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

Nota

Per auto scacco si intende quando viene scoperto attacco al proprio re precedentemente coperto dal pezzo appena mosso.

- ***1) secondo le regole degli scacchi*** *(con limitazioni da specifiche)*
  1) Lo scacco viene segnalato e annullata la mossa in caso causi 'auto scacco'
  2) Viene visualizzato e autorizzato il percorso possibile per il pezzo selezionato
  3) Viene segnalato lo scacco semplice
  4) *Non e' gestito arrocco*
  5) *Non e' gestito movimento iniziale dei pedoni*

  Sono gestiti
  1) *promozione pedone*

2) ***quando una partita è finita***

Premessa

La maggior parte delle decisioni in merito al fine partita per stallo e patta ,sono di fatto richieste dai giocatori:per la forzatura delle queste condizioni ,vengono presi in considerazione il numero minimo pezzi e la loro possibilità di muoversi .

Dinamica

Il controller riceve (da mouse o da lista demo) la mossa distinguendo l'evento in due fasi : *selezione* e *mossa*.
Alla *selezione* del pezzo ,verificato che abbia almeno una possibilità di muoversi e che sia del colore corretto,viene creato un array contenente tutti i percorsi possibili del medesimo.
Nella fase di *mossa* la mossa viene validata se conforme ai percorsi ammessi dall'array di controllo creato nel momento della selezione .
In caso di 'auto scacco' la mossa viene annullata.

Condizioni gestite come fine partita :

1) scacco matto re sotto scacco e senza possibilita di risolvere scacco
2) *troppi pochi pezzi per entrambi i giocatori (impostato a 3)*
3) *regola ripetizione mosse (una certa posizione si è ripetuta per 3 volte anche non di seguito sulla scacchiera)*
4) *impossibilita di muovere pezzi per entrambi i giocatori (stallo)*
5) *regola patta 50 mosse* (no mosse pedoni / no catturato pezzo da entrambi)

Ci si e' liberamente , senza contatti ne validazioni alcune, riferiti a
http://scacchi.qnet.it/manuale/scopo.htm

3) classi di test JUnit
Sono presenti test JUnit e demo di partite .

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

Varie
Il codice per i listener oggetti swing e' stato scritto per java8.
I commenti nel codice sono stati fatti in formato JDoc .
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 (accodandola a un file in directory data) dandogli un nome che verrà richiesto dopo aver cliccato sul tasto salva partita e resa disponibile nella combo box per essere rieseguita.

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.