

Dipartimento di Ingegneria e Scienze dell’Informazione e Matematica

CORSO DI LAUREA IN INFORMATICA

Insegnamento Laboratorio di programmazione ad oggetti

Da sostituire con nome dell’applicazione

|  |  |  |  |
| --- | --- | --- | --- |
| **Membri del team** | | | |
| **Cognome e nome** | **Matricola** | | **Indirizzo e-mail** |
| **Marinucci Alessandro** | 261682 | | alessandro.marinucci2@student.univaq.it |
| Odoardi Davide | 292216 | | davide.odoardi@student.univaq.it |
| Fasciano Davide | 245433 | | davide.fasciano@student.univaq.it |
|  | |  | | |
|  | | | | |
|  | |  | | |

A.A. 2022/2023

Sommario

[Capitolo 1 – Caso di studio 2](#_Toc947706070)

[1.1 – Descrizione dell’applicazione 3](#_Toc1934072429)

[1.2 – Funzionalità 4](#_Toc827954761)

[1.3 – Domain Model 6](#_Toc1612348205)

[Capitolo 2-Implementazione 7](#_Toc1310444048)

# Capitolo 1 – Caso di studio

## 1.1 – Descrizione dell’applicazione

*L'obiettivo di questo progetto è progettare e implementare una versione digitale del gioco da tavolo* ***Risiko****, utilizzando il linguaggio di programmazione* ***Java****. L'applicazione sfrutta i concetti fondamentali della* ***programmazione orientata agli oggetti****, come classi, ereditarietà, polimorfismo e gestione degli eventi, per creare un'esperienza di gioco fedele all'originale.*

***Caratteristiche principali:***

* **Interfaccia Utente:** Basata su console, con utilizzo della libreria Jansi per migliorare la formattazione e i colori del testo.
* **Numero di Giocatori:** Supporta da 2 a 6 giocatori, permettendo partite multigiocatore locale.
* **Mappa di Gioco:** Rappresentazione dei territori e dei continenti, inclusi confini e connessioni, per simulare fedelmente il tabellone di Risiko.
* **Gestione delle Armate:** Implementazione delle meccaniche di distribuzione, attacco, difesa e spostamento delle armate tra territori.
* **Carte e Obiettivi:** Introduzione di carte territorio, carte obiettivo e jolly, che influenzano la strategia di gioco e determinano le condizioni di vittoria.
* **Dadi Virtuali:** Simulazione del lancio dei dadi per risolvere le fasi di attacco e difesa tra i giocatori.
* **Stato del Gioco:** Controllo delle diverse fasi del gioco, come la distribuzione iniziale delle armate, le fasi di attacco, rinforzo e spostamento.
* **Persistenza:** Possibilità di salvare e riprendere una partita in corso, garantendo flessibilità e continuità nell'esperienza di gioco.
* **Log delle Azioni:** Registrazione dettagliata su file di tutte le azioni svolte durante la partita, come attacchi, movimenti e cambi di turno, per permettere un'analisi post-partita.

***Tecnologie e Approcci Utilizzati:***

* **Programmazione Orientata agli Oggetti:** Uso estensivo di classi e oggetti per modellare le entità del gioco (giocatori, territori, armate, ecc.).
* **Collezioni Java:** Utilizzo di strutture dati come liste, mappe e set per gestire gli elementi dinamici del gioco.
* **Stream ed Espressioni Lambda:** Applicazione di Stream API ed espressioni lambda per operazioni efficienti sulle collezioni.
* **Gestione degli Eventi:** Implementazione di meccanismi per gestire eventi di gioco, come attacchi e cambi di turno.
* Modularità del Codice: Suddivisione del codice in pacchetti distinti (core, datamodel, service, impl, utils) per una migliore organizzazione e manutenzione.
* Persistenza dei Dati: Salvataggio e caricamento dello stato del gioco utilizzando la serializzazione degli oggetti.
* Gestione dei Colori e Formattazione del Testo: Utilizzo della libreria Jansi per migliorare l'esperienza utente nell'interfaccia console.

## 1.2 – Funzionalità

***Tipologie di Utenti:***

* ***Giocatori Umani:*** Partecipano al gioco inserendo comandi tramite l'interfaccia console.
* ***Amministratore di Gioco (Sistema):*** Gestisce le logiche interne del gioco, come la distribuzione delle armate e il controllo delle fasi.

***Funzionalità per i Giocatori Umani:***

***1.Creazione e Configurazione della Partita:***

* Inserimento del nome del giocatore.
* Scelta del colore rappresentativo.
* Selezione del numero di giocatori (da 2 a 6).

***2.Gestione delle Armate:***

* Distribuzione iniziale delle armate sui territori assegnati.
* Posizionamento strategico delle armate durante le fasi di rinforzo.
* Scambio di carte per ottenere armate aggiuntive.

***3.Movimenti e Attacchi:***

* Attacco a territori avversari adiacenti.
* Utilizzo di dadi virtuali per determinare l'esito degli attacchi e delle difese.
* Spostamento di armate tra territori controllati.

***4.Gestione delle Carte:***

* Ricezione di carte territorio dopo la conquista di almeno un territorio nel turno.
* Scambio di combinazioni di carte per ottenere rinforzi.
* Visualizzazione delle carte possedute.

***5.Obiettivi di Gioco:***

* Assegnazione di un obiettivo segreto a ogni giocatore.
* Possibilità di visualizzare l'obiettivo durante il gioco.
* Controllo del raggiungimento dell'obiettivo per determinare la vittoria.

***6.Interazione con l'Interfaccia Utente:***

* Inserimento di comandi tramite console per eseguire azioni.
* Visualizzazione dello stato attuale della mappa, delle armate e dei territori controllati.
* Accesso a un menu per salvare la partita, visualizzare informazioni o terminare il gioco.

***Funzionalità per l'Amministratore di Gioco (Sistema):***

***1. Inizializzazione della Partita:***

* Creazione della mappa con territori e continenti.
* Distribuzione casuale dei territori tra i giocatori.
* Calcolo del numero di armate iniziali in base al numero di giocatori.

***2.Gestione delle Fasi di Gioco:***

* Controllo delle fasi: distribuzione armate, attacco, spostamento.
* Gestione del turno dei giocatori e dell'ordine di gioco.
* Monitoraggio delle condizioni di vittoria.

***3.Gestione delle Carte e del Mazzo:***

* Creazione del mazzo di carte territorio e obiettivo.
* Distribuzione delle carte obiettivo ai giocatori.
* Gestione del mazzo di pesca e delle carte scartate.

***4.Persistenza e Salvataggio:***

* Salvataggio dello stato della partita su file per permettere di riprendere il gioco in un secondo momento.
* Caricamento di una partita precedentemente salvata.

***5.Log delle Azioni:***

* Registrazione su file di tutte le azioni svolte durante la partita (es. attacchi, movimenti, distribuzioni).
* Creazione di un log dettagliato per analisi o debugging.

***6.Gestione degli Eventi di Gioco:***

* Calcolo dei rinforzi basato sul controllo di territori e continenti.
* Verifica delle regole durante gli attacchi e le difese.
* Gestione delle eliminazioni dei giocatori e della fine della partita.

## 1.3 – Domain Model

## ***Capitolo 2-Implementazione***

***2.1 – Struttura dei Pacchetti***

*L'applicazione è organizzata in una serie di pacchetti per migliorare la modularità e la manutenibilità del codice. La struttura dei pacchetti è la seguente:*

* **it.univaq.disim.lpo.risiko.core.datamodel:** Contiene le classi che rappresentano i dati e le entità del dominio (es. Giocatore, Mappa, Territorio, Carta, ecc.).
* **it.univaq.disim.lpo.risiko.core.service:** Definisce le interfacce dei servizi che gestiscono la logica di business.
* **it.univaq.disim.lpo.risiko.core.service.impl:** Contiene le implementazioni delle interfacce dei servizi.
* **it.univaq.disim.lpo.risiko.core.utils:** Include classi di utilità utilizzate in tutta l'applicazione.

***Package Datamodel***

***Classe Gioco:*** Rappresenta lo stato globale della partita, inclusi i giocatori, la mappa, il mazzo di carte e lo stato del turno.

***Attributi Principali:***

* **fase:** La fase corrente del gioco.
* **giocatori:** Lista dei giocatori partecipanti.
* **mappa:** La mappa di gioco.
* **mazzoDiCarte:** Il mazzo di carte territorio.
* **currentTurnState:** Lo stato corrente del turno.
* **roundCount:** Il numero di turni trascorsi.
* **logFileName:** Il nome del file di log corrente.
* **partitaInCorso:** Indica se la partita è in corso.
* l**oadedGame:** Indica se la partita è stata caricata da un salvataggio.
* **ritornaAlMenu**: Indica se il gioco deve tornare al menu principale.

***Metodi Principali:***

* Costruttore per inizializzare un nuovo gioco.
* Getter e setter per gli attributi.

***Classe Giocatore:***

*Modella un partecipante al gioco, con informazioni sulle armate, i territori controllati, le carte possedute e l'obiettivo assegnato.*

***Attributi Principali:***

* nome: Il nome del giocatore.
* colore: Il colore delle armate del giocatore.
* armate: Il numero totale di armate possedute.
* territori\_controllati: Lista dei territori controllati.
* carte: Lista delle carte possedute.
* obiettivo: L'obiettivo segreto assegnato.
* haRicevutoCartaBonus: Indica se il giocatore ha ricevuto una carta bonus nel turno corrente.
* territoriConquistatiNelTurno: Numero di territori conquistati nel turno.

Metodi Principali:

* Metodi per aggiungere e rimuovere territori.
* Metodi per aggiungere e rimuovere carte.
* Metodi per gestire le armate.
* Getter e setter per gli attributi.

***Classe Mappa:***

Rappresenta il tabellone di gioco, contenente i continenti e i territori.

Attributi Principali:

* continenti: Lista dei continenti sulla mappa.
* Metodi Principali:
* Metodi per ottenere i continenti e i territori.
* Getter e setter per gli attributi.

***Classe Continente:***

Descrizione: Modella un continente sulla mappa, contenente una lista di territori.

Attributi Principali:

* nome: Il nome del continente.
* territori: Lista dei territori nel continente.

Metodi Principali:

* Getter e setter per gli attributi.

***Classe Territorio:***

Modella un territorio specifico, con informazioni sul giocatore che lo controlla, il numero di armate e i territori adiacenti.

Attributi Principali:

* nome: Il nome del territorio.
* giocatore: Il giocatore che controlla il territorio.
* armate: Il numero di armate presenti nel territorio.
* territoriAdiacenti: Lista dei territori adiacenti.
* continente: Il continente a cui appartiene il territorio.

Metodi Principali:

* Metodi per aggiungere e rimuovere armate.
* Metodi per gestire i territori adiacenti.
* Getter e setter per gli attributi.

***Classe Carta (astratta):***

Classe base per le carte del gioco.

Attributi Principali:

* tipo: Il tipo di carta (Cannone, Fante, Cavaliere, Jolly).
* territorio: Il territorio associato alla carta .

Metodi Principali:

* Getter e setter per gli attributi.

***Classe CartaTerritorio:***

Estende la classe Carta, rappresenta una carta territorio specifica.

Attributi Principali:

* Eredita gli attributi dalla classe Carta.
* Metodi Principali:
* Costruttore per inizializzare la carta con un territorio e un tipo.

***Classe CartaObiettivo:***

Rappresenta un obiettivo segreto assegnato al giocatore.

Attributi Principali:

* descrizione: La descrizione dell'obiettivo.

Metodi Principali:

* Costruttore per inizializzare l'obiettivo con una descrizione.
* Getter per la descrizione.

***Classe MazzoDiCarte:***

Gestisce il mazzo delle carte territorio.

Attributi Principali:

* carte: Una LinkedList di carte disponibili nel mazzo.
* Metodi Principali:
* pescaCarta(): Permette di pescare una carta dal mazzo.
* restituisciCarte(List<Carta> carte): Restituisce le carte scambiate al mazzo.
* inizializzaMazzo(List<Territorio> territori): Inizializza il mazzo con le carte territorio e jolly.

***Classe TurnState:***

Contiene lo stato corrente del turno di un giocatore.

Attributi Principali:

* currentPhase: La fase attuale del turno (START\_TURN, DISTRIBUTE\_ARMIES, MENU, END\_TURN).
* armateDaDistribuire: Il numero di armate da distribuire nel turno.
* turnoTerminato: Indica se il turno è terminato.
* armateTotali: Il totale delle armate ricevute all'inizio del turno.
* numeroTerritori: Il numero di territori controllati all'inizio del turno.
* numeroContinenti: Il numero di continenti controllati all'inizio del turno.

Metodi Principali:

Getter e setter per gli attributi.

Continua per i vari package......(da verificare se dobbiamo scrivere tutto)