Scacchi in Java

Progetto del corso
Programmazione ad Oggetti

A.A. 2007/08

Si devono scrivere le classi Java che implementano il gioco degli scacchi in Java. Scopo del progetto sarà strutturare opportunamente il codice dimodochè la scomposizione del problema segua il più possibile la filosofia object–oriented.
NB: non si deve scrivere un sistema in grado di giocare contro un avversario umano, ma un sistema che implementa la scacchiera del gioco, ovvero che mantiene la posizione corrente del gioco ed esegue una singola mossa o una sequenza di mosse verificandone la correttezza.

Descrizione del Progetto e Suggerimenti

Idealmente dovrebbe essere scritto del codice adatto a essere riutilizzato ed esteso per implementare anche per altri giochi su scacchiera a due giocatori, analoghi agli scacchi ma con diverse regole (come Dama, Go, etc.). Per far ciò è necessario implementare delle classi astratte o interfacce opportune, di utilizzo generale non ancora istanziate con le regole del gioco degli scacchi.

Ecco un ipotetico modo di procedere:

classe `Scacchiera` Per semplicità supponiamo che la scacchiera sia rettangolare e bidimensionale. La scacchiera dovrà quindi avere un costruttore che riceve come argomenti la dimensione orizzontale H e verticale V della scacchiera. La scacchiera dovrà avere un metodo che verifica se una data coppia di coordinate è valida. La scacchiera dovrà memorizzare la posizione corrente del gioco, e quindi ogni casella dovrà contenere un riferimento al pezzo che la occupa oppure un puntatore `null`, nel caso sia vuota. Scrivere anche un metodo per visualizzare la posizione corrente della scacchiera (potete usare la `toString()`).

Ad esempio, la posizione iniziale potrà semplicemente essere visualizzata come segue:

```
T C A D R A C T
P P P P P P P P
. . . . . . . .
. . . . . . . .
. . . . . . . .
. . . . . . . .
p p p p p p p p
t c a d r a c t
```

con l'idea che lettere minuscole rappresentano pezzi bianchi e lettere maiuscole i pezzi neri (P=pedone, T=torre, A=alfiere, C=cavallo, R=re, D=donna) e il punto rappresenta la casella vuota. La scacchiera delegherà ai pezzi la stampa di se' stessi.

classe `Pezzo` Ogni pezzo dovrà avere gli attributi che specificano il colore (o giocatore di appartenenza), un metodo per visualizzarlo (potete usare la `toString()`), e siccome in generale le mosse sono legali in riferimento allo stato della scacchiera, anche un riferimento alla scacchiera in cui è posizionato il pezzo.

classe `Partita` È la classe che riceve i messaggi che gestiscono la partita:

inizio nuova partita causa l'inizializzazione della scacchiera con la posizione iniziale degli scacchi;

mossa riceve le coordinate iniziali e finali della mossa[1], chiede alla scacchiera se nella casella di partenza c'è un pezzo del giusto colore, e delega al pezzo stesso la verifica della legalità della mossa. Infine esegue lo spostamento del pezzo, valutando se la partita è finita perchè uno dei due giocatori ha vinto;

fine partita ad esempio per abbandono di uno dei due giocatori.

stampa mosse giocate visualizza la sequenza di mosse finora giocate (potrebbe essere l'ideale implementazione della `toString()` per questa classe);

ritiro di una semi-mossa tra i metodi di `partita`, inserire la possibilità di ritirare una mossa (cosa assolutamente vietata quando si gioca tra gentiluomini `:)`.

Suggerisco di definire una classe per ogni diverso tipo di pezzo (ciascuna classe sarà sottoclasse della classe generale `Pezzo`) e delegare ai pezzi stessi il controllo della legalità di una mossa: ciò permetterà di rendere il codice più semplice ed estensibile[2].

Potrebbe essere alquanto laborioso implementare correttamente la verifica dello scacco matto: provateci. Potrebbe essere utile implementare nelle classi che definiscono il comportamento dei pezzi, un metodo che restituisce tutte le case controllate da un pezzo, e tutte le sue possibili mosse. Di conseguenza, anche la classe `Scacchiera` può determinare tutte le mosse possibili (o tutte le case controllate) di ciascun giocatore, chiedendole a tutti i singoli pezzi (ricordatevi infatti che una mossa è legale se il re non rimane in una casa controllata da un pezzo avversario).

Eventuali situazioni anomale (tipo mosse illegali etc.) andrebbero preferibilmente gestite con eccezioni.

Il test del buon funzionamento può avvenire sia con una semplice interazione con l'utente (in tal caso suggerisco di codificare un piccolo set di comandi per inizio partita, mosse etc.), sia, più semplicemente con una classe `TestaScacchiera` che nel `main()` invia messaggi alle altre (inizio, mosse etc.), simulando una o più partite.

[1] tradizionalmente, come nella battaglia navale, le colonne sono indicate con lettere, e le righe (dette *traverse*) con numeri. Voi potete comunque usare una coppia di numeri se preferite

[2] ad esempio, ci sono varie versioni di scacchi *eterodossi*, in cui ci sono pezzi con movimenti diversi rispetto a quelli tradizionali

Facoltativo: la Quadriglia

Chi volesse sperimentare le virtù composizionali della Programmazione ad Oggetti, potrebbe provare a verificare che con poco sforzo dovrebbe riuscire a implementare il gioco della quadriglia, con cui gli stacanovisti della scacchiera passano il tempo morto ai tornei tra una partita e l'altra.

Un gioco di quadriglia consiste sostanzialmente di due giochi di scacchi. Due coppie di giocatori si affrontano su due scacchiere. La situazione iniziale di ciascuna scacchiera è identica a quella degli scacchi tradizionali. In una coppia di giocatori alleati, uno gioca col nero e l'altro col bianco (nella realtà seduti dalla stessa parte del tavolo). Il giocatore che ha i bianchi, quando effettua una presa, passa al suo compagno il pezzo nero che ha catturato. Il giocatore che ha i neri, quando effettua una presa, passa al suo compagno il pezzo bianco che ha catturato. Un giocatore, a ogni turno, può scegliere uno tra due tipi di mossa:

- una normale mossa del gioco degli scacchi;
- se ha pezzi a disposizione forniti dal compagno, aggiungere un pezzo in una casella libera, con l'unico vincolo che i pedoni possono essere aggiunti solo nelle prime 6 righe, per evitare che promuovano troppo facilmente.

La vittoria di un giocatore blocca la partita e implica la vittoria della coppia.

Le due partite sono per il resto indipendenti (cioè le mosse di ciascuna partita si alternano tra i due giocatori, ma è possibile che in una scacchiera si facciano molte mosse, mentre nel frattempo sull'altra non ci siano state mosse).

Ovviamente, oltre a istanziare due partite di scacchi, bisognerà aggiungere delle strutture dati e dei metodi per gestire la comunicazione tra le due partite.

Facoltativo: l'Interfaccia Grafica

Chi avesse piacere di imparare e sperimentare la grafica, dovrà studiarsi le classi predefinite che permettono di visualizzare oggetti grafici e l'interazione con tastiera e/o mouse.

L'interfaccia grafica dovrebbe essere una sorta di strato di software che gestisce l'interazione con l'utente e genera gli stessi messaggi che nel progetto non grafico vengono inviati alle classi che gestiscono lo stato del gioco.

Quindi, dovrebbe avere una certa *indipendenza* dalle classi che gestiscono lo stato del gioco e verificano la legalità delle mosse, e comunicare con esse attraverso invocazioni di metodi (in entrambe le direzioni).

Importante

L'eventuale svolgimento delle parti facoltative, ovviamente, sarà tenuto in considerazione, ma la valutazione del progetto dipenderà prevalentemente dall'eleganza con cui viene strutturato e scritto il codice, oltre che dalla consapevolezza con cui sapete difendere le vostre scelte progettuali all'orale.