**Italian Checkers**

**Relazione di progetto del corso 088851**

**Immagine che contiene testo, logo, Carattere, Elementi grafici

Descrizione generata automaticamenteProgetto Software**

**Autore:***Tony David Matrundola*

**Matricola:***962910*

**Anno Accademico** 2024 – 2025

Politecnico di Milano

**Indice**

1. [Introduzione](#_bookmark1) 3

1.1 [Obiettivi del Progetto](#_bookmark2) 3

1.2 [Regole della Dama Italiana](#_bookmark3) 3

1.3 [Tecnologie Utilizzate](#_bookmark4) 4

2 [Architettura del Sistema](#_bookmark8) 4

2.1 [Panoramica Architetturale](#_bookmark9) 4

**2.2** [**Model-View-Controller (MVC)**](#_bookmark11) **4**

**2.3 Client-Server 4**

**2.4** [**Strategy Pattern**](#_bookmark13) **5**

3 [Implementazione Dettagliata](#_bookmark16) 6

3.1 [Package Model](#_bookmark17) 6

3.2 [Package Client](#_bookmark18) 6

3.3 [Package Server](#_bookmark19) 6

3.4 [Intelligenza Artificiale](#_bookmark20) 6

4 [Interfaccia Grafica](#_bookmark22) 9

4.1 Menu Principale 9

4.2 Board 9

4.3 VictoryScreen 10

4.4 DrawScreen 11

5 [Networking e Comunicazione](#_bookmark25) 11

5.1 [Protocollo di Comunicazione](#_bookmark26) 11

6 Test e Coverage 12

7 Configurazione e Build 13

**Capitolo 1**

**Introduzione**

Il progetto prevede la realizzazione di un applicativo software denominato **Italian Checkers**, che implementa una versione completa del gioco della dama italiana. Il sistema permette di gestire partite in tre modalità diverse: locale (due giocatori), CPU (contro intelligenza artificiale) e online (multiplayer via rete). Le informazioni di gioco sono gestite attraverso una robusta architettura client-server, garantendo sincronizzazione e coerenza dello stato di gioco. L'interfaccia utente è realizzata utilizzando JavaFX, offrendo un'esperienza moderna e intuitiva.

## **1.1 Obiettivi del Progetto**

Il progetto Italian Checkers si propone di implementare una versione completa e funzionale del gioco della Dama Italiana, con le seguenti caratteristiche principali:

**Completezza delle regole**: Implementazione fedele delle regole ufficiali della Dama Italiana

**Modalità multiple**: Supporto per gioco locale, contro CPU e online

**Interfaccia moderna**: GUI responsive e intuitiva realizzata con JavaFX

**Qualità del software**: Codice ben documentato, testato e manutenibile

## **1.2 Regole della Dama Italiana**

La Dama Italiana segue regole specifiche che differiscono dalle varianti internazionali:

1. **Scacchiera**: 8x8 caselle, gioco solo su caselle scure
2. **Pezzi iniziali**: 12 pedine per giocatore
3. **Movimento pedine**: Solo in avanti, diagonalmente
4. **Cattura**: Obbligatoria quando possibile, salto diagonale
5. **Multi-jump**: Catture multiple consecutive obbligatorie
6. **Promozione**: Pedina diventa dama raggiungendo l’ultima riga
7. **Dame**: Movimento in tutte le direzioni diagonali
8. **Vittoria**: Eliminare tutte le pedine avversarie
9. **Patta**: 40 mosse senza catture

## **1.3 Tecnologie Utilizzate**

* + - * **Java 21**: Linguaggio principale con supporto per record e pattern matching
      * **JavaFX 21**: Framework per interfaccia grafica
      * **Maven**: Build tool e gestione dipendenze
      * **JUnit 5**: Framework di testing unitario
      * **Mermaid 10.9.0.1**:Diagrammi UML

# **Capitolo 2**

# **Architettura del sistema**

## **2.1 Panoramica Architetturale**

Il sistema Italian Checkers adotta un’architettura modulare basata sul pattern Model-View-Controller (MVC) esteso per supportare funzionalità di rete. La struttura principale e organizzata in quattro package principali: client (interfaccia utente e controllo client), server (logica server e AI), model (modello dati del gioco), e common (utilità condivise).

**2.2 MVC (Model-View-Controller)**

L'applicazione segue il pattern **Model-View-Controller** per garantire una separazione netta delle responsabilità:

* **Model:** Gestisce la logica di gioco, lo stato della scacchiera e le regole
* **View:** Interfaccia grafica JavaFX per l'interazione con l'utente
* **Controller:** Coordina le interazioni tra Model e View

┌─────────────┐ ┌─────────────┐ ┌─────────────┐

│ View │◄──►│ Controller │◄──►│ Model │

│ (JavaFX) │ │ (Game │ │ (Board, │

│ │ │ Logic) │ │ Rules) │

└─────────────┘ └─────────────┘ └─────────────┘

**2.3 Architettura Client-Server**

Per la modalità online, il sistema utilizza un'architettura distribuita

┌─────────────┐ ┌─────────────┐ ┌─────────────┐

│ Client 1 │◄───────►│ Server │◄───────►│ Client 2 │

│ (Player 1) │ TCP │ │ TCP │ (Player 2) │

└─────────────┘ └─────────────┘ └─────────────┘

### **2.4 Strategy Pattern**

Utilizzato per gestire diverse modalità di gioco:

* Modalita` locale (due giocatori)
* Modalita` CPU (con AI)
* Modalita` online (rete)

Segue **Diagramma architetturale:**

Immagine che contiene testo, schermata, diagramma, linea

Descrizione generata automaticamente**Capitolo 3**

**Implementazione Dettagliata**

## **3. 1 Package Model**

Il package model contiene le entità fondamentali del gioco, implementate seguendo principi di incapsulamento e coesione. La classe **Piece** estende StackPane per fornire rappresentazione visiva e comportamenti della pedina, con gestione eventi per drag and drop, validazione dei controlli di integrità nelle operazioni, e gestione della promozione e trasformazione in dama. L’enum **PieceType** definisce i tipi di pedine con direzione di movimento: GRAY muove verso il basso , WHITE muove verso l’alto. La classe **Tile** rappresenta una casella della scacchiera con funzionalità di evidenziazione, utilizzando GameConfig per dimensioni e posizionamento.

## **3. 2 Package Client**

Il **ChessBoardClient** funge da controller principale che gestisce l’interfaccia utente e coordina le interazioni. Contiene i componenti principali come la matrice di tile, gruppi per elementi grafici, stato del gioco con modalità , turni e multi-jump.

Il sistema supporta **tre modalità** distinte: locale senza connessioni di rete, e modalità online/CPU con connessione al server.

## **3. 3 Package Server**

L’architettura server utilizza un pattern multi-threaded per gestire connessioni concorrenti. Il server accetta connessioni, gestisce modalità multiplayer e CPU , e crea handler dedicati per ogni partita.

Il **ClientHandler** gestisce la logica completa di una partita, inclusa la gestione dell’AI per modalità CPU, controllo delle catture obbligatorie, sistema multi-jump, e conteggio delle mosse per la regola dei 40 turni.

## **3. 4 Intelligenza Artificiale**

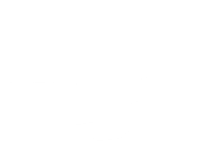
L’intelligenza artificiale è implementata nella classe **CheckersAI** ed implementa un approccio basato su euristica e priorità per prendere decisioni strategiche. L’algoritmo trova tutte le mosse possibili, le ordina per priorità decrescente, e seleziona casualmente fra le mosse migliori.

Di seguito riportato il sistema di priorità del CPU:

|  |  |  |
| --- | --- | --- |
| **Tipo Mossa** | **Priorit`à** | **Bonus** |
| Cattura pedina normale | 10 | +2 per cattura dama, +1 per promozione |
|  |  |  |
| Promozione | 8 | Quasi importante quanto cattura |
| Avanzamento verso promozione | 1-5 | Basato su distanza dalla promozione |
|  |  |  |
| Movimento dama | 3 | +1 se vicino a nemici |
| Movimento sicuro | +2 | Bonus per mosse che non creano vulnerabilità |
|  |  |  |

Immagine che contiene diagramma, testo, schizzo, Piano

Descrizione generata automaticamenteSegue **Diagramma UML:**



**Capitolo 4**

**Interfaccia Grafica**

## **4. 1 Menu Principale**

Immagine che contiene testo, schermata, Carattere, design

Descrizione generata automaticamenteIl menu principale (**StartScreen**)offre selezione modalità con pulsanti stilizzati ed effetti hover:

## **4. 2 Board**

La board di gioco è composta di scacchiera 8x8 con caselle alternate chiare/scure, evidenziazione delle caselle valide in verde, indicazioni multi-jump obbligatori, messaggi di stato del gioco. Ci sono inoltre **Timer** per gestione del tempo con posizionamento dinamico basato sul giocatore, e **ScoreDisplay** per statistiche partita con funzionalità toggle.

Immagine che contiene schermata, quadrato, testo, gioco da tavolo

Descrizione generata automaticamenteSegue **schermata della modalità CPU:**

Segue **schermata di esempio delle mosse della Dama (modalità locale con due Timer):**

Immagine che contiene Giochi e sport indoor, gioco da tavolo, schermata, quadrato

Descrizione generata automaticamente

## **4. 3 VictoryScreen**

Il VictoryScreen presenta la schermata finale con le statistiche della partita, due bottoni con effetto hover di cui “Play Again” per giocare di nuovo e “Main Menu” per tornare al menu principale.

Immagine che contiene testo, schermata, quadrato

Descrizione generata automaticamenteDi seguito una **schermata di vittoria:**

## **4. 4 Draw Screen**

Nel caso in cui si arrivasse a 40 mosse senza cattura viene mostrata la stessa schermata ma che indica il pareggio (**Draw**)  
Segue un **esempio**:

Immagine che contiene testo, schermata, quadrato, cerchio

Descrizione generata automaticamente

**Capitolo 5**

**Networking e Comunicazione**

## **5. 1 Protocollo di Comunicazione**

Il sistema utilizza un protocollo testuale personalizzato per la comunicazione client-server:

|  |  |  |
| --- | --- | --- |
| **Tipo** | **Formato** | **Descrizione** |
| Mossa normale | x1 y1 x2 y2 NORMAL | Movimento da (x1,y1) a (x2,y2) |
| Cattura | x1 y1 x2 y2 KILL cx cy | Cattura con pedina eliminata in (cx,cy) |
| Mossa rifiutata | x1 y1 x2 y2 NONE | Server rifiuta la mossa |
| PING | PING | Server indica inizio turno client |
| Fine partita | x1 y1 x2 y2 END1/END2 | Vittoria giocatore 1 o 2 |
| Patta | x1 y1 x2 y2 DRAW | Partita terminata in patta |

**Capitolo 6**

**Test e Coverage**

Il progetto implementa una strategia di testing completa con diversi livelli:

**Unit Tests** per singole classi e metodi , ad esempio nel model per logica pedine , mosse e regole o tests per l’AI.

**Integration Tests** per test di integrazione tra componenti, come test per comunicazione Client-Server.

Immagine che contiene testo, schermata, software

Descrizione generata automaticamente

**Capitolo 7**

**Configurazione e Build**

Il progetto utilizza Maven per la gestione delle dipendenze e del build process. La configurazione include Java 21, JavaFX 21 per GUI, Junit5 per testing completo e plugin per assemply JAR.

Per funzionare correttamente si dovrà prima avviare l’applicativo Server e poi il/i Client. Per avviare un’istanza del Server è necessario aprire l’eseguibile da riga di comando e navigare nella directory dove si trova l’eseguibile e digitare il seguente comando:

java -jar DamaProject-1.0-SNAPSHOT-jar-with-dependencies.jar -s

In questo modo si aprirà un’istanza del server sula porta 1234.

In seguito per avviare uno o due Client dopo aver fatto la stessa procedura va digitato il seguente comando:

java -jar DamaProject-1.0-SNAPSHOT-jar-with-dependencies.jar -c

Si aprirà così il menu principale dove si potrà scegliere la modalità di gioco desiderata, nel caso si scegliesse l’online il Server resterà in attesa di un secondo Client che a sua volta dovrà cliccare sulla modalità online per avviare la partita.