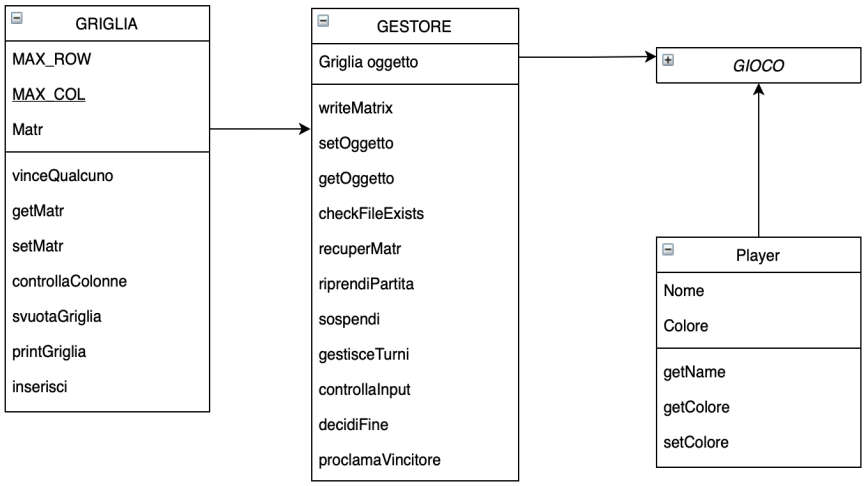
Progetto Forza 4

Federico Iltchev - 1842707

Settembre 2021

1 Introduzione

Forza4 è un gioco a cui si partecipa in due. Si gioca su una griglia, cioè una matrice composta da 6 righe e 7 colonne, in cui si infilano pedine dall'alto. Dopo aver scelto il colore delle pedine, il giocatore si alterna in turni per inserirle nella griglia. Lo scopo del giocatore è riuscire ad allineare in orizzontale, verticale e diagonale 4 pedine.



2 Descrizione delle Classi

Gestore

La classe Gestore si occupa di gestire le logiche del gioco. Essa ha come attributi un oggetto **Griglia**, che utilizzamo come piattaforma del gioco.
Il metodo **getOggetto** restituisce l'oggetto Griglia, utile ad esempio se vogliamo recuperarla per sospendere la partita.
Il metodo **setOggetto** permette di modificare la piattaforma a esecuzione in corso, nel caso in cui un giocatore decida di sospendere una partita e cominciarne una nuova.
Il metodo **gestisceTurni** permette di alternare i due giocatori, specialemnte nel caso di caricamento di una partita salvata.
Il metodo **controllaInput** controlla che l'input sia un intero, e non qualsiasi altro tipo.
Il metodo **checkFileExists** controlla se esiste un file con il nome che gli viene dato in input, nella cartella in cui vogliamo salvarlo.
Il metodo **writeMatrix** scrive la matrice, o Griglia, nel file che diamo in input alla classe. Per gestire i possibili file con nomi uguali, li salva in un formato creato appositamente(.fdf).
Il metodo **recuperMatr** importa la matrice,o Griglia, da un file e la da in output.
Il metodo **sospendi** salva la partita in un file fdf. Utilizza il metodo *checkFileExists* per controllare se esistono file con lo stesso nome e la stessa estensione. Crea quindi un file e utilizza il metodo *writeMatrix* per salvare la Griglia nel file.
il metodo **caricaPartita** permette di continuare una partita sospesa. Utilizza il metodo *checkFileExists* per controllare se esistono file con il nome che cerchiamo e il metodo *recuperaMatr* per estrarre la matrice dal file.
Il metodo **gestisciPlay** permette infine di gestire i valori interi che possono essere attribuiti alla variabile play durante la partita. Se entrambi i giocatori hanno effettuato almeno una mossa, con l'aiuto del metodo *sospendi* possiamo mettere in pausa una partita e salvarla in un file. Se vorremo continuare a giocare, utilizza il metodo svuotaGriglia della classe Griglia per svuotare la piattaforma di gioco e azzerare i turni, cominciando così una nuova partita.
Il metodo **decidiFine** gestisce la possibilità di cominciare una nuova partita. Con il meotodo *svuotaGriglia* resetta la piattaforma di Gioco e permette di cominciare una nuova partita.
il metodo **proclamaVincitore** stampa a video il vincitore.

Griglia

Gestisce la piattaforma su cui avviene il gioco.
Essa contiene i metodi per la gestione della matrice. il costruttore inizializza una matrice vuota. il metodo **getMatr** permette di ottenre la matrice.

il metodo **setMatr** permettere di accedere alla matrice, di modificarla e sovrascriverla.
il metodo **vinceQualcuno** controlla che un giocatore abbia allineato le 4 pedine nelle varie direnìzioni.
Il metodo **Inserisci** prende la colonna in input e inserisce la pedina del giocatore che sta effettuando il proprio turno.
Il metodo **printGriglia**,presa in input una Griglia,la stampa su terminale, riga per riga con caratteri tabulari
Il metodo **controllaColonna** controlla che il numero di colonna inserita non sia più grande o più piccolo dell'effettivo numero di colonne esistenti.
Il metodo **svuotaGriglia** è molto utile per svuotare la matrice dalle pedine.

Gioco

La classe che utilizziamo per l'esecuzione del Gioco, in cui raccordiamo la classe Gestore con la classe Player.

Player

Rappresenta il giocatore, attraverso i due attributi che lo distinguono: il nome e il colore.
Il costruttore istanzia un oggetto di tipo giocatore con il nome che gli viene fornito in input.
Il metodo **getNome** ritorna il valore nell'attributo nome dell'oggetto Player.
Il metodo **getColore** ritorna il valore nell'attributo colore dell'oggetto Player.
Il metodo **setColore** permette di inserire o modificare il colore del singolo Player.

3 Descrizione delle funzionalità

Assegnazione colori

All'interno della classe Gioco, creiamo due Player chiedendogli di inserire il Nome. Al primo assegnamo il colore Rosso, con una pedina "R", al secondo il colore Blu, con una pedina "B".

```
Inserire il nome del Giocatore 1:
Andrea
Inserire il nome del Giocatore 2:
Francesco
```

Alternanza turni

Il metodo **gestisceTurni** della classe GESTORE si occupa di tenere il conto del turno a cui siamo arrivati, contando ogni volta il numero di pedine inserite.
Per gestire invece la scelta del giocatore a cui tocca, a inizio turno controllo che il turno sia pari o dispari: se pari toccherà al Giocatore2, altrimenti al Giocatore1.

```
/* gestisce i turni */
if (turn%2==0) {
    playerColour = Gioc2.getColore();
    System.out.print(Gioc2.getNome() + ", colore " + Gioc2.getColore() + ", scegli una colonna o digita il codice '1234' per sospenderla: "
} else {
    playerColour = Gioc1.getColore();
    System.out.print(Gioc1.getNome() + ", colore " + Gioc1.getColore() + ", scegli una colonna o digita il codice '1234' per sospenderla: "
}
```

Stampa Griglia e messaggio

Il metodo **printGriglia** si occupa di stampare a video la piattaforma del Gioco, con la parte superiore aperta per far capir eche viene inserita da sopra la pedina. Al di sotto della Griglia viene invece stampato un messaggio, il quale ricorda all'utente la sua pedina, e gli da la possibilità di sospendere la partita. Quest'ultima solo nel caso in cui entrambi i giocatori abbiano effettuato almeno una mossa.

```
| | | | | | | |
---------------
| | | | | | | |
---------------
| | | | | | | |
---------------
| | | | | | | |
---------------
| | | | | | | |
---------------
| | | | | | | |
---------------
<------------->
 1 2 3 4 5 6 7

1, colore R, scegli una colonna o digita il codice '1234' per sospenderla:
```

Rileva vittoria o parità

Il metodo **vinceQualcuno** ogni fine turno controlla nelle 4 direzioni (verticale,orizzontale, entrambi le diagonali) che vi siano 4 pedine allineate. Se restituisce true, un giocatore avrà vinto.

```
public static boolean vinceQualcuno(char player, char[][] matr){
    boolean var = false;
    //controlla a destra
    for (int row = 0; row<MAX_ROW; row++){
        for (int col = 0;col < MAX_COL - 3;col++){
            if (matr[row][col] == player   && matr[row][col+1] == player &&matr[row][col+2] == player &&matr[row][col+3] == player){
                var = true;
                return var;
            }
        }
    }
```

La variabile `winner` all'interno della classe Gioco contiene il risultato del suddetto metodo.
In caso di vittoria, verrà invocato il metodo **proclamaVincitore** della classe Gestore che stamperà una stringa acclamando il vincitore.
Nel caso invece in cui dopo tutti i turni non ve ne sia ancora uno, verrà stampata una stringa a indicare l'avvenuto pareggio!

Sospensione

Il metodo **sospendi** all'interno della classe Gestore permette di sospendere la partita. Dopo averci dato la possibilità di inserire il nome, crea un file nel quale salva la matrice. Il metodo **writeMatrix** si occuperà poi di scrivere la matrice nel file. Esso sarà salvato con un'estensione creata appositamente per il progetto, in modo da non creare confusione con eventuali altri file omonimi con diversa estensione.

Ripresa Partita Salvata

Il metodo **caricaPartita**, chiamato nella classe Gioco, ci permette di cominciare una partita salvata. Il metodo **recuperMatr** si occupa di riprendere la matrice salvata nel file. Il metodo **setMatr** si occuperà poi di inserirla come piattaforma corrente del gioco.