Alessandro Fuochi Id083311 Univr

SPECIFICHE

Progetto di Programmazione II 2014/2015

Il progetto consiste nello sviluppo di un software per il gioco degli scacchi. Il programma dovrà visualizzare una scacchiera con i pezzi degli scacchi posizionati nella configurazione iniziale. L'interfaccia grafica deve rendere possibile di muovere i pezzi ***secondo le regole degli scacchi***. Il software deve riconoscere ***quando una partita è finita*** e deve proporne un'altra.

- Non è previsto che il software implementi un'intelligenza artificiale, è solo un gioco fra umani
- Non è prevista l'implementazione delle regole dell'arrocco e del movimento iniziale dei pedoni

Si chiede invece:

- di scrivere alcune ***classi di test JUnit*** per il software sviluppato
- di gestire lo sviluppo tramite un sistema di controllo distribuito delle versioni del codice sorgente, tipo git su github/bitbucket

Il software dovrà essere realizzato in Java, usando l'***interfaccia grafica Swing***.
La consegna avverrà inviando al docente il riferimento al repository (tipo indirizzo git) da cui scaricare il software.

Interpetazione Conformità

1) ***secondo le regole degli scacchi***
(con limitazioni da specifiche)

Parte il bianco cliccando su un pezzo(fase Selezione) e successivamente cliccando su una destinazione possibile (fase Mossa) (*guardare variazione colore bordi caselle*) ammessa .
Segue analogo per il nero fino a fine partita.

1) Lo scacco viene segnalato e annullata la mossa in caso causi 'auto scacco'.
2) Viene visualizzato il percorso possibile del pezzo selezionato
3) Viene segnalato lo scacco semplice
4) *Non e' gestito arrocco*
5) *Non e' gestito avanzamento a 2 pedoni iniziale*

Sono gestiti

1) *promozione pedone*
2) *regola patta 50 mosse* (no mosse pedoni / no catturato pezzo da entrambi)

2) quando una partita è finita

Premessa

Il controller riceve (da mouse o da lista demo) la mossa in due fasi *selezione* e *mossa*.
Alla '*selezione'* del pezzo ,verificato che abbia almeno una possibilità di muoversi
e sia del colore corretto,viene creata una maschera con tutti i percorsi possibili del medesimo.
Nella fase di '*mossa*' la mossa viene accettata in base alla maschera creata nel momento della selezione che praticamente costituisce un filtro di controllo.
La maggior parte delle condizioni di stallo e patta sono di fatto pattuite fra i giocatori : sono state prese in considerazione le condizioni di numero minimo pezzi e di possibilità di muoversi del re anche se non attaccato.

A questo punto interviene il test *patta 50 mosse, stallo, scacco matto*:

Patta 50 mosse:
1)verifica eleminazione pezzi in ultime 50 mosse
2)verifica movimento pedoni
Stop in caso condizioni patta 50 mosse e proposta nuova partita.

Stallo: (Stall)

1) verifica numero minimo di pezzi
2) con verifica possibilità di movimento del re (non in scacco) per determinare lo stallo.
Stop in assenza di possibilità di prosecuzione gioco per i due re e proposta nuova partita.

Successivamente se si verifica la condizione di eventuale promozione del pedone ,si riverifica stallo in quanto cambiata la capacita offensiva del pezzo .

Scacco matto (Checkmate)
1) verifica che esista un attacco
2) eventuale verifica di possibile risposta.
Stop in assenza di possibilità per il re sotto scacco e proposta nuova partita.
In caso di 'auto scacco' la mossa viene annullata.
Successivamente se si verifica la condizione di eventuale promozione del pedone ,si riverificano scacco, matto in quanto cambiata la capacita offensiva del pezzo.

3) classi di test JUnit
Sono presenti test JUnit e demo di scacchi ,promozioni pedone, ecc.

4) interfaccia grafica Swing
Gli oggetti impiegati sono compatibili Swing .*La visualizzazione di questo documento (formato pdf) all'interno dell'applicazione viene aggiunta solo se l'ambiente lo permette ed è installato un lettore .pdf di default*. Questo per garantire la compatibilita del resto dell'applicazione

Note varie

Tutte le relazioni e le logiche "dati" sono nella classe model.Default

Per facilitare le prove è stata inserita la possibilità di salvare la partita (accodata a file) dandogli un nome che verrà richiesto dopo aver cliccato sul tasto salva partita e resa disponibile nella combo box per essere rieseguita.

E' presente JDoc per le classi piu complesse ('" non auto esplicative "')

File icone dei pezzi

Nella direcory data sono contenuti i file .bmp delle icone dei pezzi.Sono state utilizzate immagini 'free' senza diritti. La loro presenza viene controllata all'esecuzione dell'applicazione.
Il programma verifica la sitassi delle path richiesta dal sistema operativo e usa path compatibili sia 'like' Unix che Windows. ***In questo senso il programma e' stato provato su Ubuntu e su Windows con Java 8 (Sviluppato con Eclipse (Luna)).***

Memoria occupata dall 'applicazione

Per contenere l'occupazione della memoria si e' scelto di:

1) utilizzare (quando possibile) strutture realizzate da bytes (-127/+128) e boolean (true/false)
2) inserire le copie di backup delle matrici su array list con la creazione del "record" di backup prima del backup e la rimozione (quando possibile) del "record" dopo il restore.

Una indicazione approssimata (segue piu' l'utilizzo di memoria da parte di jvm che dell'applicazione) è visibile durante l'uso del programma.