RISK

UNIVERSITÀ
DEGLI STUDI
DI PALERMO

Progetto di Metodi Avanzati per la Programmazione
Anno: 2019

Riccardo Parrino **0704026**
Dario Curreri **0705923**

abstract

Realizzazione del famoso gioco da tavola Risiko in linguaggio JAVA, utilizzando, dove possibile ed efficace, i design pattern definiti dalla GoF ed i principi SOLID.
L'utilizzo di queste tecniche ha contribuito ad affrontare e risolvere alcune difficoltà incontrate durante le fasi di progettazione e di scrittura, ottenendo alla fine un codice di qualità, riutilizzabile e modificabile.
Il videogioco realizzato permette ad un giocatore reale di confrontarsi con dei giocatori virtuali attraverso un'interfaccia grafica realizzata utilizzando la libreria SWING di JAVA.

Indice

Descrizione del gioco

Risiko è un gioco da tavolo di strategia che prevede da 3 a 6 giocatori. Lo scopo del gioco è il raggiungimento di un obiettivo predefinito, segreto e diverso per ciascun giocatore, che può consistere nella conquista di un certo numero di territori, nella conquista di due o più continenti o nell'annientamento di un giocatore avversario.
Per maggiori informazioni consultare, in particolare la sezione COME SI GIOCA, il regolamento di gioco a questo link.

Requisiti dell'utente

Panoramica del progetto

Il sistema software deve implementare il gioco da tavolo Risiko, in linguaggio JAVA.
Per questa implementazione è previsto uno ed un solo giocatore reale e da 2 a 5 giocatori virtuali. Questi ultimi dovranno essere animati da un'intelligenza artificiale, che potrà assumere tre diversi stili di gioco: bilanciato, aggressivo e conservativo. Il sistema, una volta avviato, dovrà generare un menu iniziale, in cui è possibile scegliere il nome del giocatore reale, il colore delle armate dello stesso, il numero di partecipanti alla partita e lo stile di gioco di ognuno dei giocatori virtuali. Una volta scelti i partecipanti della partita, il menu presenterà un bottone da cui potrà essere avviata la partita.
Il sistema disporrà di un'interfaccia grafica composta in tre parti:

- il board di gioco
- il pannello di log
- il pannello di gioco, che riporta le informazioni del giocatore reale e le informazioni riguardo il turno attuale.

Il board di gioco dovrà presentare i territori del classico gioco da tavolo Risiko. Ogni territorio dovrà riportare il nome, il numero di tank in esso posizionati e il giocatore che lo possiede. Inoltre i vari territori devono poter essere selezionabili, per poter svolgere le varie fasi di gioco.
Il pannello, riportante le informazioni sul giocatore reale, dovrà presentare informazioni circa il nome, il colore da esso scelto, il numero di territori da esso posseduti, le carte che possiede, l'obiettivo che dovrà perseguire durante la partita, la fase di gioco in cui esso si trova, il giocatore che attualmente detiene il turno e un bottone che gli permetterà di passare alla fase successiva del gioco.
Il pannello di log riporterà le varie azioni intraprese dai giocatori come il posizionamento di un'armata, la conquista di un nuovo territorio da parte di un giocatore in uno scontro, lo spostamento di un certo numero di armate da un territorio ad un altro ed il numero di armate ancora da posizionare.

L’inizio della partita sarà preceduto da una fase di preparazione. Appena la fase di preparazione sarà dunque conclusa, il gioco comincerà.
L’inizio del gioco sarà annunciato tramite un messaggio mostrato nella GUI.
Il turno di gioco, di un qualsiasi giocatore, sarà diviso in tre fasi, cioè: Fase di Rinforzo, Fase di Attacco e Fase di Spostamento.
Nello stadio iniziale, la fase di preparazione, verranno distribuiti i territori ai vari giocatori, la carta dell’obiettivo e le armate che egli stesso dovrà posizionare. Le armate dovranno essere posizionate a tre a tre, fino ad esaurimento.
Nella fase di rinforzo, il sistema dovrà attribuire un numero di armate al giocatore corrente in funzione del numero di territori posseduti ed in funzione degli eventuali continenti posseduti, quindi in accordo con le classiche regole del Risiko. Il numero di armate da posizionare verrà mostrato nella GUI, tramite un messaggio.
In questa fase, il sistema deve dare la possibilità di scambiare un tris di carte dei territori con il corrispondente numero di armate.
Inoltre il sistema non dovrà permettere la terminazione di questa fase, se il giocatore corrente non ha posizionato tutte le armate a disposizione.
La fase di rinforzo dovrà essere terminata con un click sul bottone “End Reinforcement”, presente nel pannello del giocatore.
Nella fase di attacco deve essere possibile attaccare un territorio occupato da un altro giocatore. Il sistema dovrà implementare le regole del gioco reale, ovvero: il territorio da attaccare deve essere confinante con quello posseduto dal giocatore corrente; il territorio attaccante deve avere almeno due armate in esso posizionate. Il sistema dovrà lanciare un numero di dadi, sia di attacco che di difesa, coerentemente con il numero di armate possedute dai due territori (per semplicità, verranno lanciati il maggior numero di dadi in accordo con le regole descritte). Il sistema dovrà confrontare i dadi estratti:
se il dado di attacco ha un valore strettamente maggiore al dado di difesa allora il territorio di difesa perderà un’armata; altrimenti il territorio attaccante perderà un’armata.
I dadi lanciati dovranno essere mostrati nella GUI. Inoltre dovranno essere notificatii territori coinvolti nell’attacco, i giocatori che possiedono tali territori e le armate che hanno vinto lo scontro. Conclusa la fase di confronto dei dadi, il territorio attaccato passerà in possesso dell’attaccante se tale territorio sarà rimasto senza alcuna armata. In tal caso, il territorio
attaccante dovrà passare sul territorio attaccato un numero di armate almeno pari a quelle che hanno partecipato allo scontro. Tale dato verrà preso in input dalla GUI. Se un giocatore, reale o virtuale, vincerà un territorio, allora sarà mostrato un messaggio nell’interfaccia. La fase di attacco continuerà finché il giocatore vorrà attaccare territori posseduti da altri giocatori o finché questo non raggiungerà il proprio obiettivo. Il giocatore reale può decidere di ripetere lo scontro appena effettuato cliccando sul bottone “Continue” presente nell’interfaccia in cui vengono mostrati i dadi lanciati. La fase di attacco dovrà essere terminata con un click sul bottone “End Attack”, presente nel pannello del giocatore.

Nell'ultima fase di gioco, ovvero la fase di spostamento, il sistema dovrà dare la possibilità di effettuare al più uno spostamento di un numero arbitrario di armate, da un proprio territorio ad uno confinante, comunque da egli stesso posseduto.
Alla fine di questa fase, se il giocatore ha conquistato almeno un territorio nella fase precedente allora questo dovrà pescare una carta dal mazzo delle carte dei territori, se nel mazzo vi è almeno una carta.
Se un giocatore perde tutti i suoi territori allora il sistema dovrà provvedere a rimuovere tale giocatore dalla partita.
Il sistema dovrà chiudere il gioco quando uno dei giocatori raggiungerà il suo obiettivo.
L'inizio di una qualsiasi fase di gioco, di un turno del player reale dovrà essere annunciato da un messaggio mostrato a GUI.
Inoltre, durante il primo turno di gioco del player reale, il sistema fornirà un tutorial per facilitare l'interazione del giocatore reale con l'interfaccia grafica.

Storie utente e Scenari

Storia utente: Posizionamento armate

Bob è il giocatore che attualmente detiene il turno. Dal pannello del giocatore, Bob vede che possiede un numero k di armate da posizionare. Bob clicca su un suo territorio per posizionare un'armata. La fase di rinforzo di Bob non può terminare se non ha ancora posizionato tutte le armate.

Scenario: Posizionamento armate

Ipotesi iniziale: Bob vuole posizionare un'armata in un territorio.
Normale: Bob clicca sul territorio in cui vuole posizionare l'armata. Il sistema dovrà sottrarre un'armata libera da quelle di Bob.
Che cosa può andare male: Bob non ha alcuna armata da posizionare oppure Bob clicca su un territorio che non possiede.
Stato del sistema al completamento dell'operazione: se l'operazione richiesta è lecita allora viene rimossa un'armata dalle armate dell'utente, altrimenti non viene effettuata alcuna operazione.

Storia utente: Scambio di un tris

Bob è il giocatore che attualmente detiene il turno. Dal bottone "cards", Bob sa di essere in possesso di almeno tre carte dei territori, ottenute dalla conquista dei territori degli avversari. Bob decide di scambiarle con il corrispondente numero di armate.
Bob spunta le carte che vuole scambiare e se queste corrispondono ad un tris lecito allora il sistema accrediterà a Bob il corretto numero di tris, sottraendo dalle sue carte quelle appena scambiate.

Scenario: Scambio di un tris

Ipotesi iniziale: Bob vuole scambiare un tris di carte dei territori da lui posseduto con il corrispondente numero di armate.
Normale: Bob clicca sul bottone "cards" presente nel pannello del giocatore. Mette la spunta sulle tre carte che vuole scambiare, quindi clicca sul bottone "Exchange Tris" dell'interfaccia. Se il tris è lecito allora verranno attribuite le armate a Bob.
Che cosa può andare male: Il tris non è lecito. In tal caso non viene reso cliccabile il bottone "Exchange Tris".

Storia utente: Attacco di un territorio nemico

Bob è il giocatore che attualmente detiene il turno.
Bob possiede un territorio che ha un numero di armate strettamente maggiore di uno.
Quest'ultimo territorio è confinante con un territorio nemico.
Bob decide di attaccare questo territorio, selezionando i due territori coinvolti.
Il sistema lancia automaticamente il maggior numero di dadi possibile.
Il sistema visualizza i dadi lanciati in una finestra di messaggio.
Se Bob annienta tutte le armate del territorio attaccato allora riceve il territorio, su cui, dovrà spostare un numero di armate almeno pari a quelle coinvolte nello scontro. Il numero di armate da spostare verrà preso in input dalla GUI. Bob può decidere di attaccare un altro territorio, ripetere lo scontro con gli stessi territori oppure di terminare la fase di attacco.

Scenario: Attacco di un territorio nemico

Ipotesi iniziale: Bob vuole attaccare un territorio nemico.
Normale: Bob clicca sul proprio territorio da cui vuole attaccare, quindi clicca sul territorio da attaccare. Vengono lanciati, automaticamente dal sistema, i dadi. Vengono eliminate le armate di Bob e le armate del territorio attaccato, in base ai risultati del lancio dei dadi.
Se il territorio attaccato non possiede più alcuna armata allora diventa di proprietà di Bob.
Che cosa può andare male: il territorio da cui si attacca non è di Bob, il territorio da cui si attacca possiede una sola armata, oppure il territorio da attaccare non è confinante a quello da cui si attacca.

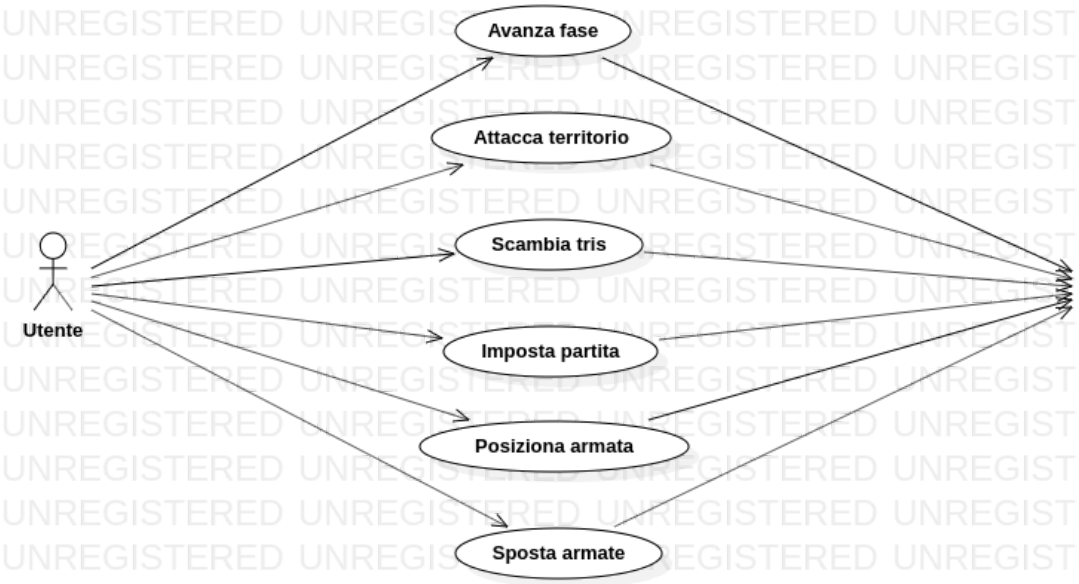
Storia Utente: Spostamento di un certo numero di armate

Bob ha appena terminato la fase di attacco.
Bob vuole di spostare un certo numero di armate da un territorio, da lui posseduto, ad uno confinante. Allora sceglie il territorio di destinazione e immette in input il numero di armate da spostare. Il sistema valuterà se tale operazione è lecita ed, in tal caso, procederà ad eseguire l'operazione richiesta.

Scenario: Spostamento di un certo numero di armate

Ipotesi iniziale: dopo la fase di attacco Bob vuole spostare un certo numero di armate da un suo territorio ad un altro suo territorio confinante.
Normale: Bob clicca il territorio da cui spostare le armate. Quindi, Bob clicca sul territorio di destinazione ed immette in input il numero di armate che vuole spostare.
Il sistema svolgerà le operazioni necessarie.
Che cosa può andare male: Bob clicca un territorio che non è in suo possesso, Bob clicca un territorio che non è confinante a quello inizialmente cliccato oppure il numero di armate da spostare è troppo grande.

Diagrammi dei Casi d'uso



1. Imposta partita

Funzione:

Scelta del numero di giocatori, del colore, del nome del giocatore reale e dello stile di gioco dei giocatori virtuali.

Descrizione:

Tramite un'interfaccia grafica si permette all'utente di scegliere il proprio colore ed il proprio nome durante la partita. Il nome verrà preso input tramite una barra di testo ed il colore tramite un menù a tendina che esibirà tutti i possibili colori: Giallo, Rosso, Verde, Blu, Magenta, Nero. Allo stesso modo sarà possibile scegliere il numero di giocatori virtuali della partita. Ad una partita di Risiko devono partecipare almeno tre giocatori, ed un massimo di sei giocatori.

Passi:

1. Scrivere il proprio nome nella barra di input
2. Scegliere il colore delle proprie armate;
3. Scegliere il numero di giocatori;
4. Scegliere le strategie di gioco per ogni player virtuale;
5. Fare click sul bottone "Inizia Partita".

2. Posiziona armata

Funzione:

L’utente sceglie di posizionare un’armata in un certo territorio.

Descrizione:

L’utente ha un certo numero di armate da posizionare. Questa funzione implementa questa azione.

Input:

Territorio in cui posizionare l’armata. Il territorio viene selezionato tramite un click sull’interfaccia grafica.

Azione:

Il giocatore seleziona il territorio. Se il numero di armate da posizionare è maggiore di 0 ed il territorio selezionato appartiene al giocatore allora il numero di armate da posizionare viene diminuito di 1 e il numero di armate posizionate nel territorio selezionato viene aumentato di 1.

Precondizioni:

Il numero di armate da posizionare deve essere maggiore di 0.

Postcondizioni:

Il numero di armate da posizionare viene diminuito di 1. Il numero di armate posizionate nel territorio selezionato viene aumentato di 1.

3. **Avanza fase**

Funzione:

L'utente passa alla prossima fase di gioco. Se la fase corrente è l'ultima allora passa il turno al prossimo giocatore.

Descrizione:

L'utente decide di passare alla prossima fase.

Input:

Click su bottone "End Stage" dell'interfaccia grafica.

Precondizione:

Devono essere svolte tutte le azioni che sono richieste per concludere la fase corre. Tali condizioni possono variare da fase a fase.

4. **Scambia un tris di carte**

Funzione:

Scambia un tris di carte dei territori possedute da un giocatore con il corrispondente numero di armate da dover posizionare.

Input:

Tre carte dei territori.

Passi:

1. Click su bottone "Cards" nel pannello del giocatore;
2. spunta tre carte dei territori, tramite click su interfaccia grafica;
3. click su bottone "Scambia".

Postcondizione:

Numero di armate da posizionare aumentate del numero di bonus corrispondente al tris.

5. Attacca un territorio

Funzione:

Il giocatore corrente attacca un territorio da lui posseduto ad uno confinante.

Input:

Territorio da cui attaccare, territorio da attaccare.

Passi:

1. Click su territorio da cui attaccare;
2. click su territorio da attaccare;
3. click su “Roll” per lanciare i dadi.

Precondizioni:

l territorio da cui attaccare deve appartenere ai territori del giocatore; il territorio da attaccare deve essere confinante al territorio da cui si attacca; il territorio da cui si attacca deve avere un numero di armate strettamente maggiore di uno;

Postcondizione:

Rimuovere le armate che sono state perse nello scontro, ovvero le armate rimosse sia dal territorio attaccante che dal territorio attaccato. Aggiornare quindi i dati del territorio, ovvero il numero di armate in esso contenute. Se il numero di armate nel territorio attaccato è pari a zero allora il sistema dovrà attribuire al giocatore attaccante il territorio attaccato. Aggiungere il territorio nella lista dei territori posseduti dal giocatore attaccante. Verificare se il giocatore attaccante ha conquistato tutti i territori di un continente, ed aggiungerlo nella lista dei continenti posseduti. Verificare se il giocatore attaccato ha perso un continente, avendo perso un territorio. Cambiare colore del territorio attaccato, con il colore del giocatore attaccante.

Altre attività:

Nel frame mostrante i dadi lanciati esiste un bottone “Continue”. Cliccando su tale bottone è possibile ripetere l’attacco con gli stessi territori coinvolti. Il numero di dadi che verranno lanciati sarà sempre coerente con le regole del Risiko. L’attacco terminerà secondo le regole descritte precedentemente.

6. Sposta armate

Funzione:

Durante la fase di spostamento permette di spostare un certo numero di armate, da un proprio territorio ad un altro.

Input:

Territorio di partenza, territorio di arrivo, numero di armate.

Precondizione:

Devono essere svolte tutte le azioni che sono richieste per concludere la fase corre. Tali condizioni possono variare da fase a fase.

Passi:

1. Click su territorio di partenza
2. Click su territorio di destinazione
3. Slider sul numero di armate da spostare
4. Click sul bottone “Move tanks” della GUI.

Requisiti Funzionali e Non Funzionali

Elenco dei requisiti funzionali:

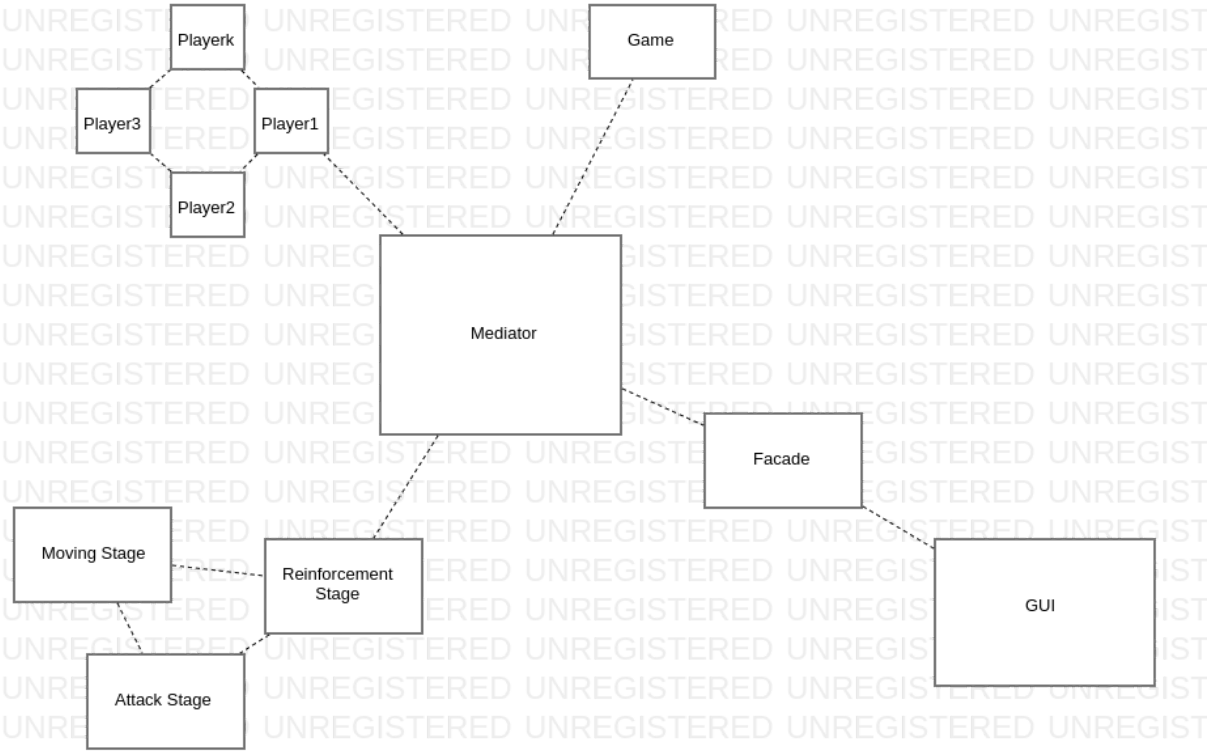
1. Scegliere il nome del giocatore
2. Scegliere il colore del giocatore
3. Scegliere il numero di giocatori
4. Disporre di una GUI per immettere l'input
5. Visualizzare un pannello con i dati del giocatore reale
6. Visualizzare un pannello di log
7. Visualizzare una board in cui sono presenti tutti i territori di gioco
8. Visualizzare una spiegazione del funzionamento della fase di gioco corrente
9. Visualizzare un messaggio che specifichi l'inizio di una fase di gioco
10. Visualizzare il numero di armate da posizionare
11. Visualizzare il turno e la fase di gioco in cui ci si trova
12. Ogni territorio di gioco deve essere cliccabile
13. Ogni territorio deve essere colorato del colore scelto dal giocatore che lo possiede
14. Visualizzare il numero di territori da esso posseduti
15. Visualizzare le carte dei territori posseduti dal giocatore
16. Visualizzare la carta degli obiettivi posseduta da un giocatore
17. Visualizzare il territorio vinto da un player virtuale
18. Posizionare un'armata in un proprio territorio.
19. Scambiare un tris di carte dei territori con il corrispondente numero di armate
20. Attaccare un territorio avversario, da uno posseduto
21. Ogni giocatore deve avere un suo mazzo di carte di territori
22. Lanciare i dadi
23. Visualizzare i dadi lanciati, i territori e i giocatori coinvolti nello scontro
24. Spostare un certo numero di armate da un territorio posseduto ad un altro ad esso confinante
25. Rimuovere un certo numero di armate da un territorio
26. Attribuire ad un giocatore una carta dei territorio
27. Cambiare colore di un certo territorio
28. Cambiare il numero di armate possedute da un certo giocatore
29. Cambiare il numero di territori posseduti da un certo giocatore
30. Concludere una partita

Elenco dei requisiti non funzionali:

1. Ogni giocatore deve avere un nome
2. Ogni giocatore deve avere un colore
3. Ogni giocatore deve avere un goal che può raggiungere
4. Ogni giocatore deve posizionare tutte le armate in suo possesso
5. Ogni giocatore può posizionare un'armata soltanto in un suo territorio
6. Ogni giocatore può avere al più sette carte dei territori
7. Il valore in armate di un tris di carte dei territori deve essere in linea con le regole del Risiko classico
8. Un giocatore può attaccare da un proprio territorio ad uno ad esso confinante
9. Il territorio da cui un giocatore attacca deve avere almeno due armate
10. Un giocatore attacca con il maggior numero di armate possibili
11. Un giocatore vince quando raggiunge il proprio obiettivo
12. Un giocatore viene eliminato quando ha perso tutti i suoi territori
13. Il gioco inizia con una fase preliminare in cui vengono distribuiti territori e armate
14. Nella fase preliminare le armate dovranno essere posizionate a tre a tre, fino ad esaurimento
15. Ogni turno ha tre fasi di gioco: Fase di rinforzo, Fase di attacco, Fase di spostamento
16. La spiegazione del funzionamento di una fase di gioco deve avvenire soltanto nel primo turno di gioco e soltanto per il player reale
17. Cambiare la label del bottone di fine stage, nel player panel

Tutti i requisiti espressi su un giocatore reale si applicano tali e quali ai giocatori virtuali.

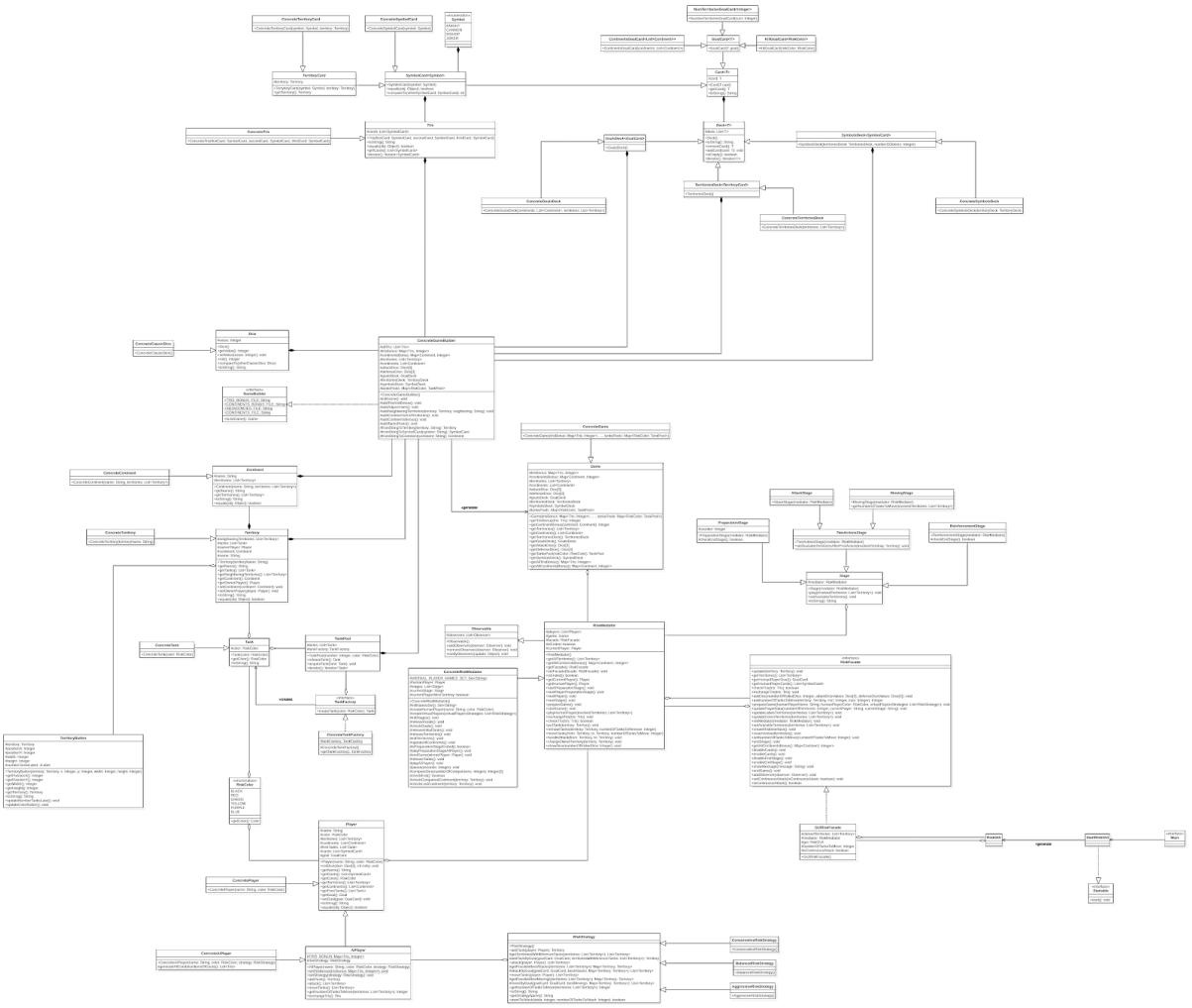
Panoramica strutturale del sistema



Il sistema è centralizzato sulla classe Mediator. Questa gestisce il turno dei vari player partecipanti alla partita; gestisce le risorse di gioco, quali carte dei territori, armate, dadi, etc.; alterna le fasi di gioco per un singolo giocatore, da Reinforcement Stage a Moving Stage, e tramite la classe Facade si interfaccia con la GUI, la quale mostra lo stato del gioco e prende in input le azioni del giocatore reale.

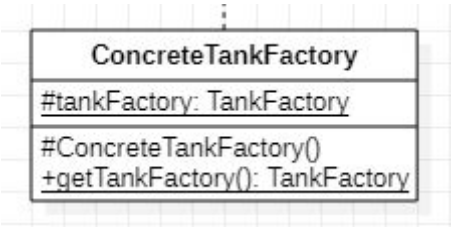
Requisiti del sistema

Diagramma di classi UML



Descrizione dei design pattern

Singleton

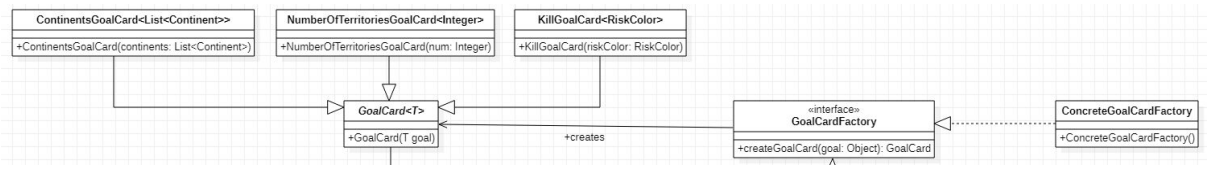


Il pattern Singleton viene utilizzato per assicurare che solo una singola istanza di una classe possa essere creata. Ciò viene realizzato:

- ponendo tutti i costruttori private o protected
- inserendo tra gli attributi della classe il riferimento all'unica istanza della classe
- fornendo un metodo public e static che provveda a restituire l'unica istanza della classe

Nell'esempio sopra riportato, l'unica istanza della classe `ConcreteTankFactory`, viene restituita attraverso la seguente chiamata:
`ConcreteTankFactory.getTankFactory()`.

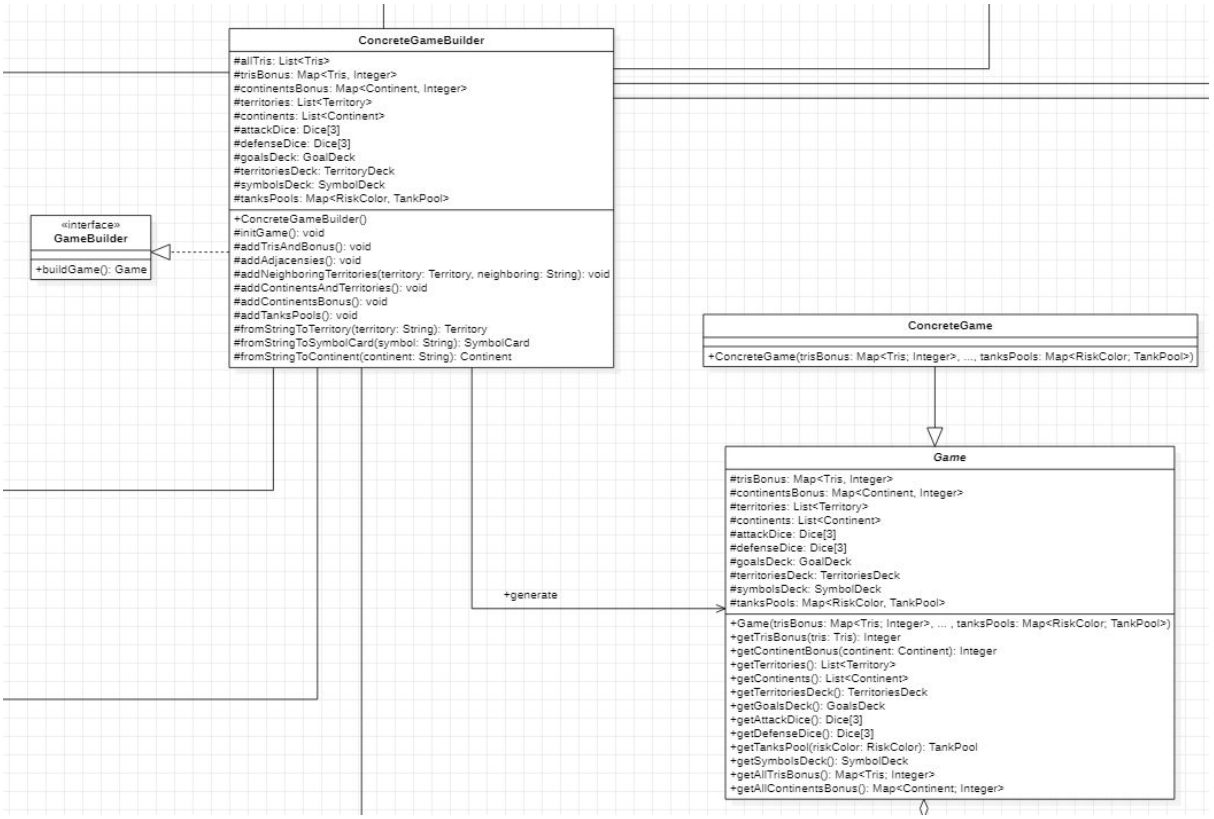
Factory



Il pattern Factory viene utilizzato per istanziare oggetti riferiti attraverso un'interfaccia / una classe astratta comune. Nell'esempio riportato:

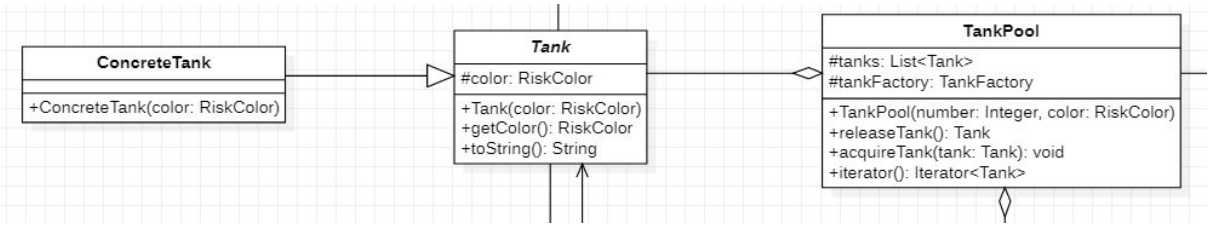
- `GoalCardFactory` è l'interfaccia che deve essere implementata dalle Factory concrete. Contiene il metodo `createGoalCard(Object goal)` che restituisce una carta obiettivo dato un oggetto generico.
- `ConcreteGoalCardFactory` è una concretizzazione dell'interfaccia `GoalCardFactory`, quindi contiene il metodo `createGoalCard(Object goal)`.
- `GoalCard` è la classe astratta comune a tutte le tipologie di carta obiettivo.
- `ContinentsGoalCard`, `NumberOfTerritoriesGoalCard`, `KillGoalCard` sono concretizzazione della classe astratta `GoalCard`.

Builder



Il pattern Builder viene utilizzato per istanziare oggetti appartenenti alla stessa classe ma con differenti strutture. Ogni volta che vi è la necessità di un oggetto con differente struttura rispetto a quello già ottenuto, è sufficiente istanziare un nuovo builder per creare il nuovo oggetto.
Nell'esempio riportato il pattern Builder è stato utilizzato per creare un'istanza della classe complessa `Game`. Tuttavia, poiché l'oggetto `Game` viene definito attraverso alcuni file di configurazione del gioco, non sono presenti nell'interfaccia `GameBuilder` i metodi `addPart / buildPart`, in quanto non vi è certezza di come siano strutturati i file, ma solamente il metodo `buildGame()` che provvede a creare un'istanza di `Game`.
Come è facilmente intuibile `ConcreteGameBuilder` e `ConcreteGame` sono rispettivamente le concretizzazioni di `GameBuilder` e `Game`.

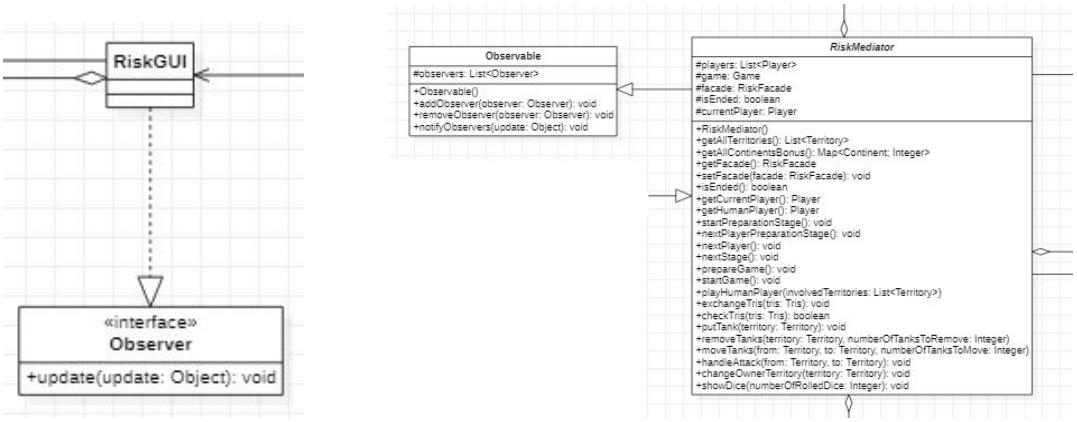
Object Pool



Il pattern Object Pool implementa il meccanismo che permette di riutilizzare e condividere oggetti già creati.
Nell'esempio riportato:

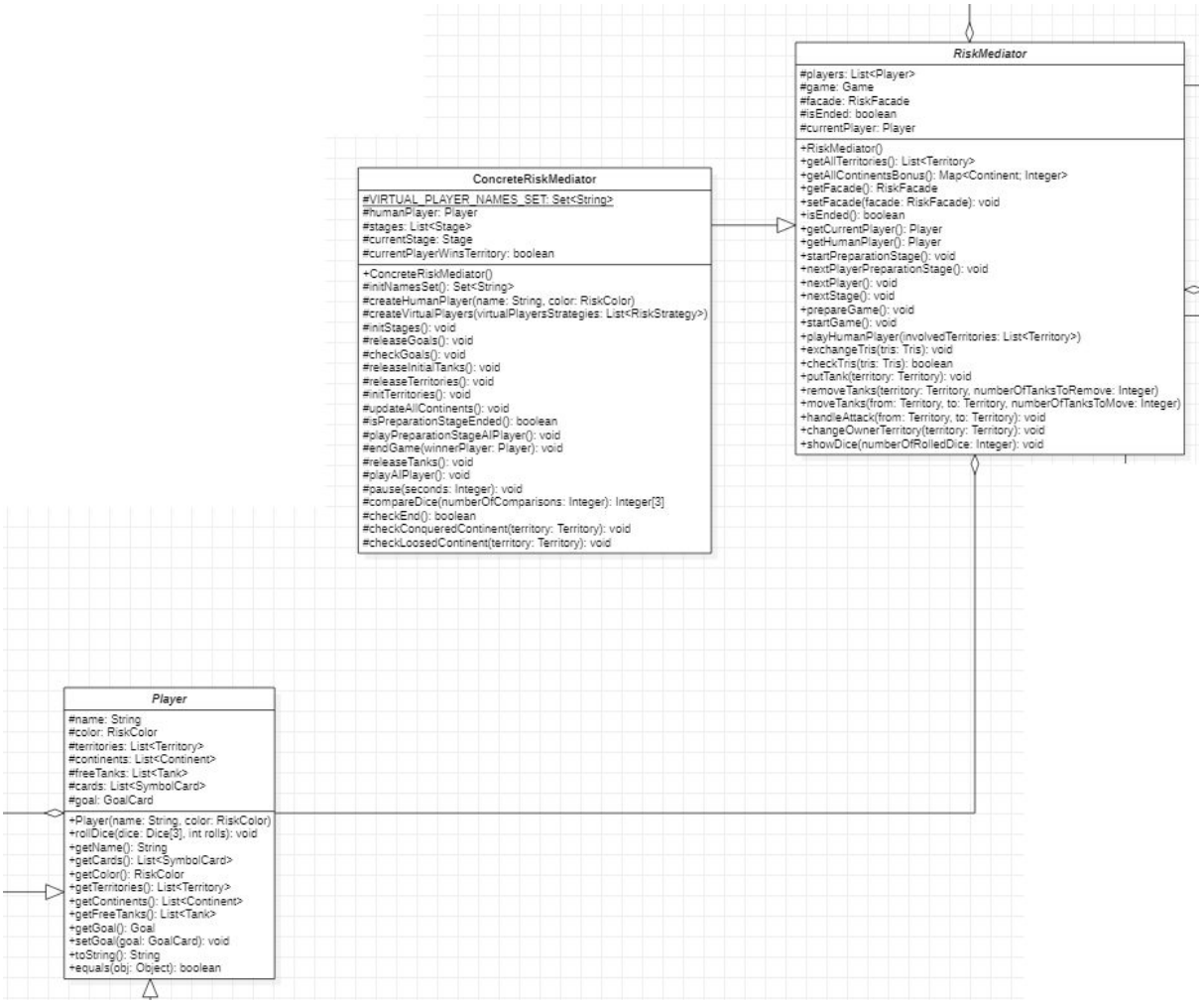
- `TankPool` è la classe che incapsula la logica di gestione dei `Tank`.
  Contiene i metodi `releaseTank` e `acquireTank`.
- `Tank` è la classe astratta che modella il concetto di armata.
- `ConcreteTank` è la concretizzazione della classa astratta `Tank`.

Observer



Il pattern Observer permette di definire una dipendenza uno a molti fra un oggetto osservato e molti oggetti osservanti, in modo tale che se un oggetto cambia il suo stato interno, ciascuno degli oggetti dipendenti da esso viene notificato e aggiornato automaticamente.
Il pattern Observer nasce dall'esigenza di mantenere un alto livello di consistenza fra classi correlate, senza produrre situazioni di forte dipendenza e di accoppiamento elevato. (da link).
Nell'esempio riportato, il `RiskMediator`, ovvero colui che garantisce che il gioco prosegua correttamente, è un oggetto osservabile. In particolare, esso viene osservato dalla GUI, in modo tale che essa aggiorni il `LogPanel` di conseguenza.

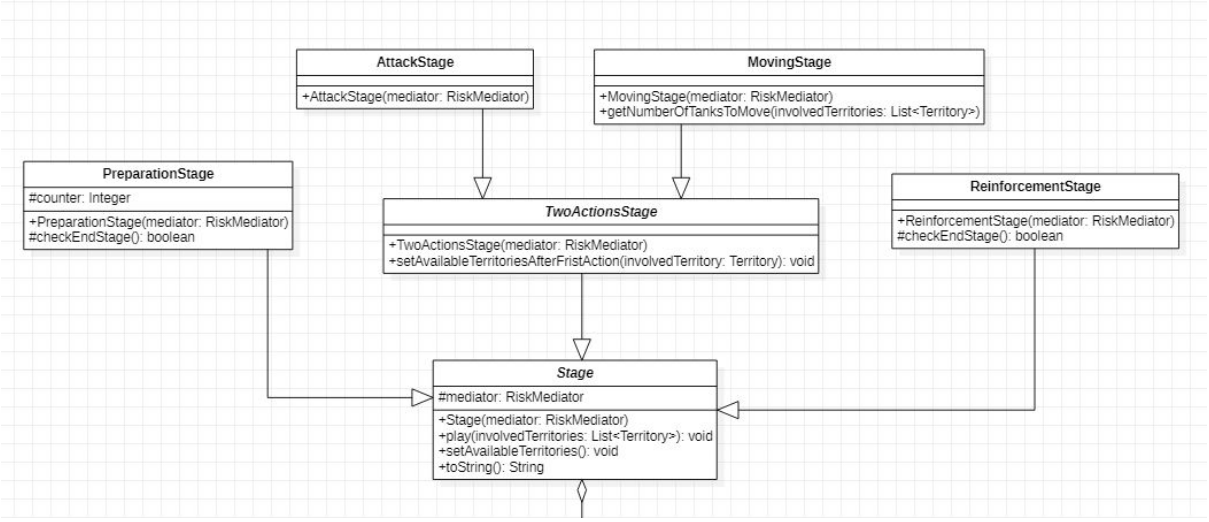
Mediator



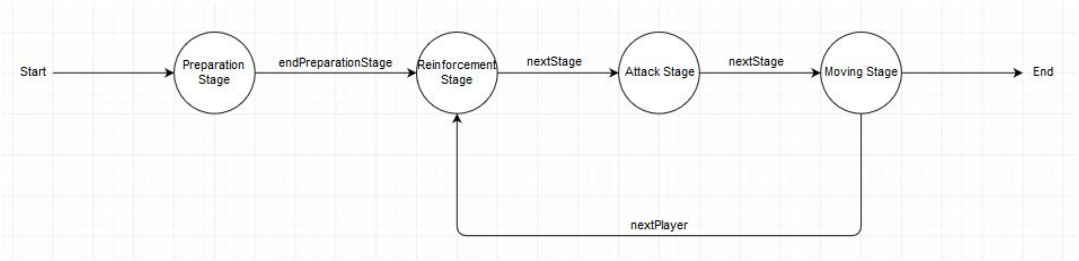
Il pattern mediator definisce un oggetto capace di gestire un insieme di oggetti, chiamati colleghi, che devono interagire tra di loro, riducendo le dipendenze che gli oggetti dovrebbero avere tra di loro.
Nell'esempio riportato:

- `RiskMediator` astrae la logica che il `Mediator` di una partita di Risiko dovrebbe implementare per far sì che il gioco proceda correttamente.
- `ConcreteRiskMediator` è la classe che concretizza il concetto di `RiskMediator` e che gestisce l'avanzamento della partita.
- Player è la classe che astrae il concetto di giocatore di Risiko. I colleghi sono delle istanze delle sottoclassi di `Player`: `ConcretePlayer` e `ConcreteAiPlayer`.

State



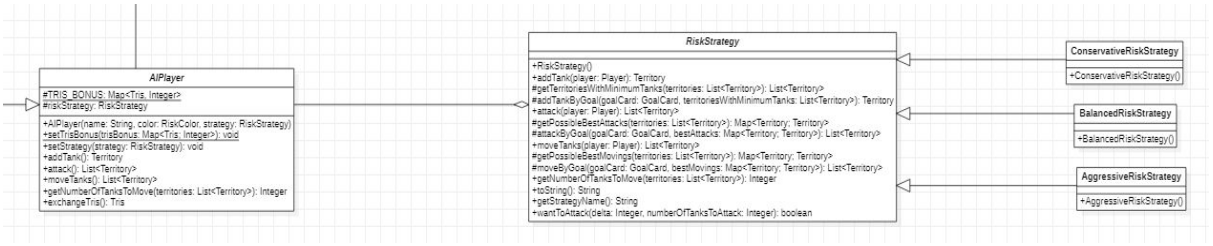
Il pattern State astrae e permette di implementare un automa a stati finiti attraverso un linguaggio di programmazione orientato agli oggetti.
L'automa a stati finiti del gioco Risiko può essere rappresentato come segue:



Nell'esempio riportato:

- Stage astrae il concetto di stato.
- TwoActionStage astrae il concetto di stato in cui è necessario compiere duo o più azioni.
- PreparationStage, ReinforcementStage, AttackStage e Moving Stage sono le classi che concretizzano il concetto di Stage corrispondenti agli stati che una partita di Risiko può assumere (guardare l'automa a stati finiti).

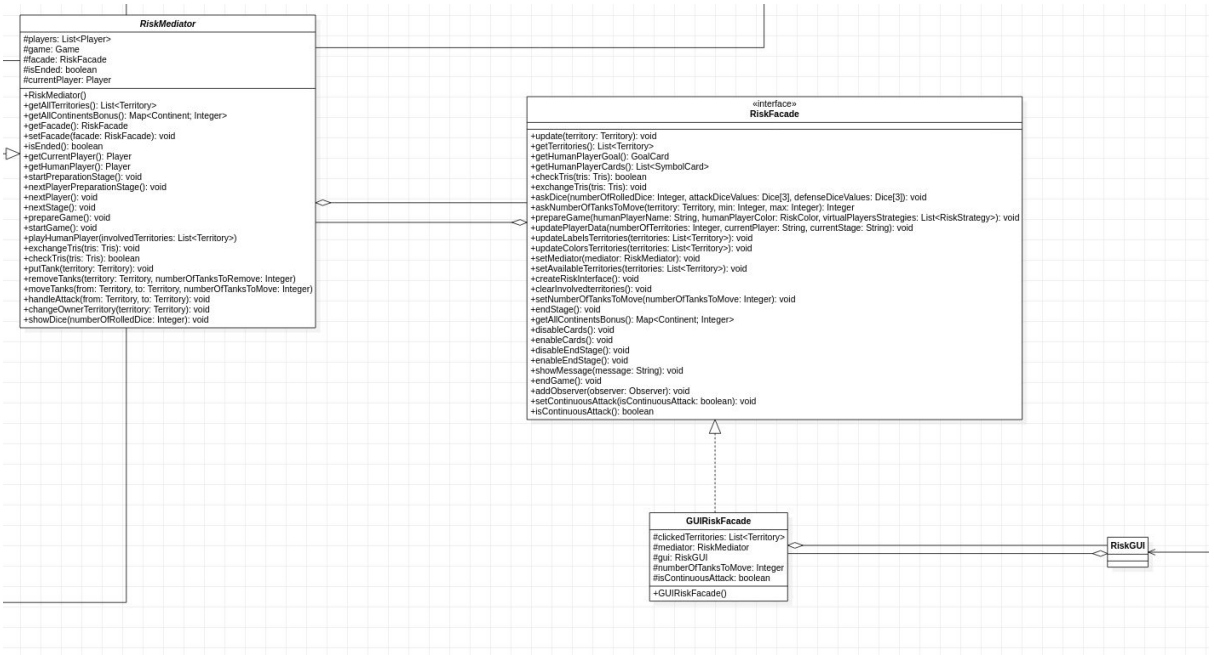
Strategy



Il pattern Strategy definisce una famiglia di algoritmi, incapsulando la logica di ognuno di essi, interscambiabili tra loro.
Nell'implementazione proposta, ogni giocatore virtuale ha una strategia di gioco che può essere più aggressiva o difensiva. Ciò viene implementato aggiungendo un riferimento a `RiskStrategy` nella classe `AIPlayer`.
Quindi, quando un player virtuale dovrà scegliere come giocare, invocherà gli algoritmi propri della strategia scelta.
In particolare:

- `RiskStrategy` definisce i metodi propri di una strategia di gioco e ne implementa altri che sono comuni a tutte le strategie.
- `ConservativeRiskStrategy`, `BalancedRiskStrategy` e `AggressiveRiskStrategy` concretizzano `RiskStrategy`, implementando i metodi richiesti, propri di uno stile di gioco.

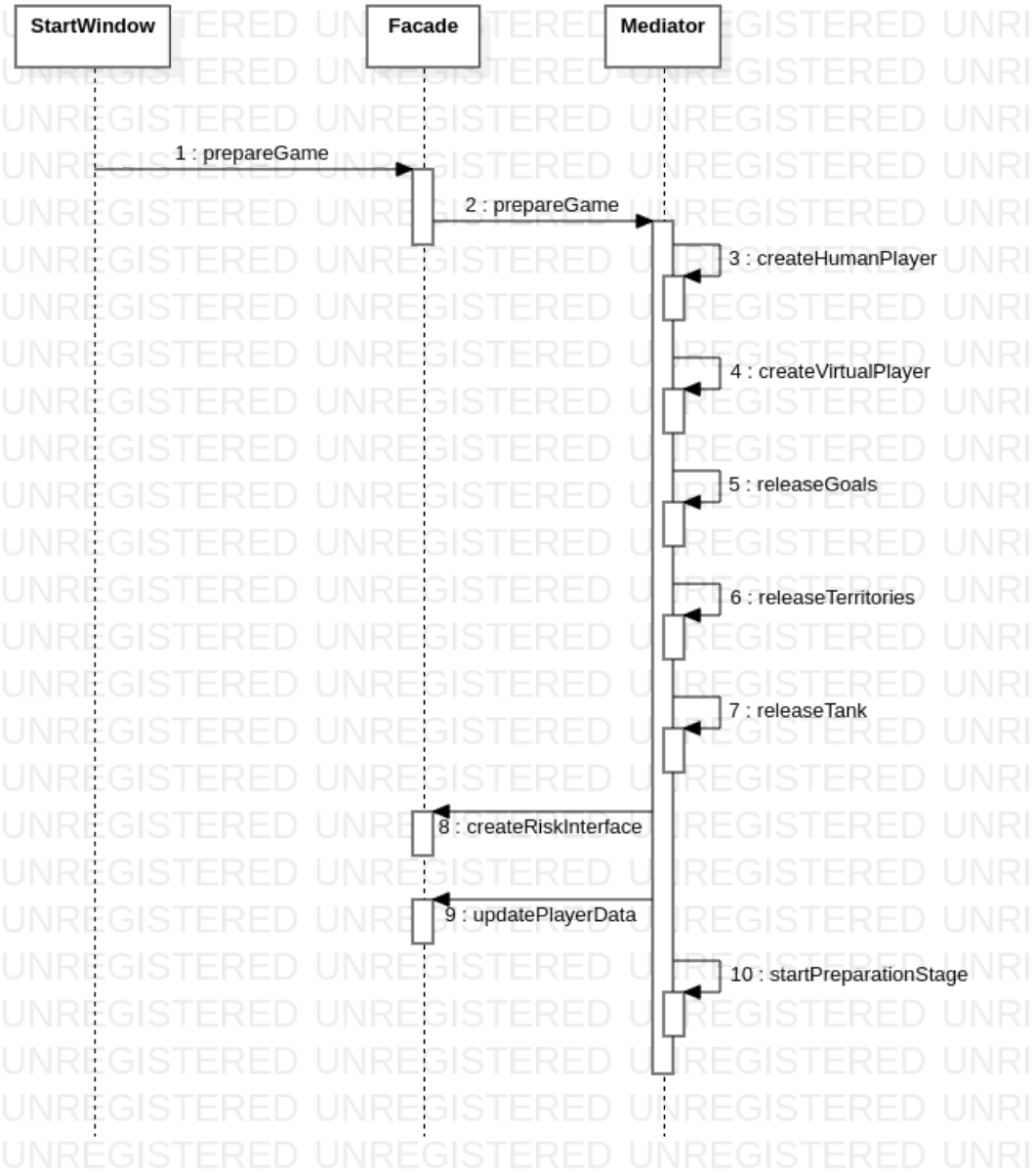
Facade



Il pattern Facade viene utilizzato per fornire un'interfaccia di un sottosistema complesso e quindi per mostrarne le sue funzionalità, senza curarsi della loro complessità. Ciò semplifica l'accesso ai dati e alle funzionalità del sottosistema complesso.
Nell'esempio riportato `RiskFacade` è un'interfaccia contenente tutti i metodi utili per garantire la comunicazione tra `RiskMediator` e l'interfaccia fornita al giocatore reale tramite cui egli compie le proprie scelte. In particolare, in questo caso, `RiskFacadeGUI` agisce da intermediario tra `RiskMediator` e `RiskGUI`.

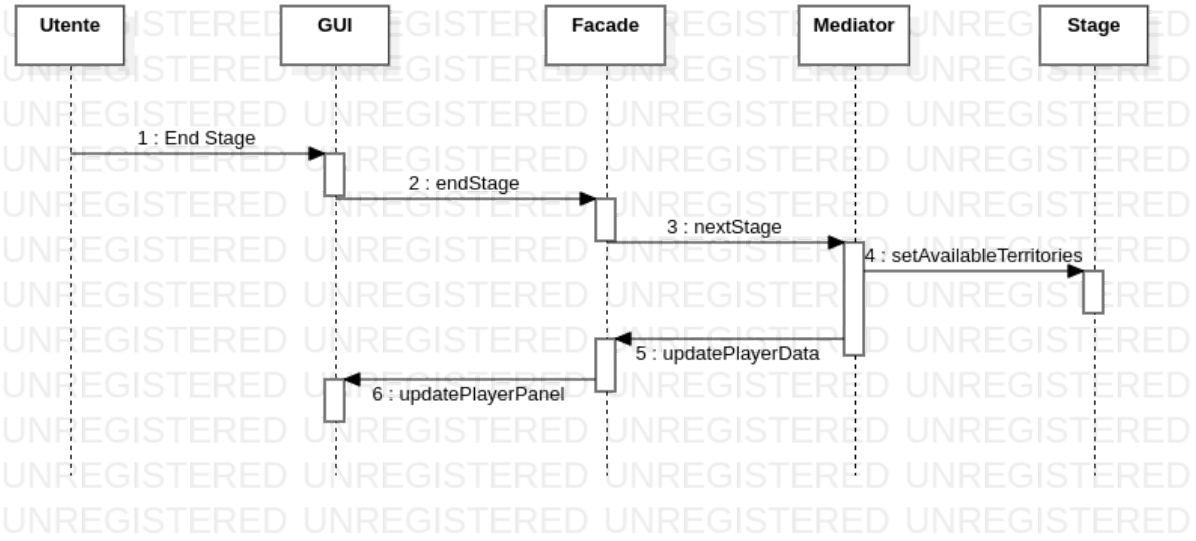
Diagrammi di sequenza

Inizializzazione del gioco



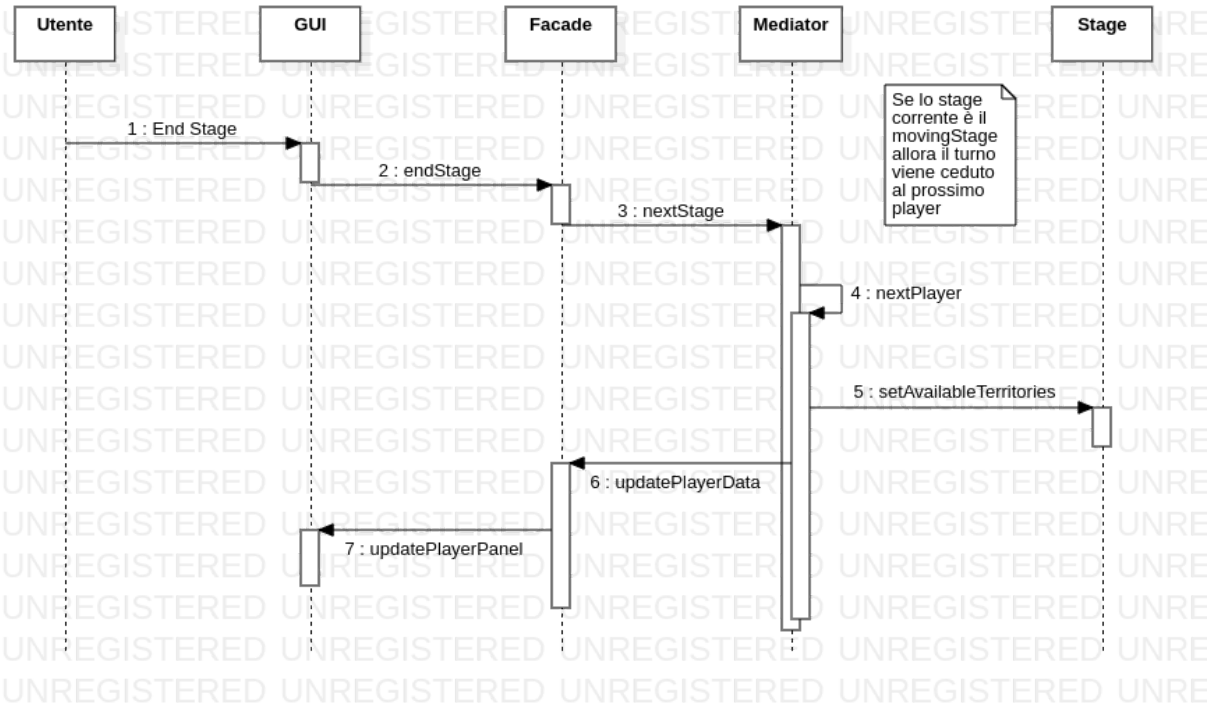
Una volta avviato il gioco, verrà mostrato un frame in cui è possibile settare le modalità di gioco della partita. In particolare, nella “startWindow”, sarà possibile impostare il nome del giocatore, il colore e le strategie di gioco dei player virtuali.
Cliccato sul bottone “start”, la classe Facade chiamerà il metodo “startGame” in Mediator che provvederà a creare il player Reale, i player Virtuali, rilasciare i territori, l’obiettivo e le armate. Successivamente si crearà la board di gioco (GUI) e la si mostra a video. Quindi inizia la fase di preparazione del gioco.

NextStage



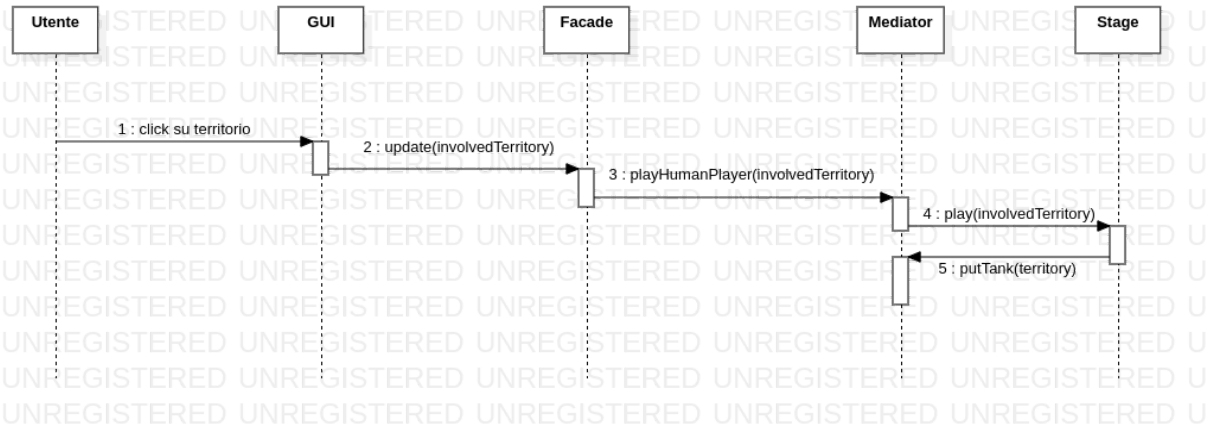
L'utente reale al termine di una fase di gioco cliccherà sul bottone "End Stage".
Il player virtuale, per terminare la fase di gioco corrente, chiamerà il metodo nextStage.
Il metodo "nextStage" setterà la nuova fase di gioco, renderà cliccabili soltanto quei territori leciti per la fase di gioco successiva e aggiornerà la label del "playerPanel".

NextPlayer



Se la fase di gioco corrente è la “MovingStage”, ovvero la fase di spostamento, allora il metodo “nextStage”, una volta invocato, chiamerà il metodo “nextPlayer”. Questo provvederà ad impostare il successivo player, ad impostare la prima fase di gioco ed ad aggiornare la board di gioco.

Reinforcement Stage

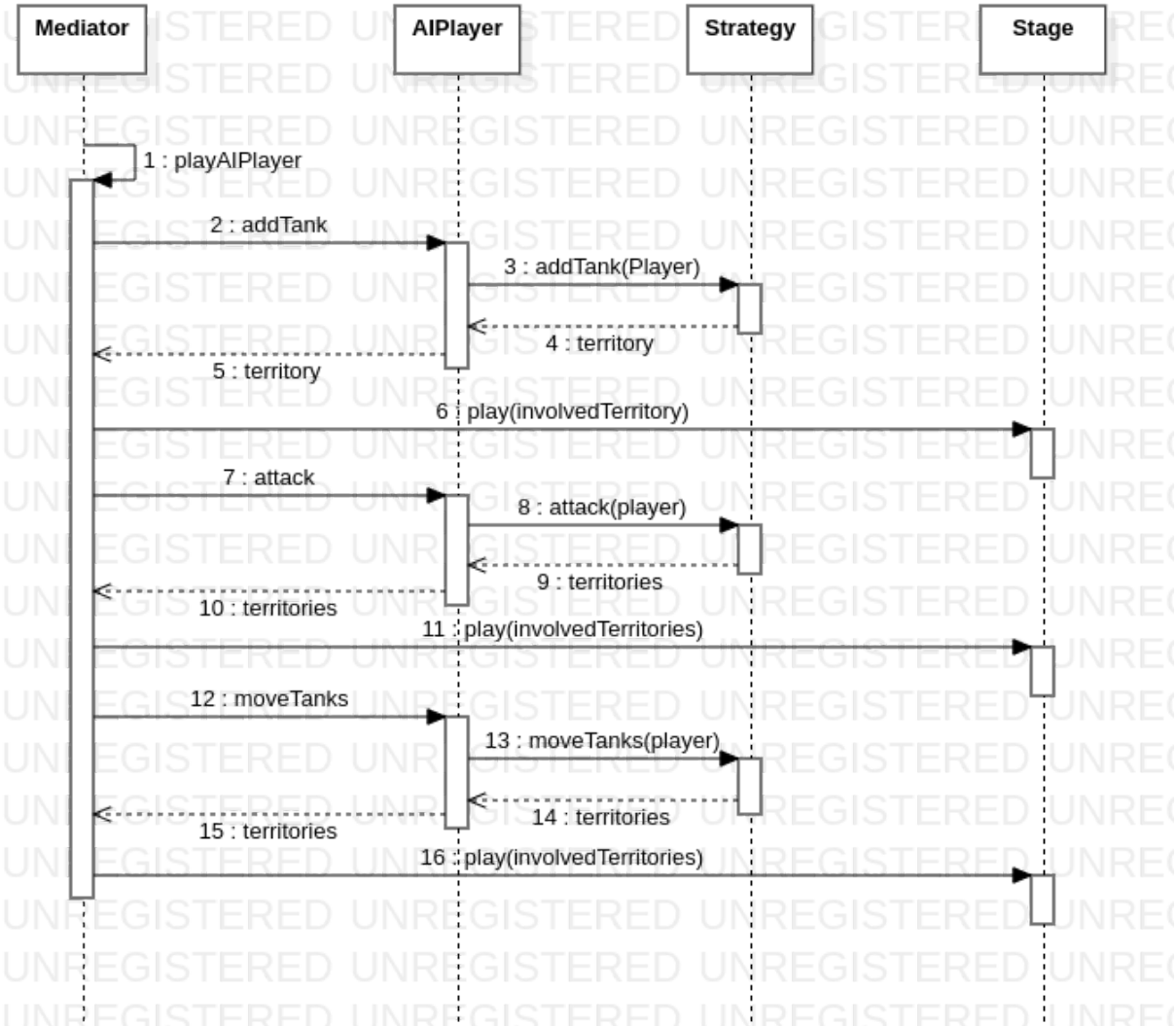


La fase di rinforzo permette di schierare le proprie armate nei propri territori. L'utente reale, cliccando su un territorio, genererà un evento, che scaturirà la chiamata del metodo "update" della classe Facade, che a sua volta chiamerà il metodo "playHumanPlayer" della classe Mediator, che chiamerà il metodo "play" della classe stage (in questo caso MovingStage), che chiamerà il metodo "putTank" di Mediator che metterà l'armata nel territorio, aggiornerà la label del territorio, aggiornerà il "PlayerPanel", e restituirà il controllo alla GUI.

AttackStage

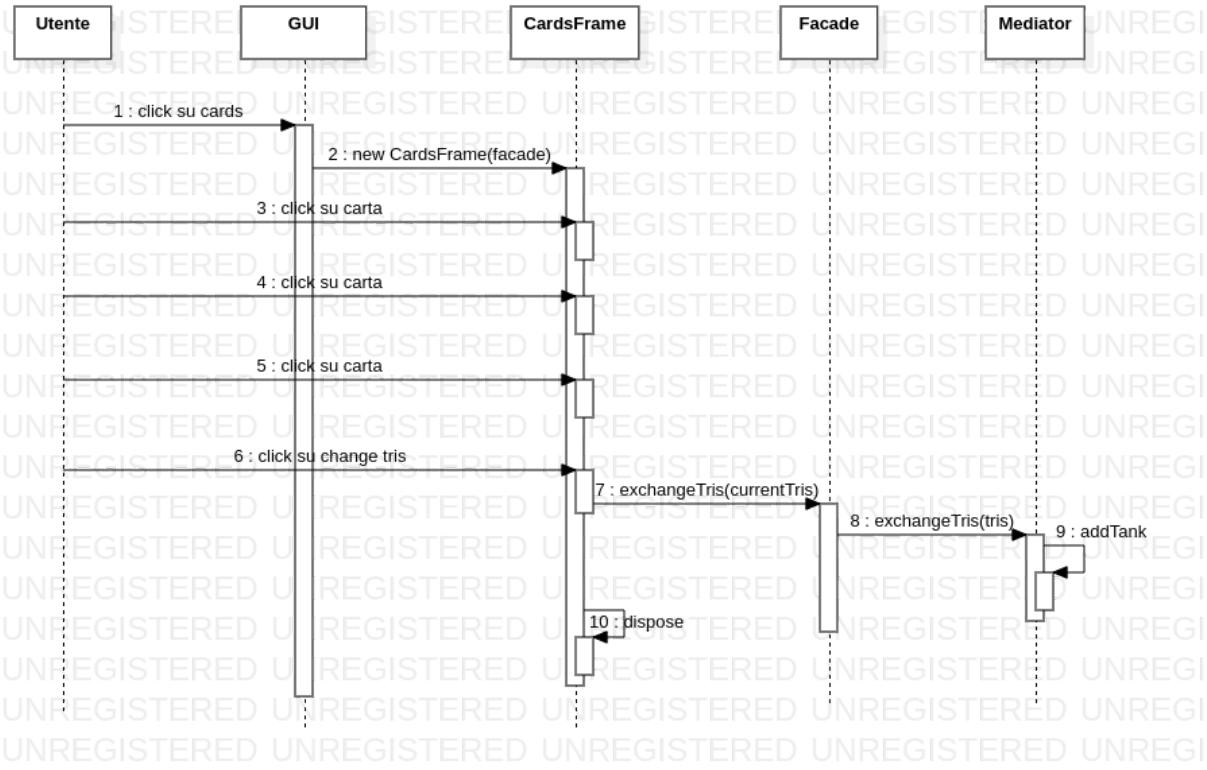
Nella fase di attacco il player umano ha la possibilità di attaccare un territorio nemico, da uno proprio, ad esso confinante. Il player umano cliccherà sul territorio da cui vuole attaccare e la gui “accenderà” di conseguenza soltanto i territori confinanti ad esso, tramite l’implementazione della classe Stage, da parte della classe AttackStage.
Cliccato il secondo territorio, la GUI passerà quest’ultimo alla classe Facade tramite il metodo “update”, che a sua volta chiamerà il metodo “playHumanPlayer”, che in questo caso riceverà i due territori cliccati. Questo chiamerà il metodo “play”, passando entrambi i territori, della classe stage, che chiamerà il metodo “handleAttack” di Mediator che eseguirà tutte le operazioni previste per attuare l’attacco.
Con lo stesso funzionamento viene implementata la fase di spostamento, Moving Stage.

PlayAIPlayer



Il player virtuale svolgerà le azioni del gioco su richiesta del Mediator.
Nella fase di rinforzo il metodo “addTank” chiederà al player virtuale su quale territorio vorrà porre l’armata. In base alla strategia selezionata l’AI deciderà di fare un operazione o meno. Come da diagramma l’AI agisce sempre in risposta alle domande del mediator.
L’AI potrà ripetere una certa operazione quante volte vorrà, a meno che il regolamento non lo impedisca.

ExchageTris



L'utente reale potrà scambiare un tris di carte dei territori cliccando sul bottone "Cards", presente nel "PlayerPanel". Quindi spunterà tre carte tra quelle mostrate e cliccando sul bottone "Change Tris" otterrà le armate da poter schierare. Ovviamente, il player virtuale potrà usufruire di questa funzione, ma non si interfaccerà con la GUI, ma chiamerà il metodo "exchangeTris" passando a parametro il tris da scambiare.

Conclusioni e possibile sviluppo futuro

Dove è stato possibile ed opportuno sono stati applicati i principi SOLID ed i design pattern, grazie ai quali è stato ridotto l'accoppiamento tra le classi, producendo così un codice di qualità.
Essendo la mappa molto personalizzabile, si potrebbero facilmente inserire nuovi file di configurazione, in modo tale da creare diverse mappe. In altre parole, qualsiasi utente potrebbe costruire a suo piacimento una nuova mappa e giocare a risiko su di essa.
In futuro, per un maggiore appeal grafico, si potrebbe anche modernizzare l'interfaccia grafica, aggiungendo qualche animazione ed introducendo elementi di interattività che coinvolgano maggiormente l'utente.
Si potrebbero affinare le strategie di gioco dei player virtuali, permettendo loro di ideare tattiche più argute. Si potrebbero dunque aggiungere nuovi stili di gioco, o magari dar loro la possibilità di cambiare stile durante il corso di una partita.
Inoltre, dal punto di vista codice è proponibile una migliore gestione delle eccezioni.