

Dipartimento di Ingegneria e Scienze dell’Informazione e Matematica

CORSO DI LAUREA IN INFORMATICA

Insegnamento Laboratorio di programmazione ad oggetti

Scacchi-IRitardatari

|  |  |  |  |
| --- | --- | --- | --- |
| **Membri del team** | | | |
| **Cognome e nome** | **Matricola** | | **Indirizzo e-mail** |
| Vampa Francesco | 286458 | | francesco.vampa@student.univaq.it |
| Altieri Giovanni | 279062 | | giovanni.altieri@student.univaq.it |
| De Vincentiis Miriam | 286239 | | miriam.devincentiis@student.univaq.it |
| Ubaldi Giammarco | 279658 | | giammarco.ubaldi@student.univaq.it |
|  | |  | | |
|  | | | | |
|  | |  | | |

A.A. 2022/2023

Sommario

[Capitolo 1 – Caso di studio 2](#_Toc38393257)

[1.1 – Descrizione dell’applicazione 2](#_Toc38393258)

[1.2 - Funzionalità 3](#_Toc38393259)

[1.3 Domain Model 4](#_Toc38393260)

# Capitolo 1 – Caso di studio

## 1.1 – Descrizione dell’applicazione

Il progetto prevede un’applicazione che permette di gestire una partita di scacchi. La partita viene giocata da due giocatori. Il sistema deve prevedere che una partita si possa svolgere tra due giocatori oppure tra un giocatore e il computer, da noi denominato Bot.

La partita viene gestita da PartitaService. Essa gestisce le mosse dei giocatori e tutto quello che ruota attorno. Creando l’interfaccia grafica abbiamo implementato una classe che gestisce la parte grafica e quindi l’interazione tra i giocatori e il codice. Quindi ci sono differenti ruoli all’interno del progetto che possono effettuare operazioni differenti sul sistema stesso.

I ruoli sono i seguenti: Giocatore Umano, Giocatore Bot, ScacchieraController e PartitaService.

Un Giocatore Umano può:

* Muovere i pezzi
* Richiedere la patta
* Annullare le ultime mosse
* Abbandonare senza salvare una partita (arrendersi)
* Abbandonare e salvare una partita
* Riprendere una partita già salvata

Un Giocatore Bot può:

* Muovere i pezzi

La ScacchieraController può:

* Aggiungere Pezzi all’interno della Scacchiera grafica
* Eseguire una mossa
* Visualizzare le mosse eseguite
* Visualizzare i pezzi mangiati
* Trovare ed elencare le mosse disponibili per un pezzo della scacchiera
* Annullare le ultime mosse e ristabilire la scacchiera precedente alla mossa eliminata
* Iniziare una nuova partita
* Salvare una partita
* Terminare una partita
* Gestisce la patta

PartitaService può:

* Recuperare il salvataggio di una partita
* Gestire il cambio turno
* Controllare lo scacco matto e lo scacco
* Salvare una partita

## 1.2 – Funzionalità

Per gestire una partita di scacchi, il progetto possiede una serie di funzionalità specifiche e di operazioni legate a dei ruoli che consentono il suo corretto funzionamento.

* Requisiti funzionali Giocatore Umano

La classe Giocatore Umano è un’estensione della classe Giocatore. Di un giocatore umano usiamo come variabili il nome e il colore del giocatore che può essere bianco o nero. Il colore lo abbiamo implementato con una variabile booleana che indica con true il bianco e con false il nero.

Il Giocatore Umano può selezionare e decidere la mossa dei pezzi presenti sulla scacchiera. Se desidera, può annullare le ultime 5 mosse da lui effettuate, potendo riprendere la partita.

Il salvataggio permette a un Giocatore Umano di poter riprendere una partita già salvata in precedenza, con l’utilizzo della Serializzazione e dell’interfaccia Serializable. Essa permette la scrittura su file per rappresentare un’oggetto attraverso flussi di byte.

Un Giocatore Umano può, inoltre, decidere di abbandonare una partita. Una volta abbandonata può scegliere di salvare tale partita o di abbandonare il gioco senza salvare, quindi arrendersi.

L’alternativa dell’abbandono di una partita è la patta. Un giocatore può richiedere la patta e solo se la patta viene accetta dall’altro giocatore, la partita si conclude con la “parità”. Questo si verifica se la partita si gioca tra due giocatori Umani. Nel caso in cui un Giocatore Umano gioca con il Bot il progetto, così implementato, non permette al Bot di accettare o no la patta. Quindi in questo caso il Giocatore Umano, quando richiede la patta effettua un annullamento della partita.

* Requisiti funzionali Giocatore Bot

Anche la classe Bot eredita metodi e variabili dalla classe Giocatore. La differenza tra Umano e Bot è decisiva perché il Bot deve solo effettuare mosse sui pezzi presenti sulla scacchiera. Il Bot non effettua mosse intelligenti ma muore un pezzo che può muoversi in una cella scelta in modo randomico. Quindi la scelta del pezzo e della mossa del pezzo sono esclusivamente casuali.

* Requisiti funzionali ScacchieraController

La classe ScacchieraController si occupa di gestire la parte grafica della scacchiera. Tale classe implementa l’interfaccia Serializable. ScacchieraController contiene sia variabili legate alla grafica come label, button, MenuItem e altro, sia variabili legate ai metodi come partita, pezzoSelezionato, colorePezzoSelezionato, utili per gestire la mossa dei pezzi.

Nel momento in cui viene creata una nuova partita, viene inizializzata una nuova ScacchieraService. ScacchieraController con altrettanti metodi renderà visibile ciò che implementa ScacchieraService.

Banalmente, ScacchieraController può iniziare una nuova partita, salvare e terminare una partita. Inoltre, può aggiungere i pezzi all’interno della scacchiera grafica, eseguire la mossa che i giocatori decidono di compiere e visualizzare i pezzi mangiati.

Graficamente, quando un giocatore seleziona un pezzo, saranno visibili le mosse che egli può effettuare identificando le celle in cui il pezzo si può muovere in due modi diversi in base all’obbiettivo ovvero se il pezzo mangia o se si sposta.

Infine, ScacchieraController gestisce l’annullamento delle ultime 5 mosse e la patta attraverso dei metodi implementati all’interno della classe. Nel momento in cui il giocatore annulla una mossa, ScacchieraController dovrà ristabilire la scacchiera precedente alla mossa che il giocatore ha eliminato; quindi, modifica le liste che contengono i pezzi mangiati, modifica le celle della scacchiera con i dovuti pezzi e modifica il file che contiene lo svolgimento della partita con le mosse effettuate.

ScacchieraController implementa la patta secondo quando detto per il Giocatore Umano. Questa classe gestisce la patta graficamente collegando il concetto di patta a strumenti grafici come button. Inoltre, ci sono degli alert che permettono ai giocatori di richiedere e confermare o no la patta.

* Requisiti PartitaService

La classe PartitaService gestisce la partita e tutti i suoi servizi non legati alla grafica. Tale classe implementa Serializable. Le variabili utilizzate sono g1 e g2 di tipo Giocatore, scacchieraController di tipo ScacchieraController utilizzate per creare una nuova partita. Ci sono altre variabili utili per lo svolgimento della partita come turnoGiocatore che con true indica il giocatore bianco e con false il giocatore nero.

All’interno della classe sono presenti più costrutti ognuno utilizzato per istanziare PartitaService in base all’utilità. Un costruttore istanzia l’oggetto partita considerando i due giocatori, il bot se è presente o meno e la scacchiera della partita univoca per questa istanza. Un secondo costrutto inizializza in maniera non statica i due giocatori della partita. Questo implica che non sia presente il bot quindi che sia una partita contro due giocatori.

PartitaService gestisce il cambio turno per far in modo che i giocatori muovano un pezzo alla volta quando è il proprio turno. Questa funzionalità non viene solo usata alla singola partita, ma viene anche utilizzata per gestire la parte grafica o un’altra funzionalità come la patta. Essa utilizza il cambio turno per richiedere e attendere la conferma della patta.

PartitaService, infatti, si occupa del salvataggio dei file delle partite; è possibile recuperare una partita salvata. Nel caso della prima, è possibile scegliere tra i file che contengono le partite salvate secondo vari criteri quali il numero delle mosse di ogni giocatore, il numero di pezzi mangiati di ogni giocatore e il valore dei pezzi ancora in gioco di ogni giocatore.

Infine, PartitaService si occupa della gestione dello scacco matto e dello scacco. Possiede un metodo che controlla se il re è sotto scacco. Questo metodo verrà invocato all’interno del metodo che esegue la mossa; tale metodo è presente all’interno della classe ScacchieraController.

## 1.3 – Domain Model

RIMUOVERE DESCRIZIONE TESTO SOTTOSTANTE

Mostrare un class diagram che rappresenta il domain model della applicazione contenente entità e relazioni tra esse. Le entità e le relazioni derivano dalle funzionalità descritte nella sezione precedente.