer cuatrimestre de 2021

Alumno:	LEVI, Dolores
Número de padrón:	105993
Email:	dolevi@fi.uba.ar

Alumno:	LAMANNA, Tobías
Número de padrón:	104126
Email:	tlamanna@fi.uba.ar

Alumno:	BIANCUZZO, Juan Ignacio
Número de padrón:	106005
Email:	jbiancuzzo@fi.uba.ar

Alumno:	SANTANDER, Valentín
Número de padrón:	105637
Email:	vsantander@fi.uba.ar

Índice

1. Supuestos

En ésta sección se señalarán las decisiones de diseño que se siguieron al momento de modelar el problema. También se mencionan algunas reglas del Juego T.E.G real que conviene tener en cuenta para comprender mejor el modelo.

- Los países se distribuyen de forma aleatoria y de forma equitativa en lo posible. Si llega a haber una cantidad de jugadores en la cual no se pueden distribuir los países de forma equitativa (Caso 3 ,4 y 6 Jugadores), el juego en sí va a decidir los Jugadores que van a tener los países de más. Ésto también es decidido de forma aleatoria.
- Cuando se distribuyen los países se le coloca a cada uno una ficha del color que le corresponde a cada jugador, en vez de darle la carta de pais al jugador y que este agregue una ficha en ese pais. De esa manera se busca imitar cómo se prepara el tablero para jugar al T.E.G en la vida real, evitando al usuario tener que armar el tablero manualmente.
- Un jugador podrá solicitar una carta de país sólo si durante su turno conquistó al menos un país. Sin embargo no son acumulativas. En caso de que se conquiste más de uno, solo se podrá recibir una carta, la cual siempre es aleatoria. Cabe destacar que las conquistas tampoco se acumulan entre rondas. Por ejemplo, si un Jugador conquistó 2 países en un turno, deberá conquistar uno nuevo en el siguiente para poder reclamar una Carta.
- Cuando un Jugador desee cambiar sus cartas por más fichas, no podrá elegir con cuales cartas se desea hacer el intercambio. El canje se realiza siempre automáticamente. Las cartas obtenidas primero tienen prioridad sobre las últimas. En otras palabras, se eligen las primeras 3 cartas que generan un canje eligiendolas en orden de llegada.
- En el T.E.G original la Ronda general, esta compuesta por 4 partes: Incorporación, Ataque, Reagrupación y Solicitar Carta de país. Se puede incorporar fichas por los paises conquistados, canjes de las tarjetas y por continentes conquistados. Al momento de jugar, vimos que da una desventaja injusta a los últimos Jugadores. Por lo tanto, y por gusto propio del grupo, decidimos separar mejor las rondas. Primero cada Jugador ataca, reagrupa y se solicita cartas, y luego se incorpora. Este cambio hace el Juego un poco más justo.
- En las reglas del T.E.G no se especifica que el usuario no ponga sus fichas en alguno de sus turnos de incorporación. Por lo tanto decidimos darle la libertad de no hacerlo en caso de que lo desee. Las fichas no utilizadas quedarán guardadas para colocarse en otro turno. Sin embargo, hacer eso no es una estrategia deseable para ganar el Juego.
- Al momento de incorporar fichas por Continente, las reglas del T.E.G indican que las mismas sólo se pueden incorporar en los países que pertenezcan a ese continente en específico. Sin embargo en éste modelo se decidió omitir esa regla. Las incorporaciones pueden hacerse en cualquier país del mapa siempre y cuando ese país pertenezca al Jugador que está reincorporando.

2. Diagramas de clase

El Juego es la clase que encierra todo el comportamiento de nuestro programa. La clase Juego decide el momento en el que se realizan las rondas y los turnos de cada Jugador.

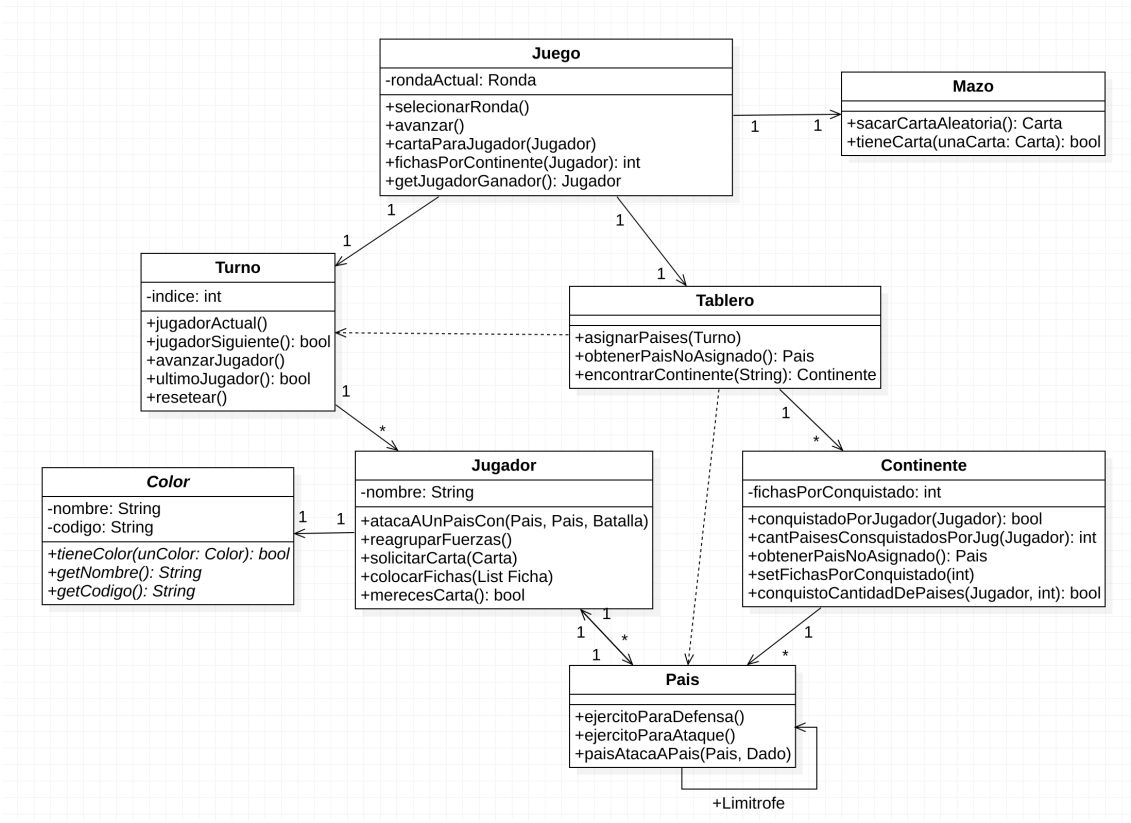


Figura 1: Diagrama de Clases General.

Nota al pie: Color es una clase abstracta que tiene a todos los colores como clases hijas. Solamente Jugador tiene como atributo a una color. Dado es una interfaz que esta implementado por DadoEstandar, que seria un dado comun de 6 lados. La utilidad de esta interfaz es que en un futuro se podria implementar un dado de 20 lados que extienda de dicha interfaz sin tener que modificar el comportamiento de la batalla.

Para lograr el ritmo y la division del juego, el Juego delega la respondabilidad de las rondas, a la clase Ronda, que determina que accion se tiene que hacer en ese momento, y prepara el juego para que se pueda ejecutar de la forma deseada. En Detalles de Implementación hay una explicación más completa de ésta Clase.

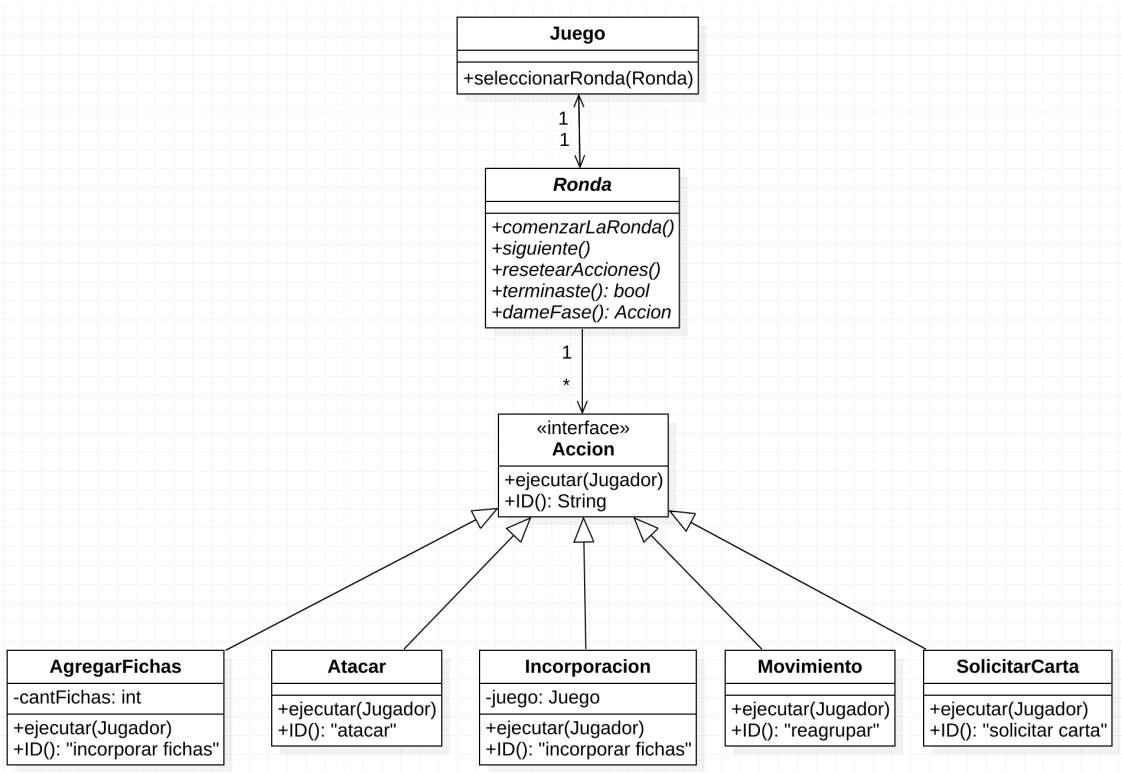


Figura 2: Diagrama de Clases Ronda

Aparte de Ronda, Juego tiene la clase Turno como atributo, y este nos permite abstraernos de cuantos jugadores hay, y si uno de los jugadores no esta, permitiendo a los metodos que usan Turno, no preocuparse por esta complejidad y simplemente tener el jugador actual.

Otra entidad importante que tiene Juego es Tablero. Tablero conoce a los Continentes. Cada continente tiene guardados sus respectivos países. Tablero tiene la responsabilidad de distribuir los Paises entre los Jugadores antes de comenzar la partida, por lo cual también tiene una referencia a Turno.

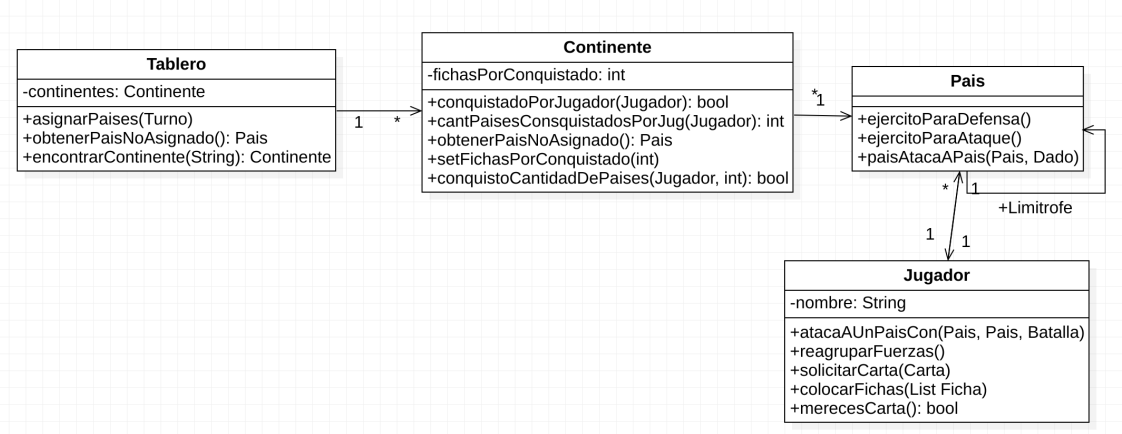


Figura 3: Diagrama de Clases Tablero

Juego también tiene una instancia de Mazo que contiene todas las instancias de las Cartas de País. El Mazo le entrega una Carta aleatoria al Jugador cuando éste conquista al menos un País en una Ronda.

Cada Jugador un nombre y un color, pero estos no necesariamente son caracteristicos de este, ya que no necesitamos deferenciarlos (la responsabilidad de diferenciarlos lo tiene el turno, pero no usa sus nombre, ni colores, para hacerlo). Cada Jugador tiene su propia instancia de Canjeador que guarda las Cartas de pais que gano el Jugador, para después intercambiarlas por más Fichas. Una vez que se intercambian las Cartas, éstas vuelven al Mazo y pueden ser adquiridas por cualquier otro jugador.

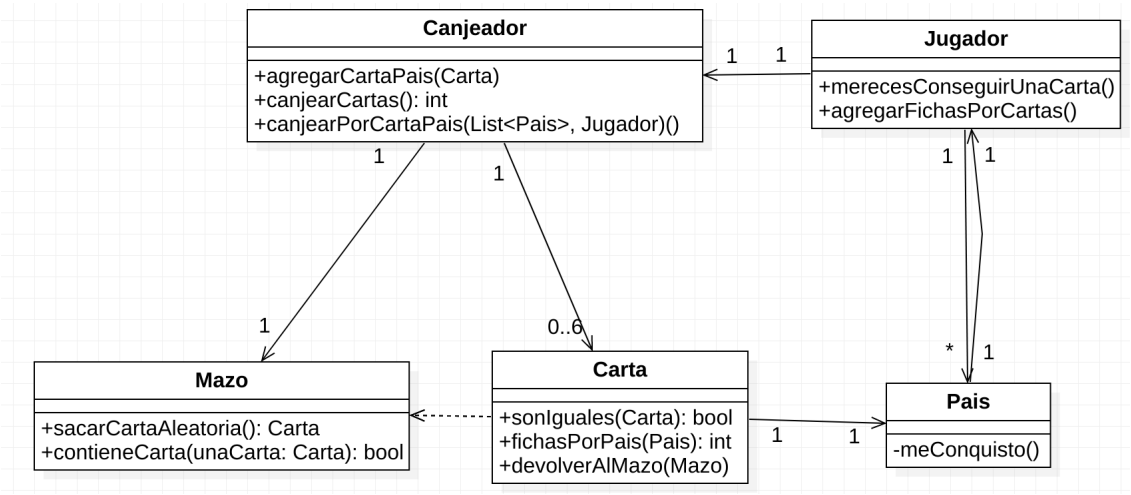


Figura 4: Diagrama de Clases Canjeador

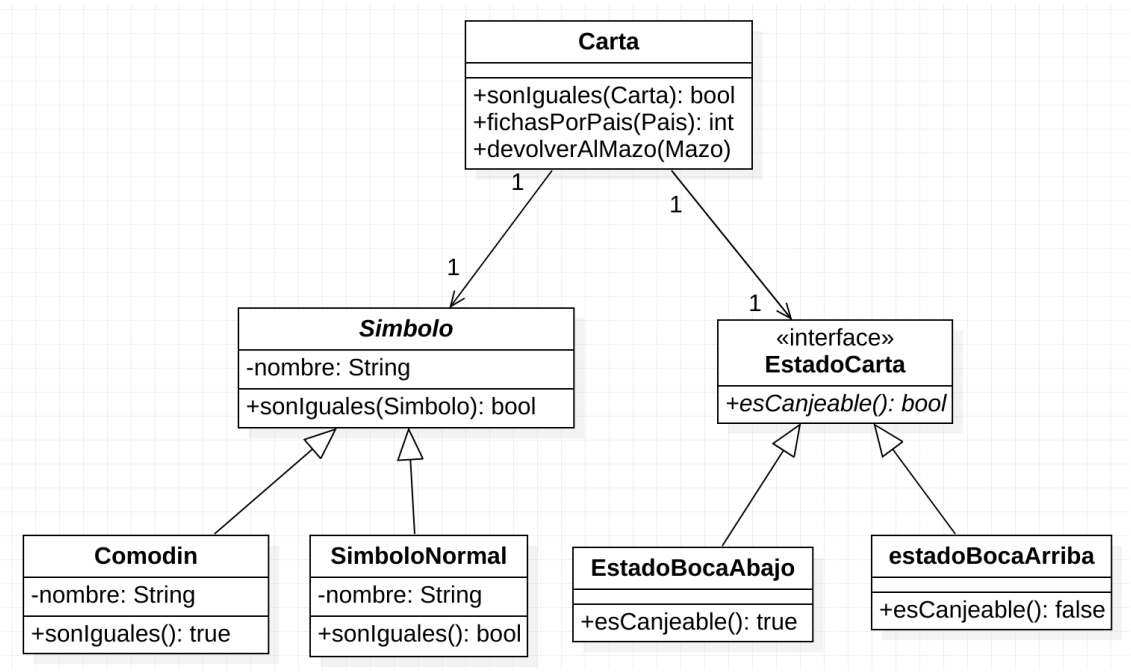


Figura 5: Diagrama de Clases Carta

Cada Jugador tiene dos Objetivos, el ObjetivoComun y un ObjetivoCompuesto, el ultimo con sus sub-ojetivos elegidos de forma aleatoria. Este último, como mencionado anteriormente, es un Objetivo que está creado por varios sub-objetivos: ObjetivoConquistaContinente, ObjetivoCantidadPaisesEnContinente u ObjetivoEliminarJugador. Después de cada Ronda de cada accion, se verifica si el Jugador cumplió alguno de sus Objetivos. De ser así, este es declarado como ganador y se termina la partida. Un Jugador gana la partida cuando, o bien cumple el ObjetivoGeneral o bien cumple todos los sub-objetivos del ObjetivoCompuesto.

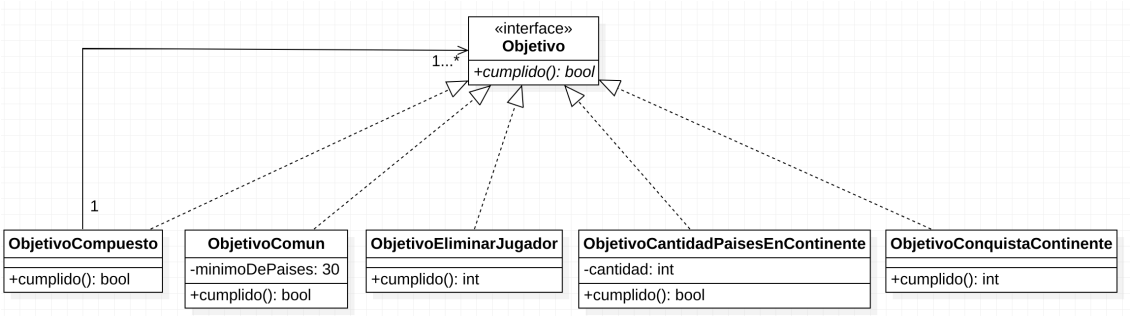


Figura 6: Diagrama de Clases Objetivo pt.1

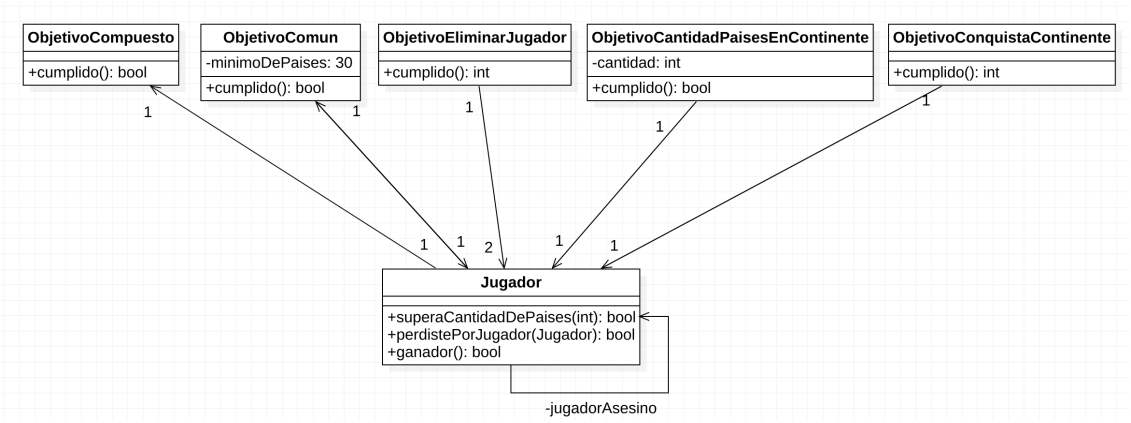


Figura 7: Diagrama de Clases Objetivo pt.2

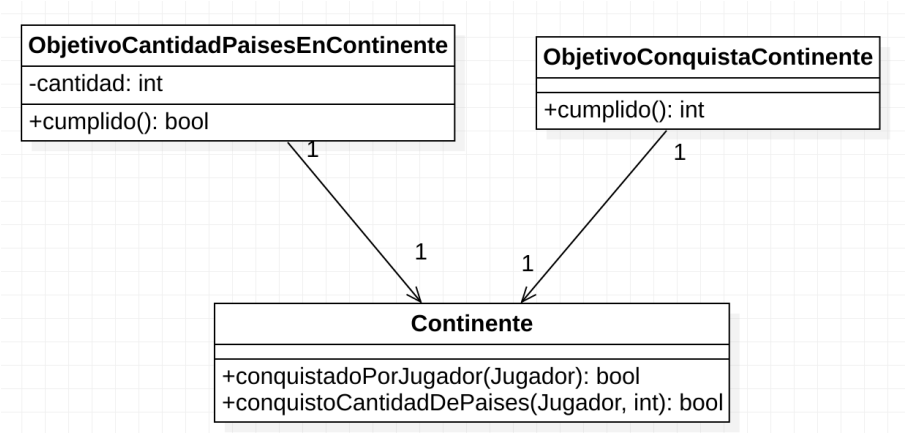


Figura 8: Diagrama de Clases Objetivo pt.3

Cuando un Pais quiere atacar a otro Pais, se crea la clase Batalla. En esta clase se encapsula todo el comportamiento de una ataque. Toda la validación sobre el ataque (si el país atacante tiene suficientes fichas, si es limitrofe al otro, y si no son del mismo equipo) se hacen fuera de esta clase, por lo cual, cuando se crea Batalla se puede tomar al ataque como seguro, liberando a Batalla de todas las responsabilidades que no tengan que ver con el ataque en sí.

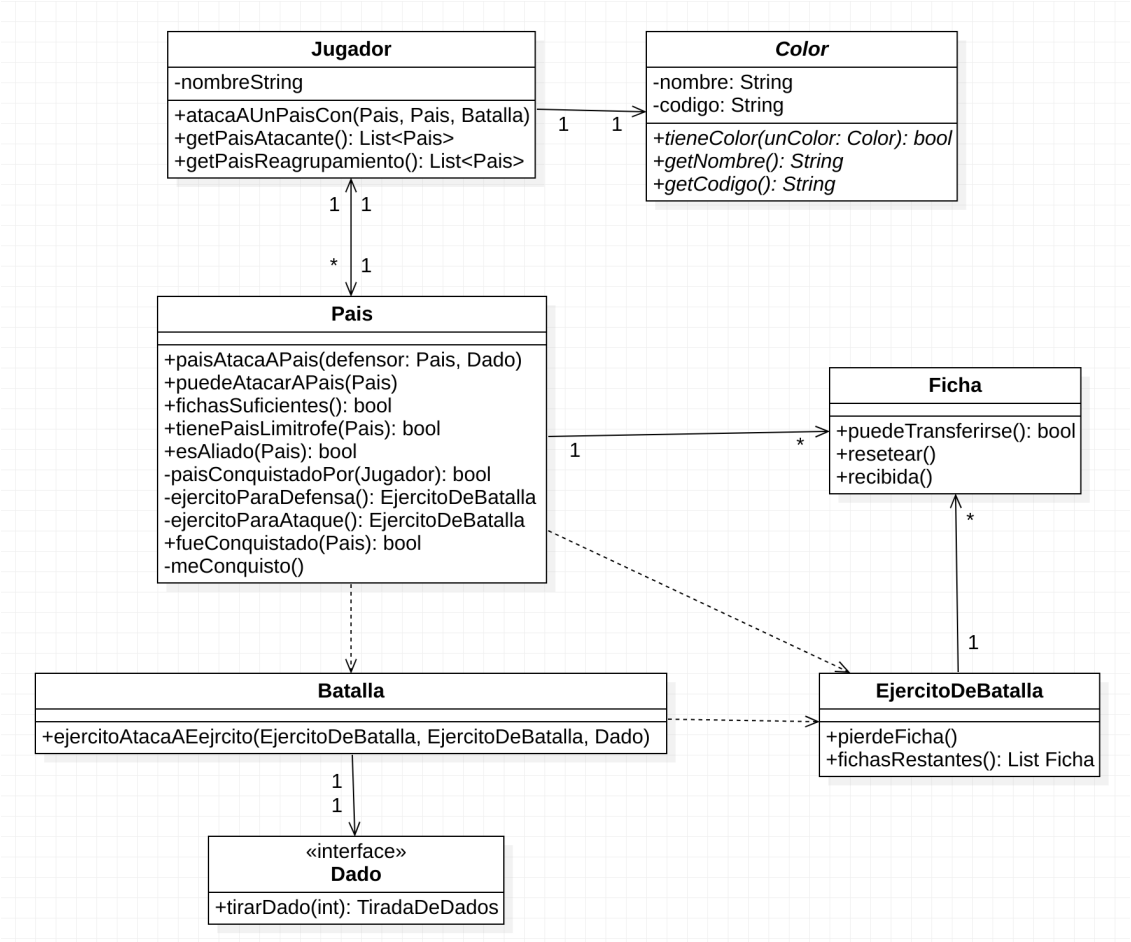


Figura 9: Diagrama de Clases Pais con Jugador.

Batalla es accedida por País una vez que se compruebe la validez del ataque. Batalla luego se encarga de organizar el ataque y controlar quién gana y pierde cierta cantidad de fichas.

Las fichas que tiene un Pais en el juego tradicional de T.E.G tienen su correlacion en el programa en forma del objeto Ficha. Esto nos permite mantenerlos lo más cerca posible al concepto del juego al crear nuestro programa. Estas se estructuran de la siguiente forma:

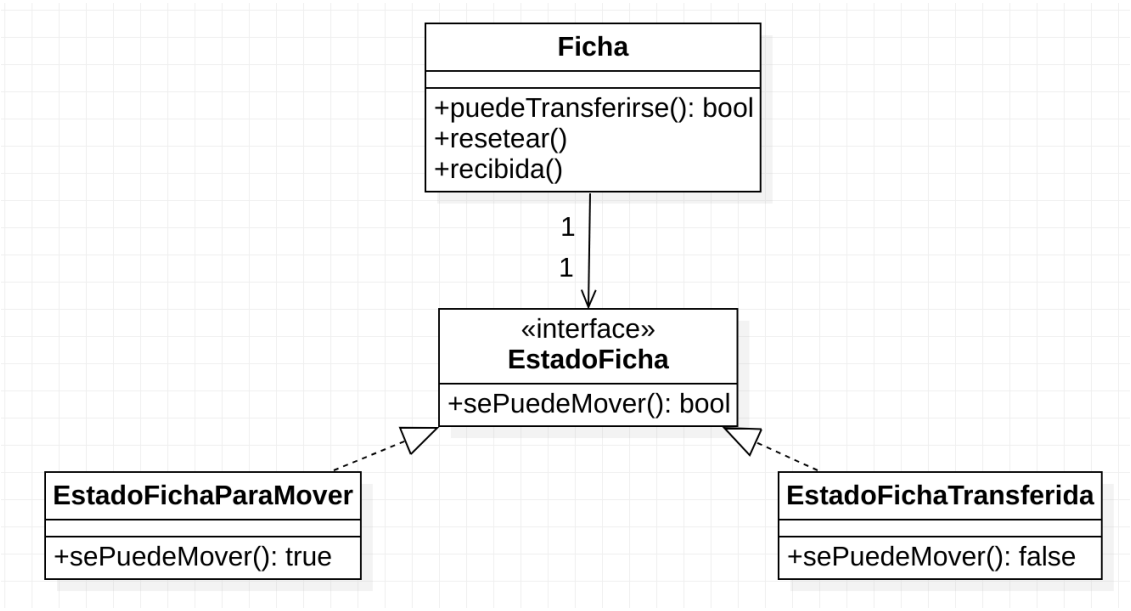


Figura 10: Diagrama de Clases Ficha

Como se puede ver en lo diagramas, usamos el patrón State varias veces: en Ronda con sus Acciones, en Carta con su EstadoCarta (la cual se usa para decidir si debe devolver Fichas o no), y en Ficha con su EstadoFicha (que se usa para decidir si una Ficha es transferible entre Países).

3. Diagramas de secuencia

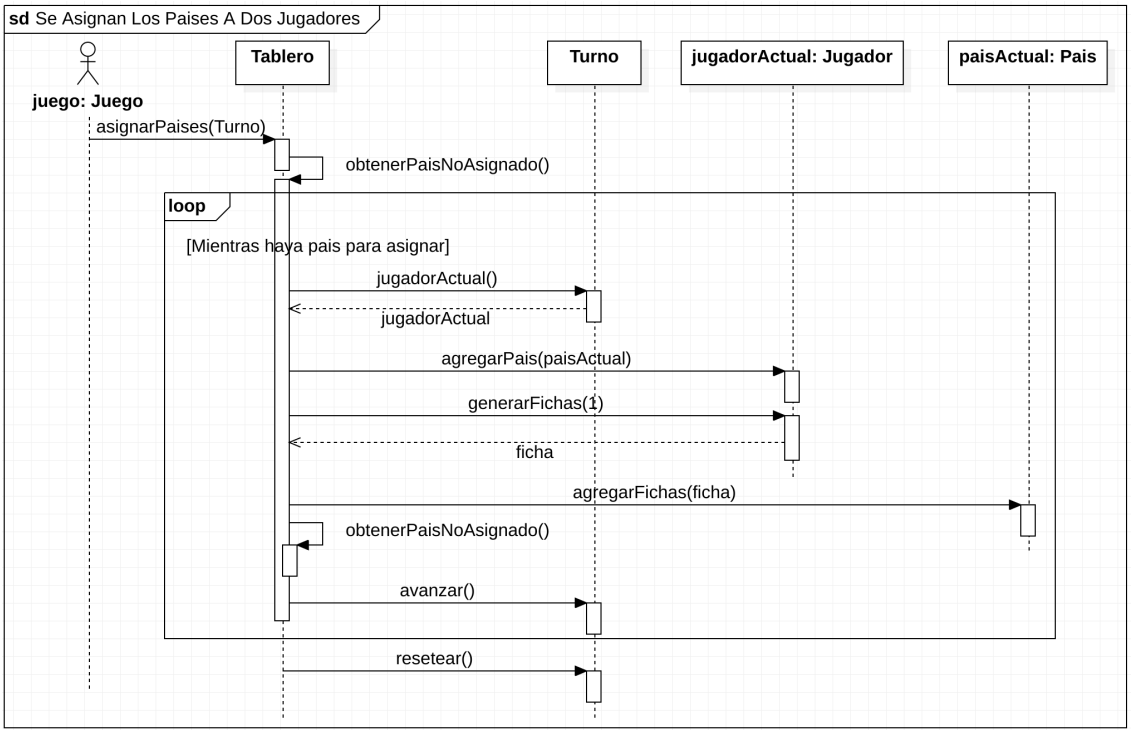


Figura 11: Diagrama de Secuencia 1: Asignación de Paises a dos Jugadores.

En este diagrama de secuencia, lo más interesante es que podemos ver como el tablero no se tiene que preocupar por el orden de los jugadores, o si uno esta jugando o no, ya que esto es responsabilidad de turno. Tablero le interesa el jugador actual, y ver como avanzar al siguiente.

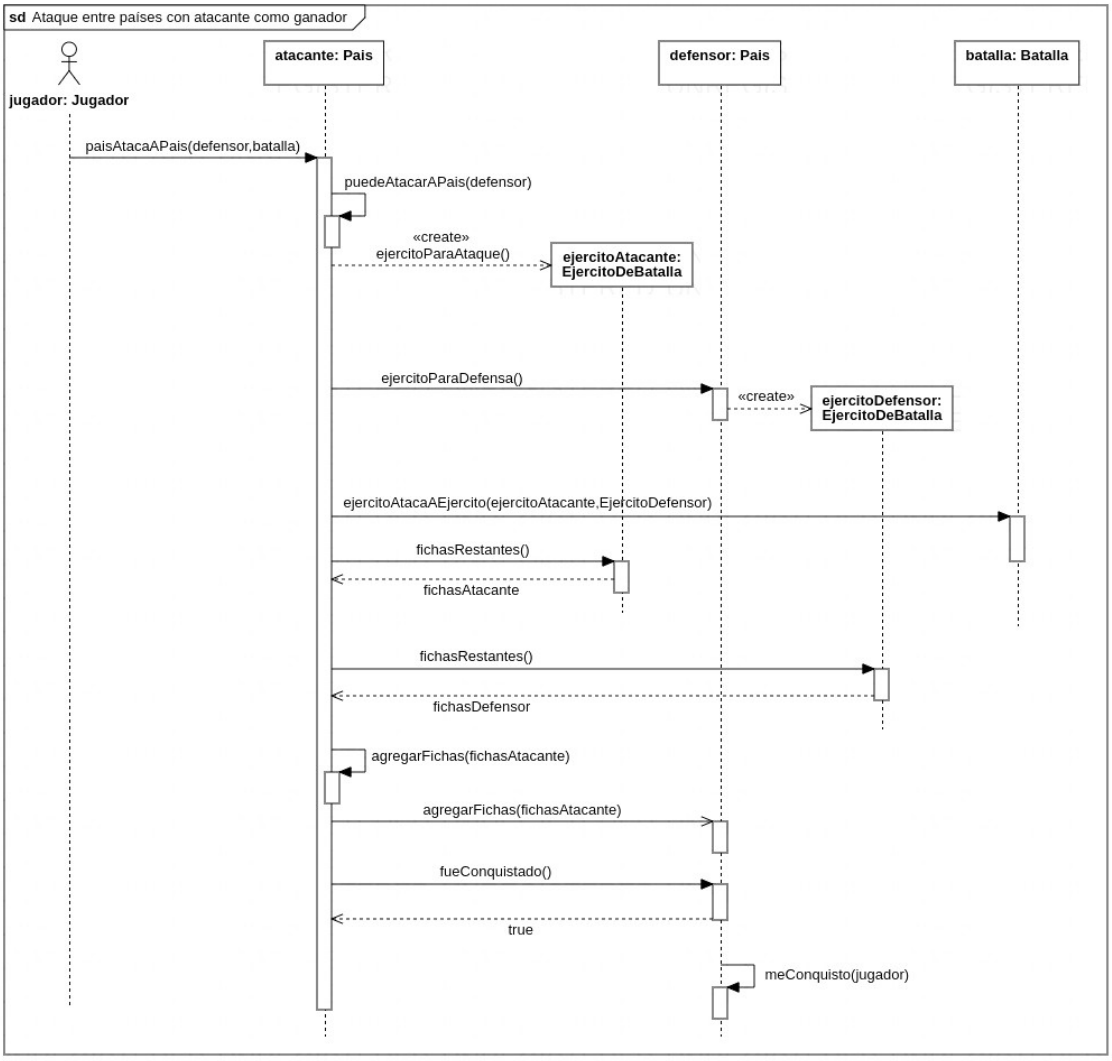


Figura 12: Diagrama de Secuencia 2: Pais ataca a otro Pais.

El diagrama muestra únicamente el caso en el que el ganador es el atacante y conquista el país defensor. Si el atacante pierde, el diagrama sería un poco diferente. El mensaje `fueConquistado()` devolvería `false`, entonces el mensaje `meConquisto()` no se ejecutaría dejando a cada Jugador con su respectivo país.

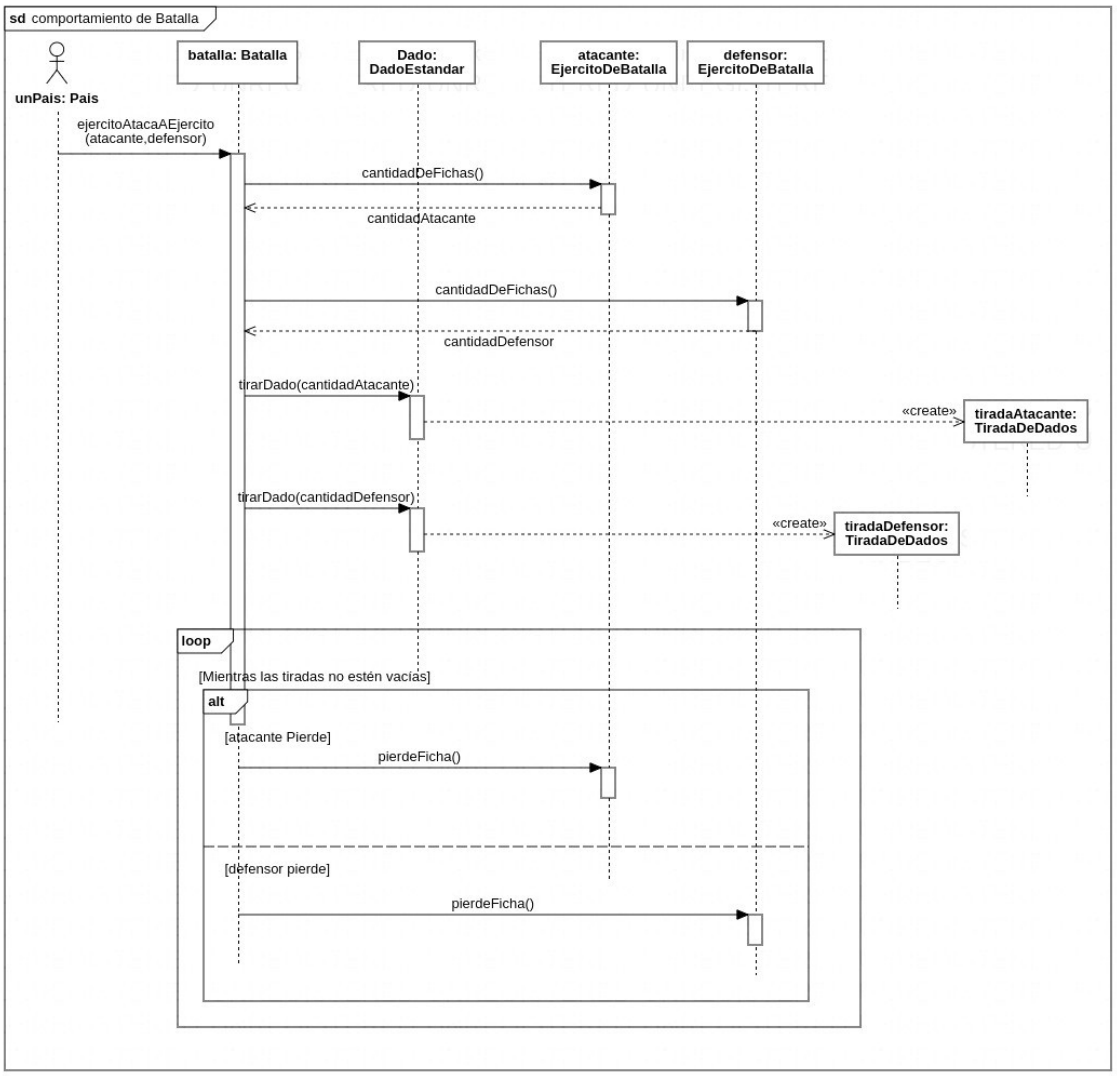


Figura 13: Diagrama de Secuencia 3: Batalla generica.

En el Diagrama de Secuencia 2 muestra el mensaje

```
ejercitoAtacaAEjercito(ejercitoAtacante,ejercitoDefensor)
```

Sin embargo se decidió describir la lógica de Batalla en éste otro diagrama para que sea más legible y prolijo. Cabe destacar que como la batalla no necesita conocer el pais atacante ni el defensor, sino con cuantas fichas se produce esta batalla, hay un menor acoplamiento entre ambas clases.

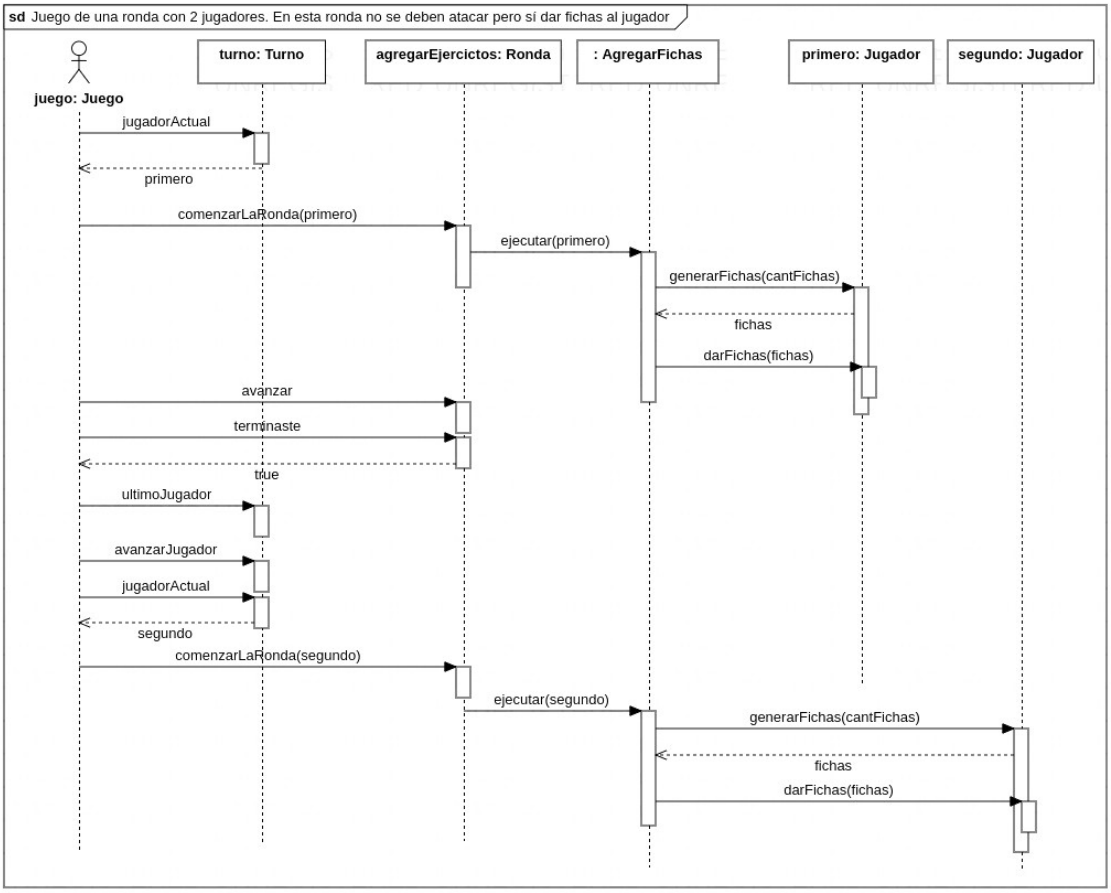


Figura 14: Diagrama de Secuencia 4: Juego de una ronda con 2 jugadores, y en esta ronda se agregan fichas.

Para simplificar la lectura del diagrama, se omitió la colocación de fichas en un país determinado. Lo que se intenta mostrar es el mecanismo de rondas y el manejo de turnos con el jugador actual.

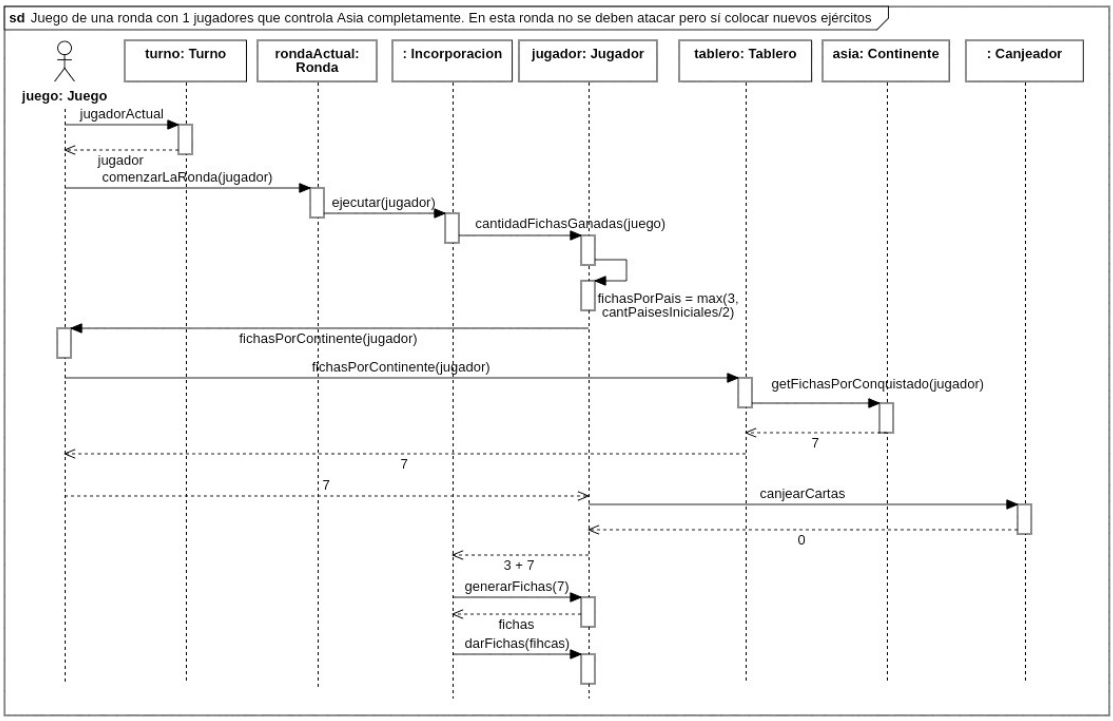
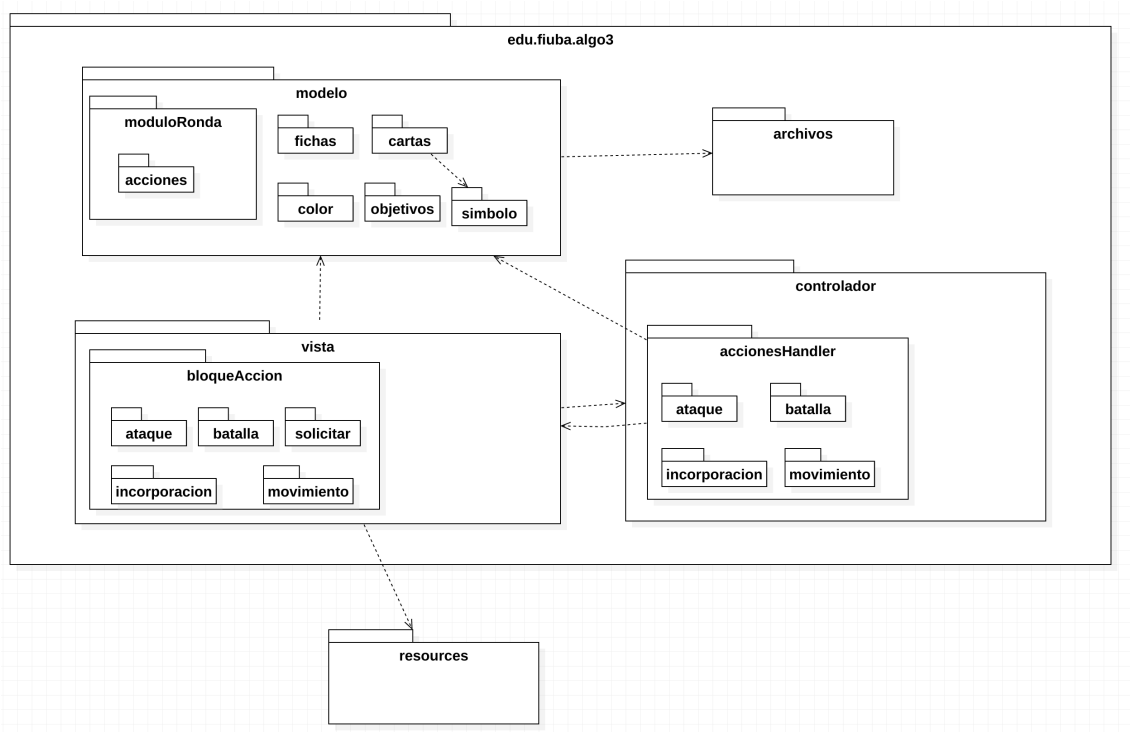


Figura 15: Diagrama de Secuencia 5: Juego de una ronda con un jugador que controla Asia completamente e incorpora fichas.

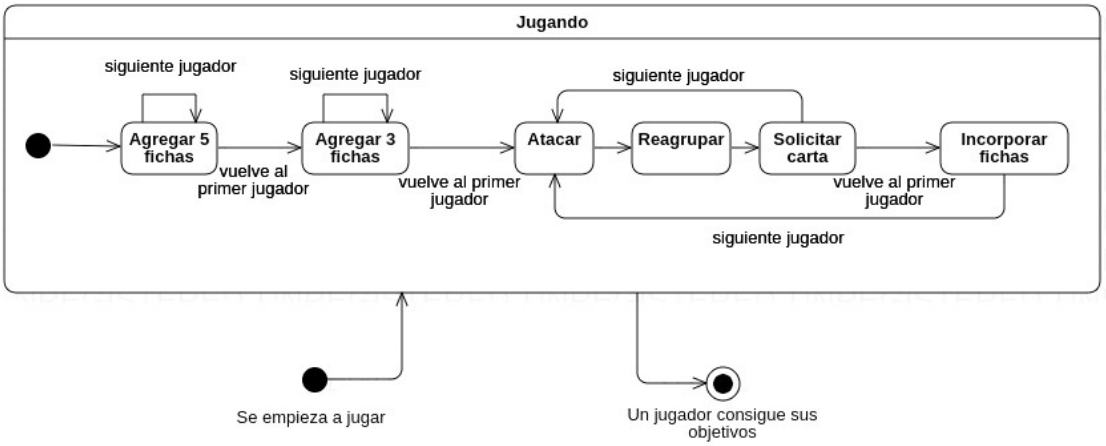
Cuando Asia es conquistado, este le debe dar siete fichas al Jugador por el cual fue conquistado. En este caso, para simplificar el testeo, tablero solo tiene al continente Asia.

4. Diagramas de paquetes



Paquetes como batalla dentro de vista importan a otros paquetes dentro de accionesHandler, pero como la vista ya importa al controlador, decidimos no agregar estas flechas ya que se sobreentiende lo ocurrido. Para organizar nuestro proyecto usamos los patrones MVC y Observer.

5. Diagramas de estado



El Juego comienza cuando luego de elegir la cantidad de jugadores y sus respectivos colores. A partir de éste momento, el juego se mantiene en estado *Jugando* hasta que uno de los jugadores cumpla con las condiciones de victoria. Primero cada Jugador coloca 5 fichas. Luego todos colocan

3. A partir de ahora cada Jugador realizará el mismo ciclo de 3 acciones: Atacar, Reagrupar y Solicitar Carta. Cuando el último Jugador termine, cada Jugador a partir del primero podrá incorporar fichas en sus países. Cuando todos incorporen, vuelve a comenzar el ciclo de 3 rondas y se repite todo hasta que haya un ganador.

6. Detalles de implementación

6.1. SetUpJuego

SetUpJuego es la clase que crea todas las entidades del modelo (lo cual incluye leer los archivos correspondientes). Una vez que el Juego es creado a partir del SetUp, no se vuelve a utilizar.

6.2. LeerArchivo

Esta clase no se muestra en los diagramas de clases ya que su única responsabilidad es leer los archivos de texto que contienen la información sobre los países, sus limítrofes y sus continentes. Esta clase luego procesa la información de tal manera que luego sea fácil de leer y convertir en objetos. Por todo esto, esta clase es solamente utilizada en SetUpJuego.

6.3. Batalla

Batalla es la clase que se encarga de realizar el ataque entre dos Países. La idea es que antes de llamar a Batalla, Jugador verifique si el ataque es válido, y luego recién ahí se puede crear una Batalla. Luego, los dos Países involucrados crean su propio EjercitoDeBatalla, el cual dependerá de si fueron creados para atacar o para defender. Cuando un Jugador pierde o gana una batalla, se cambia la cantidad de instancias de Ficha que tiene guardado el País. Y si se llega a conquistar el Pais, se le remueve el Pais al Jugador perdedor, se le agrega al ganador, y se le cambia el Color al Pais.

6.4. Ronda

Ronda tiene toda la responsabilidad de separar al juego en etapas. Cada Ronda tiene una lista de Acciones que le avisan a la Vista su estado, para que esta pueda decidir que mostrarle al usuario. Para lograr esto usamos el patron State, ya que cada Accion se lo puede tomar como el estado actual de la Ronda lo cual cambia su comportamiento.

6.5. TiradaDeDados y Dado

TiradaDeDados es una clase que se encarga de almacenar los valores que genera un Dado ordenándolos de mayor a menor. Además, entre instancias de TiradaDeDados, se pueden comparar los primeros elementos de ambas tiradas.

Dado se trata de una Interface. La única responsabilidad que tiene un Dado es la de devolver una instancia de TiradaDeDados mediante el método `tirarDado(Integer cantidadATirar)`. Según el descendiente de la interfaz, se van a agregar los valores en la tirada con un determinado criterio. Por ejemplo la clase DadoEstandar implementa el método `tirarDado` asignando valores del 1 al 6.

A través de ésta interfaz, se puede redefinir el método `tirarDado(Integer cantidadATirar)` según el comportamiento que el programador/a desee implementar. De ésta manera, se busca seguir el principio de Inversión de Control haciendo que sea más sencillo agregar nuevos comportamientos de Dado.

6.6. Turno

SetUp ya ingresa los Jugadores a Turno en forma aleatoria, por lo cual Turno solo tiene que recorrer su lista interior sin antes ordenarla. Se avanza de Jugador de forma manual para facilitar la implementacion con Ronda. Los Jugadores que perdieron no se sacan de la lista, sino que se ignoran cuando se avanza de Jugador. Cuando se llaman a los metodos `resetear` y `ultimoJugador` se ignoran otra vez los Jugadores que perdieron.

6.7. Canjeador

Cada Jugador tiene su propia instancia de Canjeador. Este se encarga de guardar las Cartas para que despues el Jugador las pueda intercambiar por Fichas cuando se este en el estado de Incorporar Ejercitos en la ultima ronda.

6.8. Mazo

Al momento de iniciar el Juego, se crea una instancia de Mazo a la cual se le agregan todas las Cartas de países. El Canjeador recibe por parámetro ese Mazo. De ésta manera el Canjeador puede devolver las cartas de nuevo al Mazo al momento de hacer un Canje por Cartas. Mazo también se usa para poder obtener instancias de Carta de manera aleatoria. Esas cartas van a parar al Canjeador correspondiente al Jugador que conquistó un Pais en un turno.

6.9. Objetivo compuesto

ObjetivoCompuesto tiene uno o varias clases que heredan de Objetivo ya que ObjticoCompuesto no debe saber que clases exactas tiene para poder mantener la abstraccion. Gracias a esto y al uso de polimorfismo, cuando se le pregunta al ObjetivoCompuesto si fue cumplido, este solo debe iterar la lista de Objetivos que tiene y preguntarles lo mismo.

7. Excepciones

AtaqueAPaisAliadoExcepcion Dos países son aliados cuando pertenecen al mismo Jugador. Según las reglas del juego, no es posible atacar a tus propias tropas. Además no sería una estrategia deseable para un jugador.

EjercitosInsuficientesExcepcion Para mantener el estado válido del juego, es necesario que todos los países tengan al menos un ejército en todo momento. Por lo tanto, no se puede permitir que un país ataque con solo una ficha. En caso de que gane, no tendría fichas para conquistar ese país, lo que dejaría al juego en un estado inválido.

NoEsLimitrofeExcepcion Las reglas del juego indican que sólo es posible realizar un ataque entre países si sus fronteras se tocan o si existe un puente entre ambos. Ésta excepción se lanza cuando se intenta realizar un ataque entre países no limitrofes.