Gruppo AM22

Note UML

Nell'implementazione scelta del MVC il model e la view non interagiscono direttamente ma sempre tramite il controller.
La classe Game implementa l'interfaccia "Model_Controller" in cui sono presenti i metodi invocati dal controller per interagire con il model.

La **classe Game** contiene al suo interno tutti i dati della partita corrente: i mazzi con le carte, i giocatori, il giocatore corrente, i punteggi e gli obiettivi comuni.
Per andare ad aggiornare (o ritornare al controller) i valori di questi attributi Game invoca a sua volta i metodi implementati dalle singole classi. Ad esempio quando il controller invoca un metodo di Game per far scartare una carta a un dato giocatore, Game a sua volta invoca un metodo di Player che andrà effettivamente a posizionare la carta.
Metodi presenti:

- SetNomeAttributo: assegna all'attributo private il valore passato come parametro;
- ShowNomeAttributo: ritorna il valore dell'attributo privato;
- AcceptPlayer(int i); accetta massimo 4 giocatori assegnandogli un codice identificativo (0,1,2,3);
- SetPlayersDefinitive(): fissa il numero definitive di giocatori (tra 2 e 4);
- PreGame(): crea i mazzi e li mescola, crea la scoreboard, assegna i colori, calcola il primo giocatore;
- PlaceFirstCard(MCard c, int i, boolean f): metodo chiamato per posizionare la carta iniziale, il booleano indica il lato della carta scelto;
- GiveCard(int i): distribuisce le carte iniziali;
- StartGame(): viene utilizzato per richiedere l'inizio del gioco. Controlla che tutto (giocatori, carte) sia stato creato, che siano distribuite e posizionate le carte iniziali, che tutti abbiano gli obiettivi comuni e segreti;
- PlayTheCard(int i, DrawCard c, int x, int y, boolean f): piazza la carta c del giocatore i nella posizione xy indicata, mostrando il lato f. Restituisce 0 se è andato a buon fine;
- CheckDeck(): controlla che entrambi i mazzi non siano empty;
- ChooseCard(int i, DrawCard c): per pescare la carta c;
- EndGame(): controlla che tutti i giocatori abbiano fatto il turno finale e decreta la fine della partita;
- FinalScore(): restituisce il vincitore, per vedere i punti poi basta fare ShowScoreBoard();

La **classe Player** contiene i dati del singolo giocatore: il numero identificativo (IDnumber), il colore (playerColor), le carte già scartate (in una matrice dinamica (ArrayList di ArrayList) di Cell), il punteggio (score), le carte in mano (cardsInHand), l'obiettivo segreto (secretGoal), un contatore del numero di risorse e oggetti posseduti (resAndObj) e il numero di righe e colonne (rows e columns) della matrice delle carte scartate.
Metodi presenti:

- setNomeAttributo: assegna all'attributo private il valore passato come parametro;
- getNomeAttributo: ritorna il valore dell'attributo privato;
- setCardsInHand(DrawCard card): aggiunge la carta card alle carte in mano (cardsInHand);
- placeCard(MCard card, int x, int y, boolean side): scarta la carta card aggiungendola alla matrice playerCards nella posizione indicata (riga x e colonna y) e rimuovendola dalle carte in

mano. Se necessario, aumenta le dimensioni della matrice per collegare la carta a quelle già presenti. Aggiorna infine il punteggio, il numero di risorse e oggetti posseduti dal giocatore e ritorna 0. Se la carta non può essere aggiunta (posizione non ammissibile o già occupata), ritorna -1 al chiamante;

- haveCard(DrawCard card): ritorna 0 se la carta card è presente tra le carte in mano del giocatore, -1 altrimenti;
- addColumnRight(): aggiunge una colonna a playerCards in posizione columns;
- addColumnLeft(): aggiunge una colonna a playerCards in posizione 0;
- addRowUp(): aggiunge una riga a playerCards in posizione 0;
- addRowDown(): aggiunge una riga a playerCards in posizione rows.

Le istanze delle **classe Cell** costituiscono gli elementi della matrice dinamica in cui sono tenute le carte scartate dal giocatore. Tra i suoi attributi ci sono, oltre alla carta, due booleani che contengono informazioni circa la disponibilità della cella.

La **classe CountRO** contiene il numero di risorse e oggetti del singolo giocatore e i metodi per aggiornare tali valori alla fine del turno del giocatore stesso in base alla matrice delle carte e all'ultima carta posizionata.

La **classe Deck (e le sue sottoclassi)** contengono le carte divise nei vari mazzi e i metodi per gestirli.
Quando viene creata un'istanza di una sottoclasse di Deck (a inizio partita), l'ArrayList deck (attributo) viene popolato con le carte opportune a seconda del tipo di mazzo.
Deck e le sue sottoclassi contengono metodi per mescolare le carte, sapere se il mazzo è vuoto e rimuovere una carta. Inoltre, ShownCards rappresenta le 4 carte visibili da cui è possibile pescare una carta e contiene il metodo placeCard(int i, DrawCards) che serve per riposizionare la quarta carta dopo averne pescata una dal tavolo.

La **Classe GoalCard** rappresenta le carte obiettivo e contiene i metodi showColor(), showContent(), showScore() che permettono di mostrare rispettivamente il colore della carta, il tipo di obiettivo e il punteggio che si ottiene.

La **Classe MCard** rappresenta tutte le altre carte e contiene i metodi per:

- getSide() e setSide() : per impostare e sapere il verso della carta;
- getCount() e setCount() : per impostare e sapere se il punteggio della carta è già stato contato;
- visibleAngle() e changeAngle(int n) : per sapere se l'angolo è visibile e poter coprirlo;
- getElementCorner(int num) : ritorna un intero corrispondente ad un determinato elemento presente nell'angolo num della carta;
- isStartingCard() : per sapere se è una carta iniziale;
- getCornersCovered() e setCornersCovered() : per sapere quanti angoli copre la carta.

La **Classe DrawCard** è sottoclasse di MCard e mi permette di sapere se la carta è una carta risorsa, inoltre mi permette di sapere il colore e il punteggio.

La **Classe GoldCard** è sottoclasse di DrawCard e rappresenta la carte oro, mi permette di sapere il numero e il tipo di condizioni che il giocatore deve avere per poter posizionare la carta.

Ogni carta (eccetto le carte obiettivo) presenta un attributo di **classe Front** (FGold per la carte oro) e uno di **classe Back** con le relative informazioni (angoli, eventuali condizioni per il posizionamento e punteggi). Quando la carta deve essere posizionata viene segnato il lato visibile all'utente tramite il valore dell'attributo side di MCard.
Per le carte obiettivo non è stata fatta la distinzione tra fronte e retro perché il retro non contiene informazioni rilevanti per il gioco, tutti i dati e i metodi per gestirli sono contenuti nella classe GoalCard.

La **classe Angle** contiene i dati relativi al singolo angolo della carta: i boolean visibility e covered indicano, rispettivamente, la visibilità dell'angolo (data come caratteristica della carta) e il fatto che sia coperto o meno (da un'altra carta posizionata su tale angolo), res e obj indicano se in tale angolo siano presenti o meno una risorsa o un oggetto mentre numangle indica quale dei quattro angoli della carta rappresenta (in alto o in basso a destra oppure a sinistra).
Per quanto riguarda i metodi:

- isCovered() viene chiamato nel momento in cui, nel caso in cui l'angolo venisse coperto da una carta posizionata dal giocatore, modifica covered in 1 (== "angolo coperto");
- getVisibility() è un metodo che ritorna un booleano che indica se l'angolo sia visibile o meno;
- getElement() è un metodo che ritorna l'oggetto presente nell'angolo (che potrebbe essere res o obj).

La **classe Back** rappresenta il retro delle carte: come attributo ha un array che riporta i quattro diversi angoli mentre come metodi:

- getVisibility è un metodo che prende in ingresso un intero che rappresenta l'angolo e ne ritorna la visibilità;
- coverAngle è un metodo che prende in ingresso un intero che rappresenta l'angolo va a modificare l'attributo covered di Angle.

La **sottoclasse Bstarting** aggiunge come attributo un array che rappresenta le tre risorse centrali che la carta Starting potrebbe presentare e un metodo getCenter che ritorna tale array.

La **classe Front** rappresenta il fronte delle carte: come attributi possiede un array che riporta i quattro angoli (di classe Angle), lo score della carta e startingCard che indica se tale carta sia una carta iniziale o meno.
Come metodi :

- isStartingCard ritorna un boolean che indica se la carta sia iniziale o no;
- getVisibility e coverAngle hanno la stessa funzione dei metodi omonimi della classe Back;
- getElementAngle prende in ingresso il numero che si riferisce all'angolo e ritorna l'oggetto presente in esso;
- getScore ritorna l'attributo score del fronte.

La **sottoclasse FGold** aggiunge come attributi la condizione condition affinché la carta possa essere posizionata (che è un array che può contenere al massimo 5 Resource) e l'attributo detail che riporta la condizione per ottenere i punti dello score.
Per quanto riguarda i metodi, returnCondition e returnDetail ritornano condition e detail.