Progetto Programmazione II

Il gioco della dama, basato sulle regole della dama italiana

Suddivisione delle classi

- controller
  - Controllore
- model
  - Cas
  - Pannello
  - PedModel
- view
  - Casella
  - Damone
  - PedView
  - Scacchiera
- Main
  - Main

Package controller

Una sola classe: Controllore.java. È il programma vero e proprio, fa comunicare il model e il view.
Variabili di istanza:

- *pedine_mangiate*: di tipo *int*, statica; tiene il conto di quante pedine ha mangiato il computer, per implementare la mangiata multipla.
- *mangiato*: di tipo *boolean*, statica; per sapere se il computer ha mangiato o meno.
- *x*: di tipo *int*, statica; la coordinata x della casella dove l'utente clicca per spostare la pedina.
- *y*: di tipo *int*, statica; come sopra, solo che è la coordinata y.
- *prima_volta*: di tipo *boolean*; inizializzata a *true*, serve per aggirare i controlli alla prima mossa della partita.

Metodi:

- *public static void* ***controlli****(int, int)*: riceve come parametri le coordinate, chiama il metodo *iniz(int, int)* per inizializzare le variabili d'istanza, poi, se *prima*_volta è *false*, chiama il metodo *valido()*; se questo ritorna *true* chiama il metodo *spostamento()* per muovere la pedina. Se *prima_volta* è *true* chiama direttamente *spostamento()*.
- *public static boolean* ***valido****()*: controllo che non l'utente non clicchi sulla stessa colonna della pedina e, o se ha cliccato per mangiare (*possoMangiare()*) , o se ha cliccato su una casella vuota vicina (*casellaVuota()*); ritorna *true* o *false* a seconda se il controllo ha successo o meno.
- *private static boolean* ***possoMangiare****()*: controlla se la mossa dell'utente per mangiare è valida o meno. Per prima cosa fa un controllo sulla mangiata avanti a destra: se si sposta di 2, e se vicino c'è una pedina dell'altro colore (nero, perché l'utente è bianco); se c'è una pedina singola (3) può mangiare, se c'è una dama (5) bisogna controllare se la pedina che vuole muovere l'utente e anch'essa una dama

(4)[1]. Dopodiché fa l'equivalente a sinistra. Poi controlla se la pedina che vuole muovere l'utente è una dama, in modo da vedere se può mangiare all'indietro.

- *private static boolean* ***possoMangiareVal****()*: controlla se l'utente può mangiare ancora con la stessa pedina; in caso contrario il turno passa al computer.
- *private static boolean* ***devoMangiare****()*: controlla per ogni pedina dell'utente se può mangiare; se è così e l'utente non può fare nessuna mossa tranne quella di mangiare. È implementato con due *for each* per scorrere la matrice della scacchiera e una serie di *if* per controllare gli spostamenti.
- *private static boolean* ***casellaVuota****()*: controlla se le caselle vicino sono vuote; se la pedina è una dama controlla anche dietro.
- *private static void* ***spostamento****()*: usa una variabile booleana interna *controllo_mangiato* che viene messa a *true* se l'utente tenta di spostarsi mentre potrebbe mangiare, in modo tale da impedire questo spostamento. Questo metodo fa tutta una serie di *if* sulle coordinate della pedina dell'utente e della casella su cui clicca, in modo da capire se lo spostamento è valido, se si è spostato per mangiare, se si è spostato avanti o indietro nel caso sia una dama, se lo spostamento è all'interno di una mangiata multipla, dopodichè controlla se l'utente ha vinto chiamando il metodo *haiVinto()*: se ritorna *true*, apre una finestra con scritto *"Hai vinto"*, altrimenti chiama il metodo *mossaIA()* che implementa la mossa del computer, e dopo chiama il metodo *haiPerso()* per controllare che abbia vinto il computer; se ritorna *true*, apre una finestra con scritto *"Hai perso"*. Altrimenti crea una nuova scacchiera con le mosse aggiornate.
- *private static boolean* ***haiPerso****()*: controlla se sulla scacchiera ci sono solo pedine nere.
- *private static boolean* ***haiVinto****()*: controlla se sulla scacchiera ci sono solo pedine bianche.
- *private static void* ***mossaIA****()*: si occupa dello spostamento della pedina del computer. Usa un array di quattro interi inizializzati a -1, che rappresentano le coordinate della cella dove era inizialmente la pedina (*array[0]* e *array[1]*) e le coordinate della casella dove mi sposto (*array[2]* e *array[3]*). Per prima cosa inizializza *mangiato* a *false* e *pedine_mangiate* a 0. Poi utilizza un *do while* per implementare la mangiata multipla: infatti il primo controllo che fa all'interno è su *pedine_mangiate* che sia maggiore di 0. Se questo controllo da esito *true* chiama il metodo *mangiata_multipla()*, altrimenti chiama *possoMangiareIA()*. Una volta che ha mangiato controlla dove ha mangiato (a sinistra o a destra) e aggiorna la posizione delle pedine, poi controlla se può mangiare ancora (mangiata multipla) e se può lo fa e aggiorna la posizione delle pedine. Se invece non può mangiare chiama *spostamentoIA(int[])* e tenta di spostarsi; se può aggiorna la posizione delle pedine, altrimenti se non si può spostare apre una finestra con scritto *"Hai vinto"*.
- *private static void* ***mangiata_multipla****(int, int, int[])*: permette all'utente di continuare finchè può; il tutto è realizzato con una serie di *if*.
- *private static void* ***spostamentoIA****(int[] array)*: utilizza una variabile booleana interna *mangiato* per memorizzare se si è spostato o meno. Se si può spostare modifica l'array in ingresso con le nuove posizioni delle pedine. Per prima cosa controlla se si può spostare senza venire mangiato; se tutti i suoi possibili spostamenti portano a una situazione di perdita della pedina, allora sceglie uno qualsiasi dei possibili spostamenti.
- *private static void* ***possoMangiareIA****(int[])*: con una serie di *if* per i controlli in cui la pedina è vicino al bordo controlla se il computer può mangiare e in caso affermativo cambia l'array delle coordinate con quelle nuove, poi sarà *spostamentoIA()* che si occupa dell'effettivo spostamento.

[1] I numeri indicano le varie componenti che ci sono sulla scacchiera: 0=casella vuota bianca, 1=casella vuota nera, 2=pedina bianca, 3=pedina nera, 4=dama bianca, 5=dama nera. Questi riferimenti sono utilizzati nella classe Pannello e nella classe PedModel del package model.

- *public static int **getMyX()***: restituisce la coordinata *x* della casella su cui ha cliccato l'utente;
- *public static int **getMyY()***: restituisce la coordinata *y* della casella su cui ha cliccato l'utente;
- *public static void **iniz**(int, int)*: inizializza le variabili di istanza *x* e *y*.

Package model

Tre classi: Cas.java, Pannello.java e PedModel.java. È la parte logica del programma.

Cas.java

È la rappresentazione logica della casella.
Variabili di istanza:

- *x*: di tipo *int*, statica; è la coordinata x delle caselle della scacchiera.
- *y*: di tipo *int*, statica; è la coordinata y delle caselle della scacchiera.

Metodi:

- *public **Cas**(int, int)*: il costruttore, inizializza le variabili di istanza con i valori passati.
- *public int **getMyX()***: restituisce la coordinata *x* della casella.
- *public int **getMyY()***: restituisce la coordinata *y* della casella.

Pannello.java

È la rappresentazione logica della scacchiera.
Variabili d'istanza:

- *matrice*: di tipo *PedModel[][]*, statica; è una matrice che rappresenta la disposizione delle pedine e delle caselle sulla scacchiera.
- *conta_utente*: di tipo *int*, statica; tiene il conto di quante volte consecutive ha mangiato l'utente.
- *cordXvecchie*: di tipo *int*, statica; memorizza la coordinata x della casella su cui l'utente si è spostato dopo aver mangiato. Serve per implementare la mangiatamultipla.
- *cordXvecchie*: di tipo *int*, statica; come sopra, solo per la y.
- *mangiato_utente*: di tipo *boolean*, statica; per sapere se l'utente ha mangiato, serve per implementare la mangiata multipla.
- *pan*: di tipo *Pannello*, statica; è il pannello che usa la classe *Controllore*.

Metodi:

- *public **Pannello()***: costruttore senza parametri, inizializza le variabili d'istanza e chiama il metodo *creaScacchiera()*.
- *Public **Pannello**(PedModel[][])*: costruttore con parametri, riceve la matrice delle pedine; inizializza la variabile d'istanza *matrice* e chiama *creaScacchiera()*.
- *private void **creaScacchiera()***: dispone le pedine e le caselle vuote sulla scacchiera.

PedModel.java

Si occupa di indicare se una cella della scacchiera contiene una pedina, una dama o se è vuota, e il colore.
Variabili d'istanza:

- *x*: di tipo *int*; è la coordinata x delle cella delle scacchiera.
- *y*: di tipo *int*; è la coordinata y delle cella delle scacchiera.
- *valore*: di tipo *int*; 1=casella vuota nera, 2=pedina bianca, 3=pedina nera, 4=dama bianca, 5=dama nera.
- *xcomune*: di tipo *int*, statica; è la coordinata x della pedina su cui ha cliccato l'utente.
- *ycomune*: di tipo *int*, statica; è la coordinata y della pedina su cui ha cliccato l'utente.

Metodi:

- *public **PedModel**(int, int, int)*: il costruttore, inizializza le variabili d'istanza.
- *public int **getMyX()***: restituisce la coordinata *x* della casella.

- *public int **getMyY()***: restituisce la coordinata *y* della casella.
- *public int **getValore()***: restutuisce la variabile d'istanza *valore*.

Package view

Quattro classi: Casella.java, Damone.java, PedView.java, Scacchiera.java. È l'interfaccia grafica vera e propria.

Casella.java

Rappresenta la casella vuota nera, estende la classe *JButton*.
Variabili di istanza:

- *x*: di tipo *int*; è la coordinata x della casella.
- *y*: di tipo *int*; è la coordinata y della casella.

Metodi:

- *public **Casella**(int, int)*: il cotruttore; inizializza le variabili d'istanza con i valori passati, imposta il colore dello sfondo, utilizza i metodi *setBorderPainted(false)* e *setFocusPainted(false)* per eliminare gli effetti grafici tipici dei *JButton*. Poi con l'*actionListener* inizializza le variabili di *Cas* e chiama il metodo di *Controllore controlli(int, int)* passandogli le sue coordinate, ottenute con *getMyX()* e *getMyY()*.
- *public int **getMyX()***: restituisce la coordinata *x* della casella.
- *public int **getMyY()***: restituisce la coordinata *y* della casella.

Damone.java

Rappresenta la dama, chiamata "damone" per non confondere la classe con il nome del progetto (*Dama*).
Estende la classe *PedView.*
Variabili di istanza: nessuna.
Metodi:

- *public **Damone**(int, int, boolean)*: il costruttore; chiama il metodo della superclasse, successivamente fa un controllo sul colore per decidere che immagine impostare, se la dama nera o la dama bianca.

PedView.java

Rappresenta la pedina, estende la classe *JButton*.
Variabili d'istanza:

- *x*: di tipo *int*; è la coordinata x della pedina.
- *y*: di tipo *int*; è la coordinata y della pedina.

Metodi:

- *public **PedView**(int, int, boolean)*: il costruttore, inizializza le variabili d'istanza, in più prende un booleano per il colore della pedina (*true* = bianco, *false* = nero); imposta anche il colore dello sfondo, cioè grigio perché le pedine si possono spostare solo sulle caselle scure. Utilizza i metodi *setBorderPainted(false)* e *setFocusPainted(false)* per eliminare gli effetti grafici tipici dei *JButton*. Con l'*actionListener* inizializza le variabili di *PedModel* con le sue coordinate ottenute con *getMyX()* e *getMyY()*.
- *public int **getMyX()***: restituisce la coordinata *x* della casella.
- *public int **getMyY()***: restituisce la coordinata *y* della casella.

Scacchiera.java

Rappresenta la scacchiera dal punto di vista grafico, estende la classe *JPanel*.
Variabili di istanza:

- *matrice*: di tipo *JButton[][]*; matrice che rappresenta la posizione delle pedine e delle caselle scure sulla schacchiera.

- *matricePanel*: di tipo *JPanel[][]*: matrice che rappresenta la disposizione delle caselle bianche.
- *chequer*: di tipo *Scacchiera*; è la scacchiera utilizzata nel *main*.

Metodi:

- *public* ***Scacchiera****()*: costruttore senza parametri, viene eseguito solo la prima volta; imposta il layout come un *GridLayout*, inizializza le variabili di istanza (le crea), e chiama i metodi *inizializzaMatrice()*, *inizializzaPanel()*, e *creaScacchiera()*.
- *public* ***Scacchiera****(int n)*: costruttore che riceve come parametro un intero, per distinguerlo dall'altro, dato che questo viene chiamato dalla seconda volta in poi.
- *private void* ***creaScacchiera****()*:dispone sulla scacchiera le pedine e le caselle vuote a seconda di che posizione occupano in *matrice*.
- *private void* ***inizializzaMatrice****()*: modifica *matrice* a seconda di come si spostano le pedine.
- *private void* ***inizializzaPanel****()*: dispone le caselle bianche sulla scacchiera.
- *private JPanel* ***aggiungiSpazio****()*: utilizzato in *inizializzaPanel()*, crea un *JPanel* vuoto con sfondo bianco.
- *private PedView* ***creaPedinaBianca****(int, int)*: utilizzato in *inizializzaMatrice()*, crea una pedina (*PedView*) bianca.
- *private Casella creaSpazio(int i, int j)*: utilizzato in *inizializzaMatrice()*, crea una casella scura vuota.
- *private PedView* ***creaPedinaNera****(int, int)*: utilizzato in *inizializzaMatrice()*, crea una pedina (*PedView*) nera.
- *public void* ***creaPannello****()*: crea la nuova scacchiera con le coordinate di *matrice*.

Package Main

Una sola classe: Main.java.

Metodi oltre al main:

- *public static void* ***creaFinestra****()*: crea la finestra; la divide in due parti con il *BorderLayout*: al centro c'è la scacchiera, a sud c'è un *JPanel*, con un *JButton* per abbandonare la partita.
- *public static JPanel* ***createSouth****()*: crea il *JPanel* con il *JButton* per arrendersi.

Problemi riscontrati

- Mancata implementazione della mangiata obbligatoria in base al numero di pedine e al loro valore.
- Crea una nuova finestra ad ogni mossa, non crea molti problemi alla giocabilità, però non siamo riusciti a fare funzionare a dovere il nostro metodo perché c'erano dei problemi con la remove() e la removeAll(). In teoria bastava rimuovere tutte le pedine e poi reinserirle usando come base la matrice modificata dopo le mosse, però siamo rimasti bloccati dai due metodi appena citati.
- Ho un costruttore "particolare" che mi richiede come parametro qualcosa che non uso, questo è servito solo per aggirare il problema precedente che non riusciva a creare il pannello, così ne creiamo uno nuovo ogni volta, che non deve avere come costruttore quello iniziale, perché se no mi viene inizializzata la Scacchiera sempre allo stesso modo, e sarebbe come non muoversi, quindi me la inizializzo io con i valori dopo gli spostamenti.
- Mancata implementazione della partita dichiarata finita in parità nel caso che le posizioni dei pezzi sulla damiera si ripetono per 3 volte durante il match, anche se non consecutivamente.