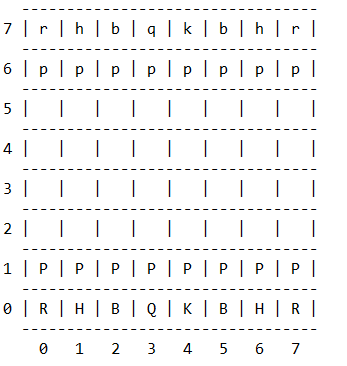
Relazione Progetto: Chess 1v1

# Introduzione

Il gioco degli scacchi consiste nel provocare scacco matto all’avversario utilizzando i pezzi a disposizione e mangiando quelli avversari qualora fosse necessario. Questo progetto tratta la realizzazione di un programma che funge da interfaccia per giocare a Scacchi uno contro uno.

# La scacchiera

La scacchiera del programma é una matrice 8x8 di Square, quest’ultima classe identifica il singolo quadrato e permette di ospitare un singolo pezzo sopra di essa, si è deciso di effettuare questa scelta dato che vi sono quadrati di colore diverso all’interno della scacchiera. La scacchiera è visualizzabile sulla console attraverso la stampa della funzione toString() in cui insiemi di righe, spazi, lettere e numeri permettono ai giocatori di vederla nel modo più semplice possibile. L’immagine sottostante riporta l’output della funzione toString() dopo l’inizializzazione della scacchiera:

Come detto precedentemente, la classe della scacchiera comprende anche il metodo initialize() che, come dice il nome, la inizializza inserendo all’interno dei quadrati delle prime e ultime due righe seguendo la configurazione classica degli scacchi, visibile nell’immagine a sinistra.

I singoli pezzi vengono inseriti singolarmente sulla scacchiera attraverso il metodo newPiece() che permette l’inserimento dell’elemento sul singolo quadrato sfruttando il campo di classe Position del primo. Questa scelta è data dal fatto che, oltre a evitare ripetizioni nel codice, il pedone a “fine corsa” viene promosso diventando un nuovo pezzo (a discrezione del giocatore).

Il metodo put() è il movimento effettivo dei pezzi sulla scacchiera e funziona prendendo il pezzo dalla prima posizione data come parametro e spostandolo nella seconda, gli eventuali controlli da fare verranno effettuanti dai metodi dei pezzi. Il motivo di questa scelta è dato

È anche possibile caricare/salvare una partita già iniziata rispettivamente con metodi loadMatch() e saveMatch che leggono/scrivono in un file di testo le informazioni relative ai pezzi.

Per evitare un eventuale “suicidio” del re, il metodo statico ChessTableSupport permette di eseguire eventuali controlli sulla mossa

# I pezzi della scacchiera

Le classi dei pezzi della scacchiera ereditano tutte dalla classe Element, poiché hanno caratteristiche comuni quali mangiare un altro pezzo o muoversi verso un'altra posizione. Per muoversi, i pezzi riempiono un ArrayList con delle possibili posizioni valide in cui può andare a partire dalla posizione in cui è il pezzo (il metodo d’inserimento varia da pezzo a pezzo). Per indicare se una possibile posizione finale è valida, si verifica se è contenuta nell’ArrayList e, nel caso lo sia, si passa al movimento vero e proprio richiamando il metodo put() spiegato in precedenza. Ci possono essere più casi nel movimento:

* Posizione libera: il pezzo trova la posizione finale libera e quindi si sposta normalmente;
* Posizione occupata: quando il pezzo trova la posizione finale occupata, controlla se è mangiabile attraverso il metodo eatable() e, dal risultato ottenuto, sceglierà se mangiarlo o meno.
* Presa al varco (solo Pedone): quando l’elemento é affiancato da un pedone, è possibile spostarsi in diagonale e mangiarlo se le condizioni per effettuare questa mossa vengono soddisfatte (vedi documentazione);
* Arrocco (Solo re): è una mossa che permette al re di evitare eventuali scacchi da parte dei pezzi avversari, le condizioni dell’arrocco vengono verificate dalla funzione canCastle().

La scelta di far eseguire i controlli direttamente ai pezzi è dovuta al fatto di non voler appesantire troppo la scacchiera.

Il re ha ulteriori funzioni che permettono di verificare se è sotto scacco oppure è presente uno scacco matto: nel caso si verifichi il primo, si stampa un avviso sulla console mentre quest’ultimo porta alla vittoria del giocatore che l’ha effettuato.

# La Console

La console è il vero motore di gioco della scacchiera ed è divisa in diverse fasi ben specifiche, la prima fase consiste nello scegliere tra iniziare una nuova partita (richiamando initialize()) oppure caricarne una già iniziata (richiamando loadMatch()), in seguito si passa alla fase di movimento: il giocatore che deve effettuare il turno può scegliere la posizione di un suo elemento oppure il salvataggio della partita corrente. Se il giocatore sceglie la prima opzione, passerà alla fase successiva: la scelta della posizione finale che porterà infine al movimento effettivo e passando il turno all’avversario.