TRABAJO PRÁCTICO N°2

Algoritmos y Programación III

Grupo: 9

Integrantes:
- Lucía Liceri Martinez
- Nahuel Gomez
- Camila Stahl

Correctores: Pablo Massuh - Maia Naftali

Informe primera entrega T.E.G.

Se desarrollaron pruebas en las que se verifica que un país pueda atacar a otro y cómo se desencadena el resultado del ataque. Realizando pair-programming se llevó a cabo el código que hace pasar las pruebas, como lo indica el TDD.

Informe segunda entrega T.E.G.

Consigna:

- Activación de las tarjetas de país en la ronda de colocación. ✓
- Juego de una ronda con 2 jugadores. En esta ronda no se deben atacar pero sí colocar nuevos ejércitos. ✓
- Juego de una ronda con 3 jugadores (el jugador 2 controla Asia completamente). En esta ronda no se deben atacar pero sí colocar nuevos ejércitos.✓
- Juego de una ronda con 2 jugadores. En esta ronda el jugador 1 ataca y conquista dos países del jugador 2..✓

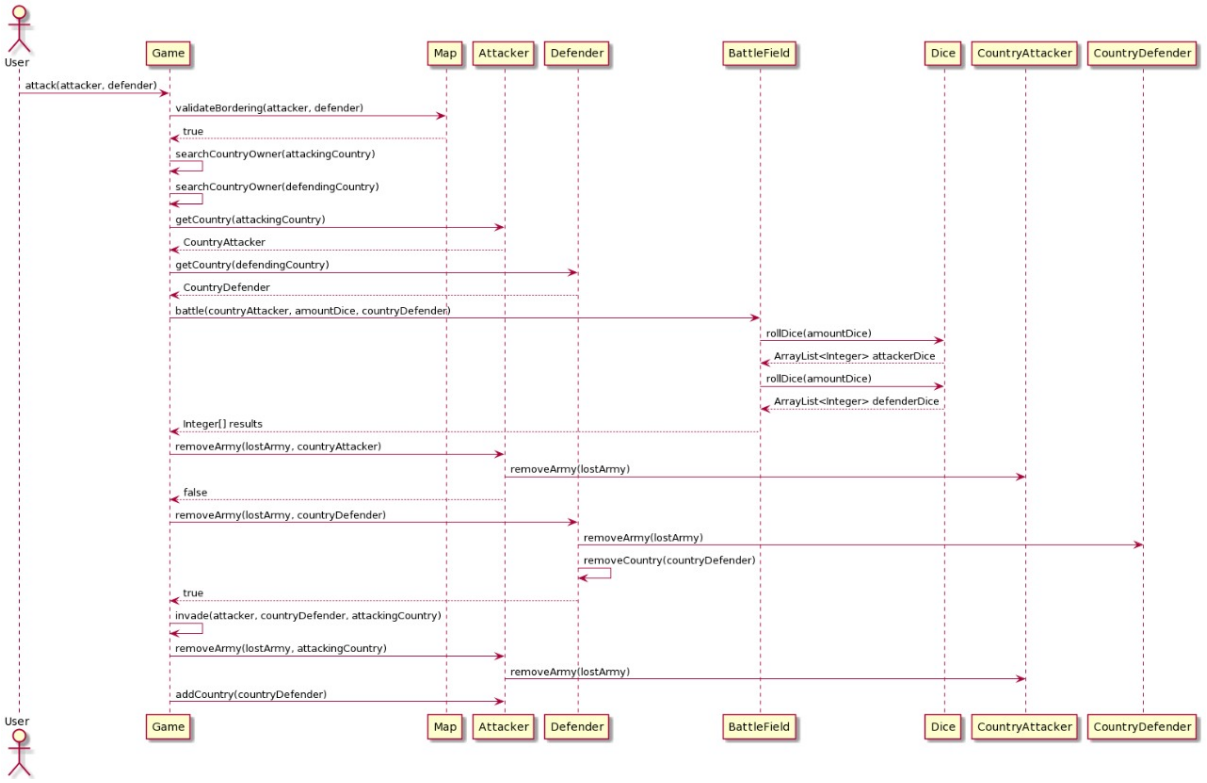
El los cambios que se añadieron en el modelo para cumplir con la consigna otorgada fueron:

- Se agregó la clase CountryCards, la cual corresponde a las cartas que se reparten al principio del juego para inicializar correctamente a cada jugador con una cierta cantidad de países en su dominio.
- Se agregó la clase Continent, la cual representa los continentes del mapa. Cada continente tiene un listado de los países contenidos en él.
- Se hizo uso de dos archivos de tipo csv, los cuales contienen la información de las cartas de países del ítem anterior, y del listado de países y continentes necesario para el funcionamiento del mapa.

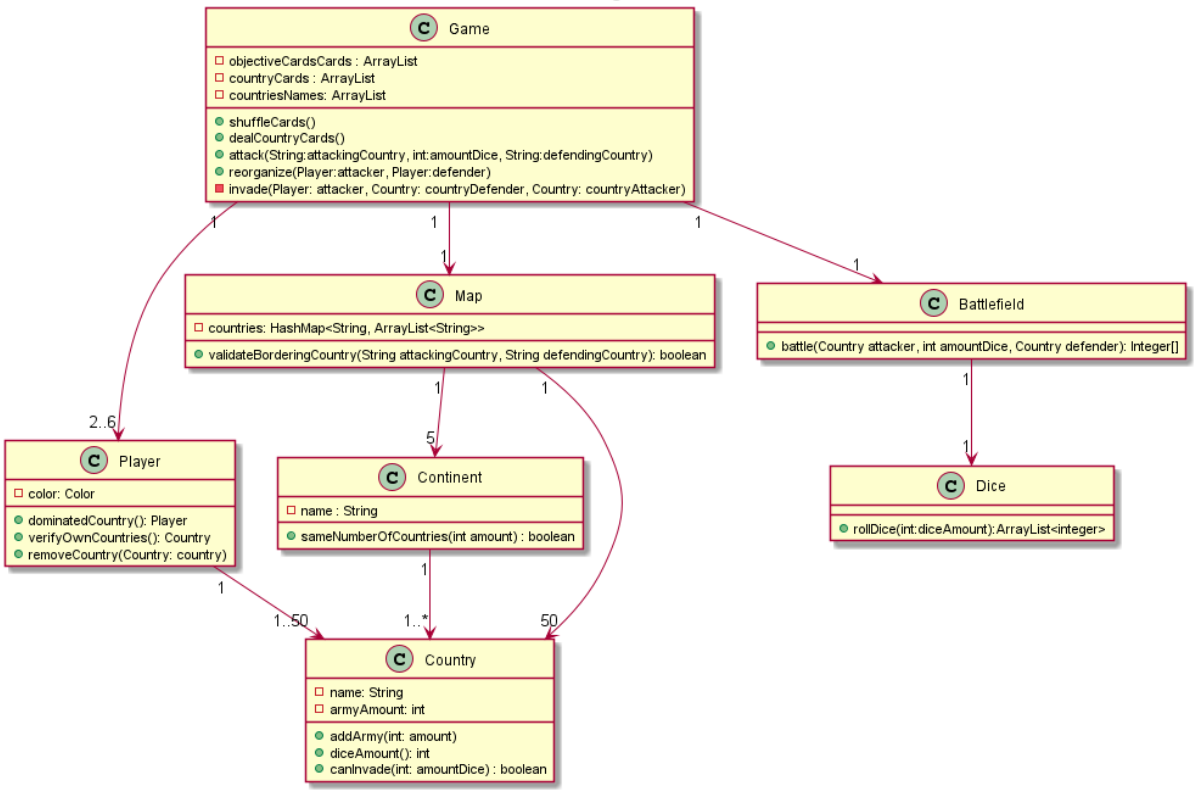
- También se agregó una clase Round, de la cual heredan PlacementRound y AttackRound, quienes se encargan de manejar el flujo de las rondas de colocación y ataque.

- También se implementaron las excepciones correspondientes para los casos límites que se dieron en las situaciones planteadas.

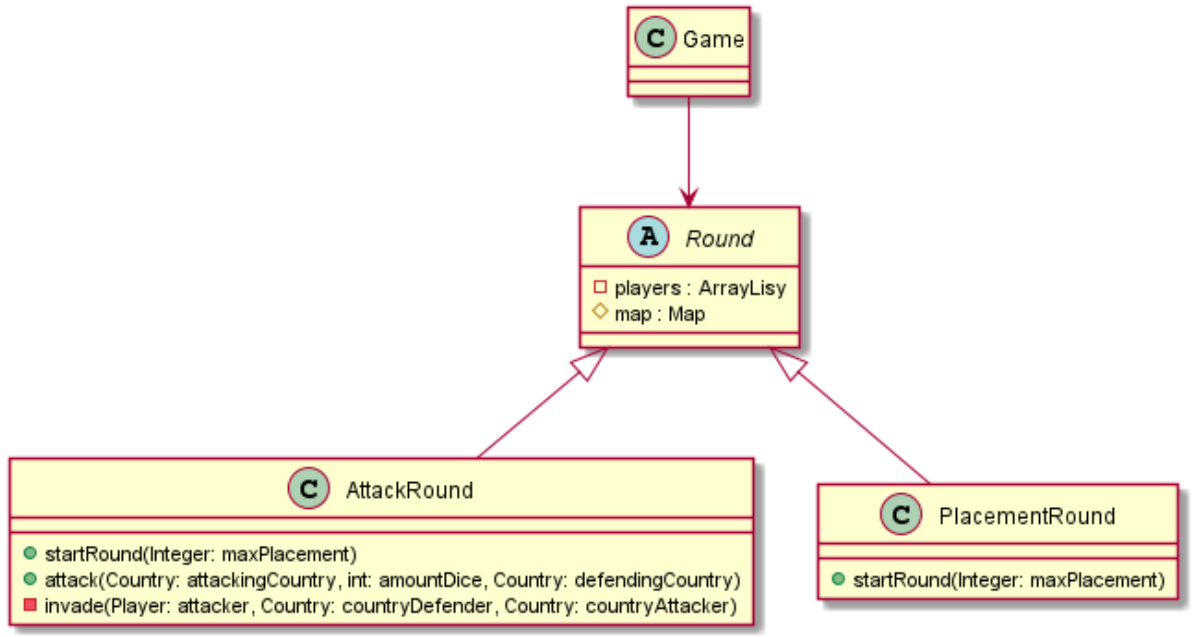
Diagramas de clase y secuencia de la entrega

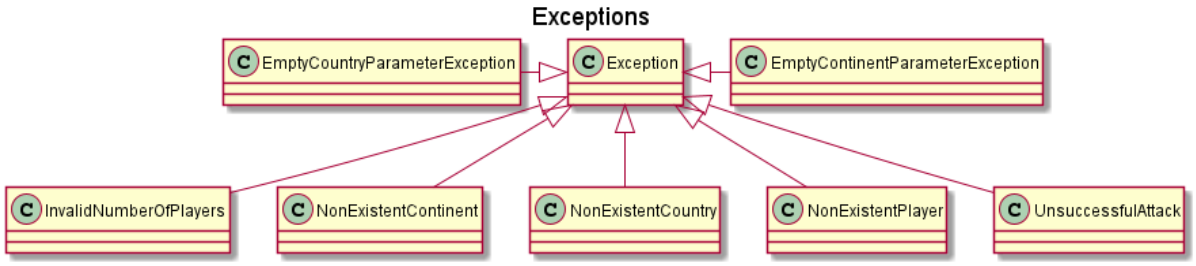


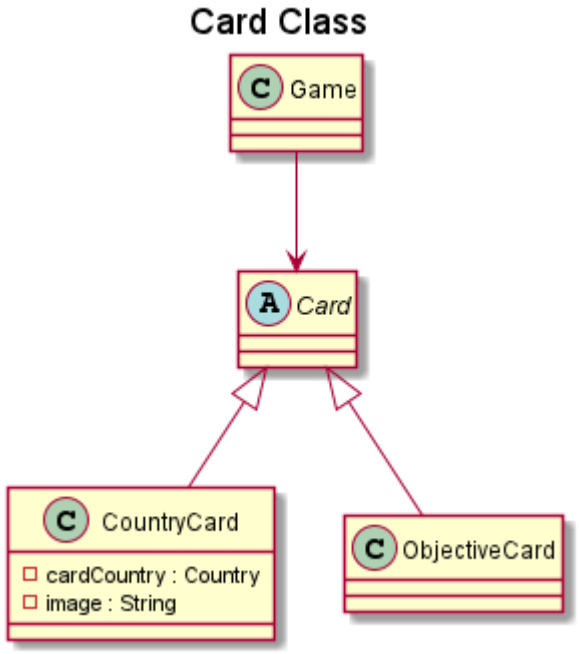
Main Diagram



Round Class




Exceptions
EmptyCountryParameterException
Exception
EmptyContinentParameterException
InvalidNumberOfPlayers
NonExistentContinent
NonExistentCountry
NonExistentPlayer
UnsuccessfulAttack


Card Class
Game
Card
CountryCard
cardCountry : Country
image : String
ObjectiveCard

Informe tercera entrega T.E.G.

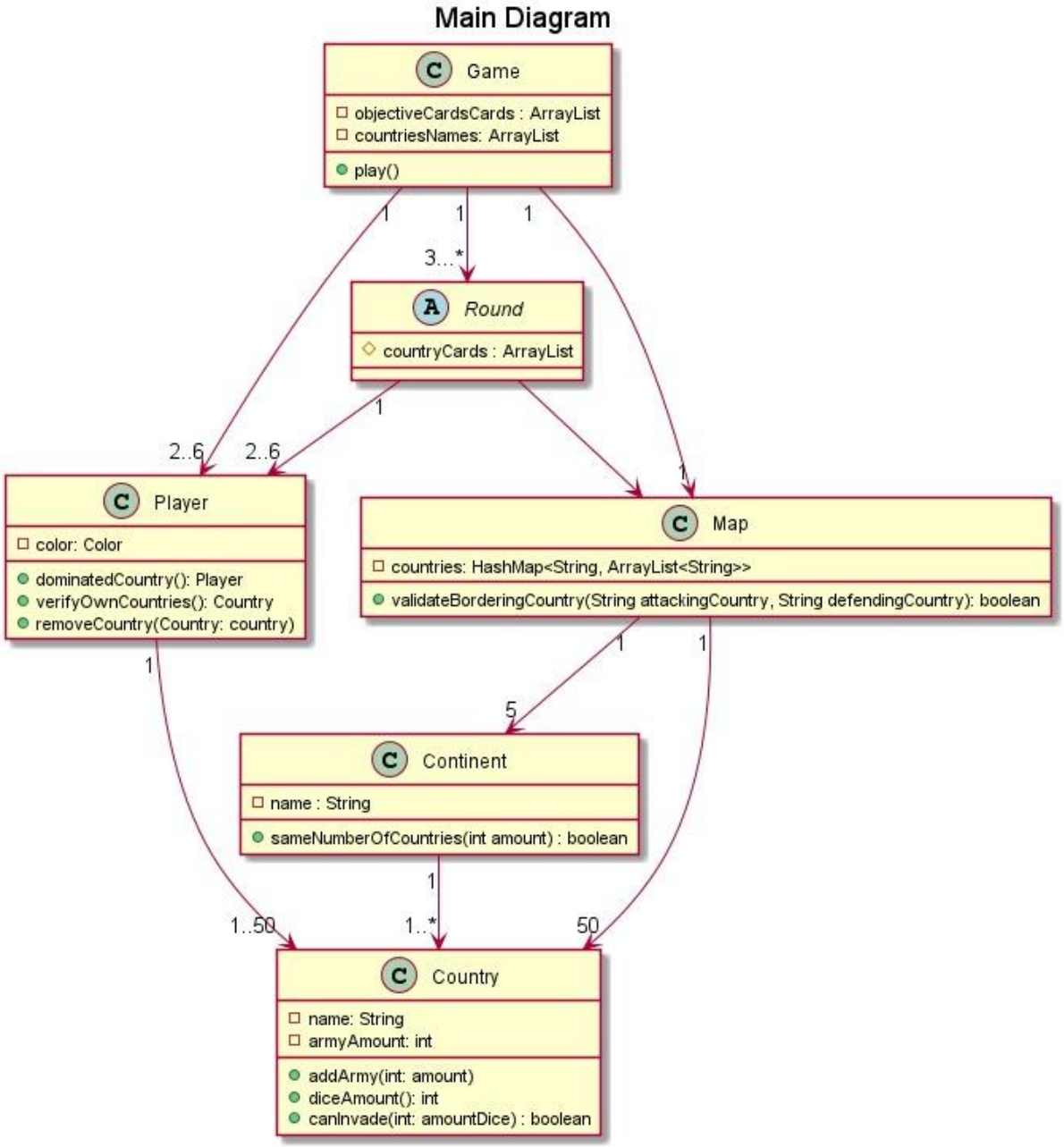
Consigna:

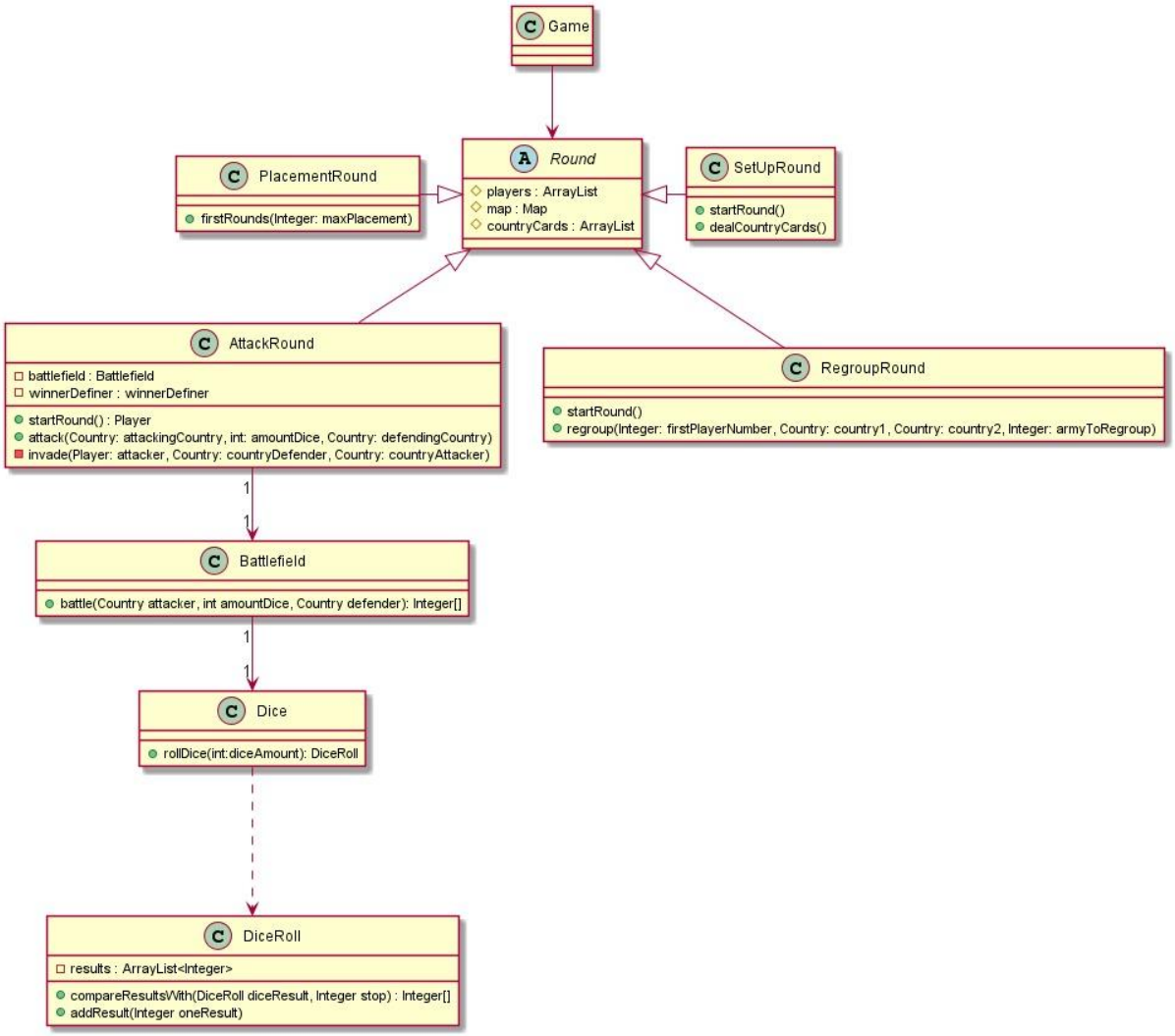
- Modelo del juego terminado
- Interfaz gráfica inicial básica: comienzo del juego y visualización del tablero e interfaz de usuario básica.

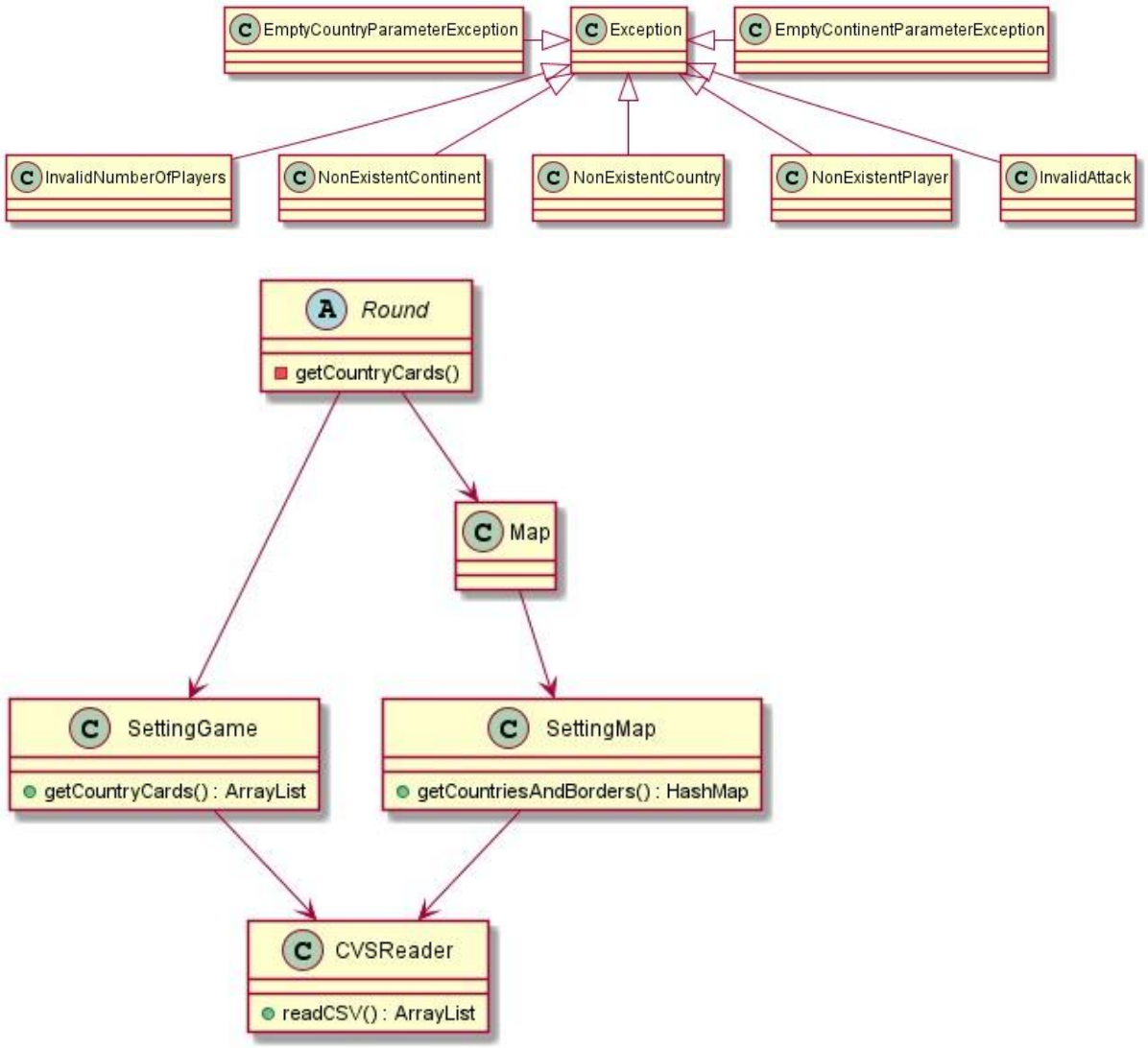
El los cambios que se añadieron en el modelo para cumplir con la consigna otorgada fueron:

- Creación de la clase CSVReader la cual tiene la responsabilidad de leer los archivos correspondientes a los países y continentes del mapa, así como las cartas de países instanciados en la clase Game.
- Se agregaron dos clases nuevas: SettingMap y SettingGame, las cuales se encargan de instanciar en la clase correspondiente lo que devuelve el lector del archivo y dejar a Game y Map en un estado válido de ejecución.
- También se creó la clase WinnerDefiner, cuya tarea es definir quién es el ganador del juego. Se decidió delegar esta tarea a otra clase, ya que a futuro puede cambiar el criterio para que un jugador gane.
- Para el funcionamiento del proyecto se pasó toda la lógica de la clase Game a las rondas de ataque, posicionamiento y reagrupamiento. Y así la única responsabilidad de Game será correr el flujo del juego, y cortarlo en caso de tener un ganador.

Diagramas de clase de la entrega




Game
Round
players : ArrayList
map : Map
countryCards : ArrayList
PlacementRound
firstRounds(Integer: maxPlacement)
SetUpRound
startRound()
dealCountryCards()
AttackRound
battlefield : Battlefield
winnerDefiner : winnerDefiner
startRound() : Player
attack(Country: attackingCountry, int: amountDice, Country: defendingCountry)
invade(Player: attacker, Country: countryDefender, Country: countryAttacker)
RegroupRound
startRound()
regroup(Integer: firstPlayerNumber, Country: country1, Country: country2, Integer: armyToRegroup)
1
1
Battlefield
battle(Country attacker, int amountDice, Country defender): Integer[]
1
1
Dice
rollDice(int:diceAmount): DiceRoll
DiceRoll
results : ArrayList<Integer>
compareResultsWith(DiceRoll diceResult, Integer stop) : Integer[]
addResult(Integer oneResult)


EmptyCountryParameterException
Exception
EmptyContinentParameterException
InvalidNumberOfPlayers
NonExistentContinent
NonExistentCountry
NonExistentPlayer
InvalidAttack
Round
getCountryCards()
Map
SettingGame
getCountryCards() : ArrayList
SettingMap
getCountriesAndBorders() : HashMap
CVSReader
readCSV() : ArrayList

Informe tercera entrega T.E.G.

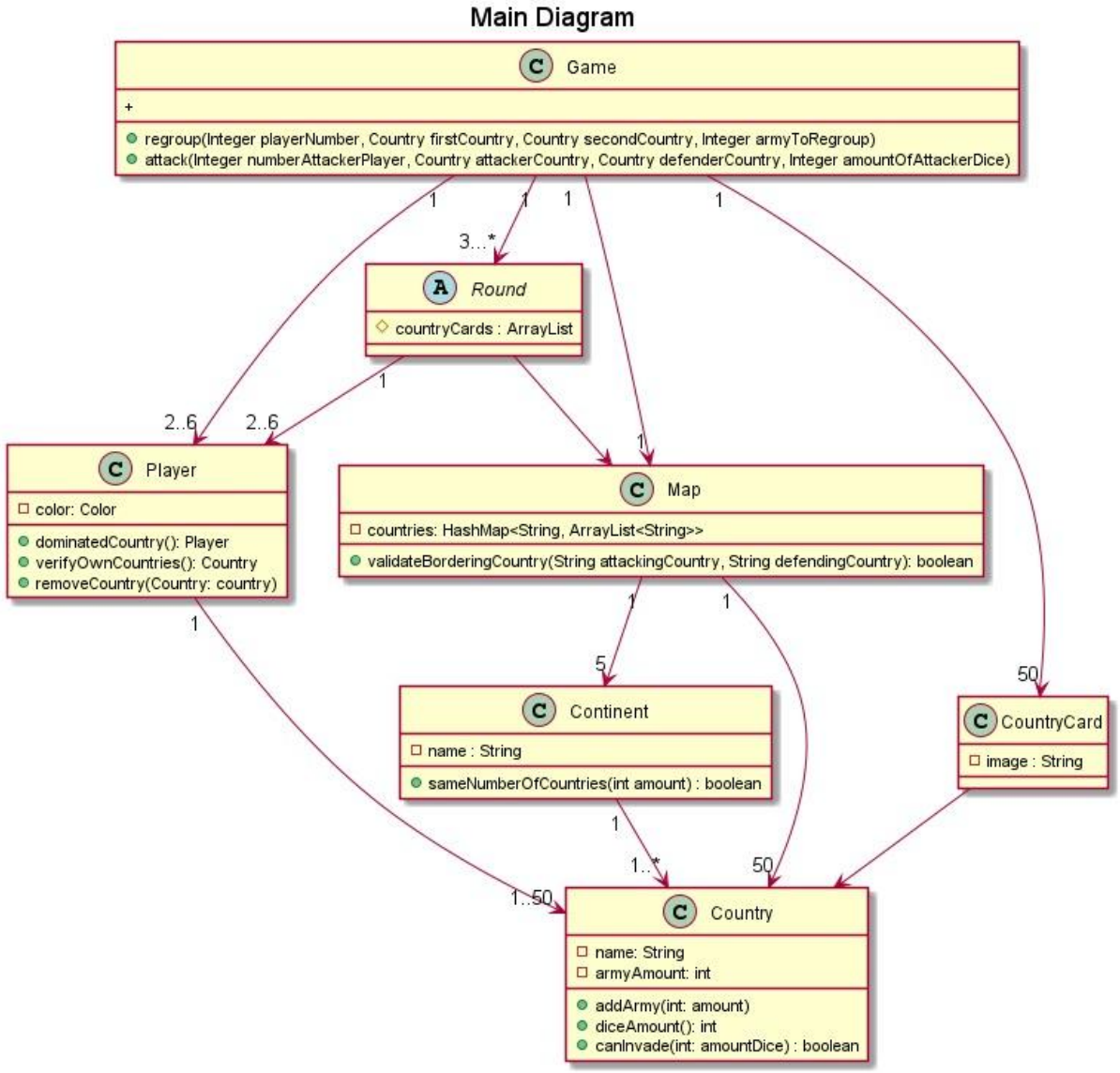
Consigna:

Trabajo Práctico completo funcionando, con interfaz gráfica final, sonidos e informe completo.

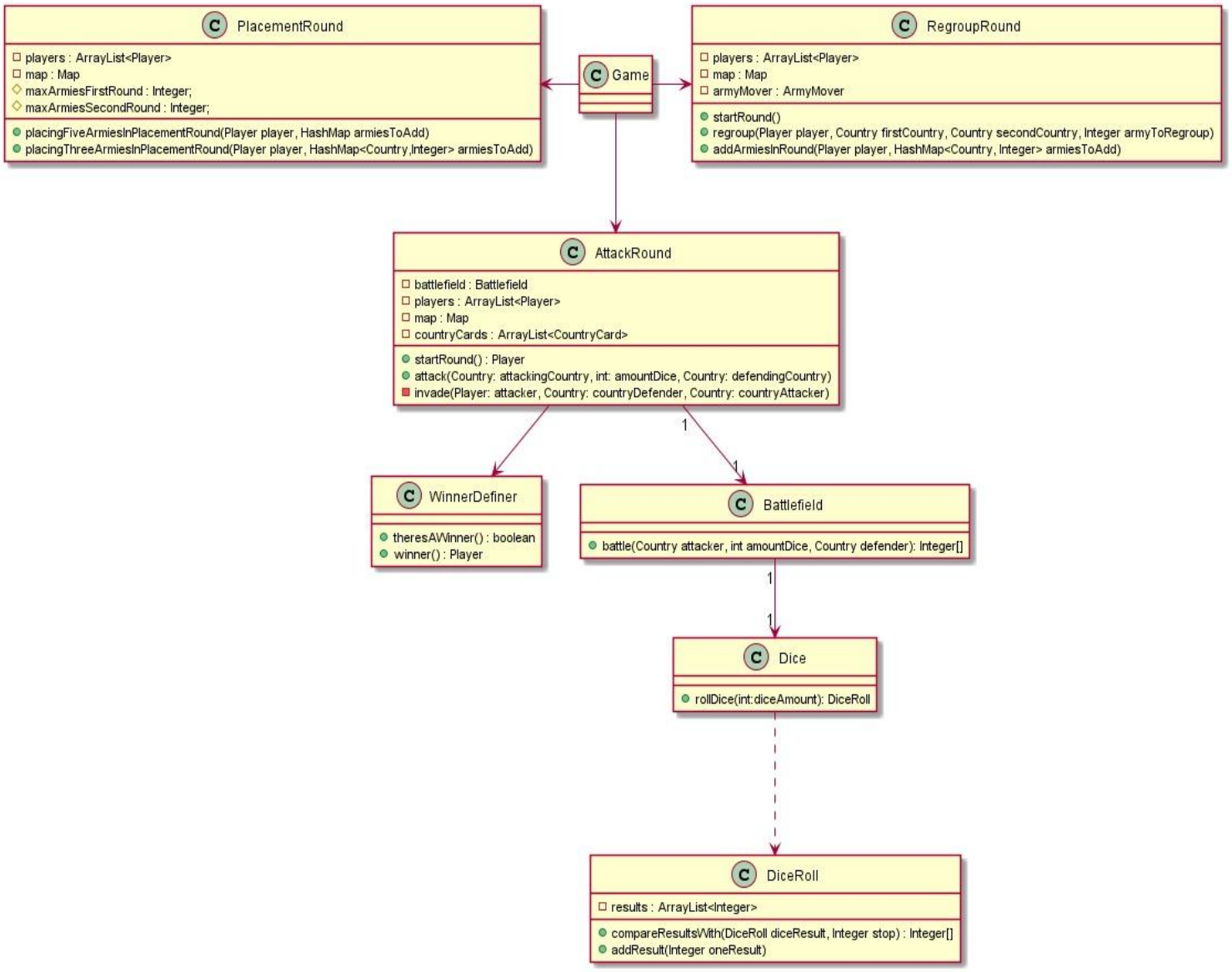
El los cambios que se añadieron en el modelo para cumplir con la consigna otorgada fueron:

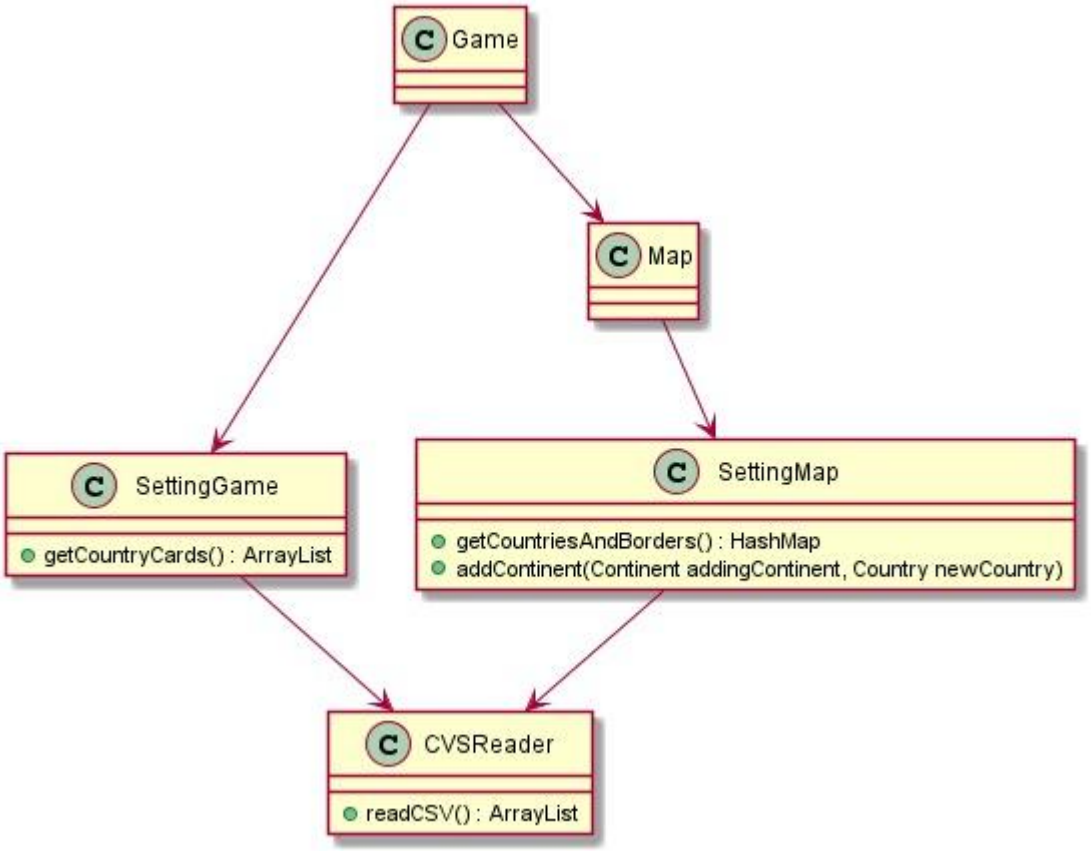
- Se implementó una interfaz gráfica usando Javafx, representando una partida del juego TEG para dos jugadores.
- Se agregaron las clases MockGame, MockAttackRound, MockBattlefield y MockDice para tener una versión controlada del funcionamiento del proyecto. La diferencia entre Game y MockGame es que la última no reparte las cartas de países, dejando este manejo de información al desarrollador de las pruebas. Luego, las demás clases difieren de sus originales en la implementación del ataque, otorgándole la victoria a un único jugador.

Diagramas de clase de la entrega



Round Class


PlacementRound
players : ArrayList<Player>
map : Map
maxArmiesFirstRound : Integer;
maxArmiesSecondRound : Integer;
placingFiveArmiesInPlacementRound(Player player, HashMap armiesToAdd)
placingThreeArmiesInPlacementRound(Player player, HashMap<Country,Integer> armiesToAdd)
Game
RegroupRound
players : ArrayList<Player>
map : Map
armyMover : ArmyMover
startRound()
regroup(Player player, Country firstCountry, Country secondCountry, Integer armyToRegroup)
addArmiesInRound(Player player, HashMap<Country, Integer> armiesToAdd)
AttackRound
battlefield : Battlefield
players : ArrayList<Player>
map : Map
countryCards : ArrayList<CountryCard>
startRound() : Player
attack(Country: attackingCountry, int: amountDice, Country: defendingCountry)
invade(Player: attacker, Country: countryDefender, Country: countryAttacker)
1
1
WinnerDefiner
theresAWinner() : boolean
winner() : Player
Battlefield
battle(Country attacker, int amountDice, Country defender): Integer[]
1
1
Dice
rollDice(int:diceAmount): DiceRoll
DiceRoll
results : ArrayList<Integer>
compareResultsWith(DiceRoll diceResult, Integer stop) : Integer[]
addResult(Integer oneResult)


Game
Map
SettingGame
getCountryCards() : ArrayList
SettingMap
getCountriesAndBorders() : HashMap
addContinent(Continent addingContinent, Country newCountry)
CVSReader
readCSV() : ArrayList

Correcciones entrega 1

- Corregir magic numbers (colores, cantidad de jugadores). ✓
- Parámetros como variables para agregar claridad al código.✓
- Revisar Test 02. Los jugadores tienen un orden.
- No pasar strings por parámetros, utilizar objetos .✓
- En caso de ser sumamente necesario hacer un método de uso exclusivo por las pruebas aclararlo en los comentarios. ✓
- Crear una entidad color cuya responsabilidad sea determinar el color de un jugador.✓
- Para resolver la arbitrariedad de los dados utilizar un Mock Dice ✓
- Aplicar Singleton a Dice. ✓
- Corregir múltiples violaciones del encapsulamiento (Tell, Don't Ask). ✓
- Agregar excepciones sobre comportamientos inesperados. ✓

Dudas

- Como testear las clases mockeadas
- Cómo solucionar String en países
- Mostrar diagramas
- Charlar csv
- Mostrar correcciones

Correcciones entrega 2

- Nombre de la excepción UnsuccessfulAttack no es claro -- InvalidAttack ✓
- Tirada De Dados clase comparadora -- encapsula la comparación y permite extensión del código ✓
- Para Map y Game -- clase que lea un archivo -- otra que convierta lo leído en instancias de las clases correspondientes -- y una clase que sea la fachada de Game y Map, donde se reciben las instancias anteriores ✓

Correcciones entrega 3

Dudas:

- En lugar de validaciones en la interfaz, se usa un Picker o ComboBox mostrando las opciones.
- Borrar jugadores si se quedan sin países o sólo saltear el turno.