Progetto Forza 4

Metodologie di Programmazione (2021-2022), JAVA

Autori:

Dino Livio, 1844312

Daniele Federici, 2022227

Introduzione

Questo Progetto riguarda la creazione del famoso gioco “Forza 4” con Linguaggio Java. Come IDE abbiamo usato Eclipse con il relativo add-on WindowBuilder per la parte grafica GUI.

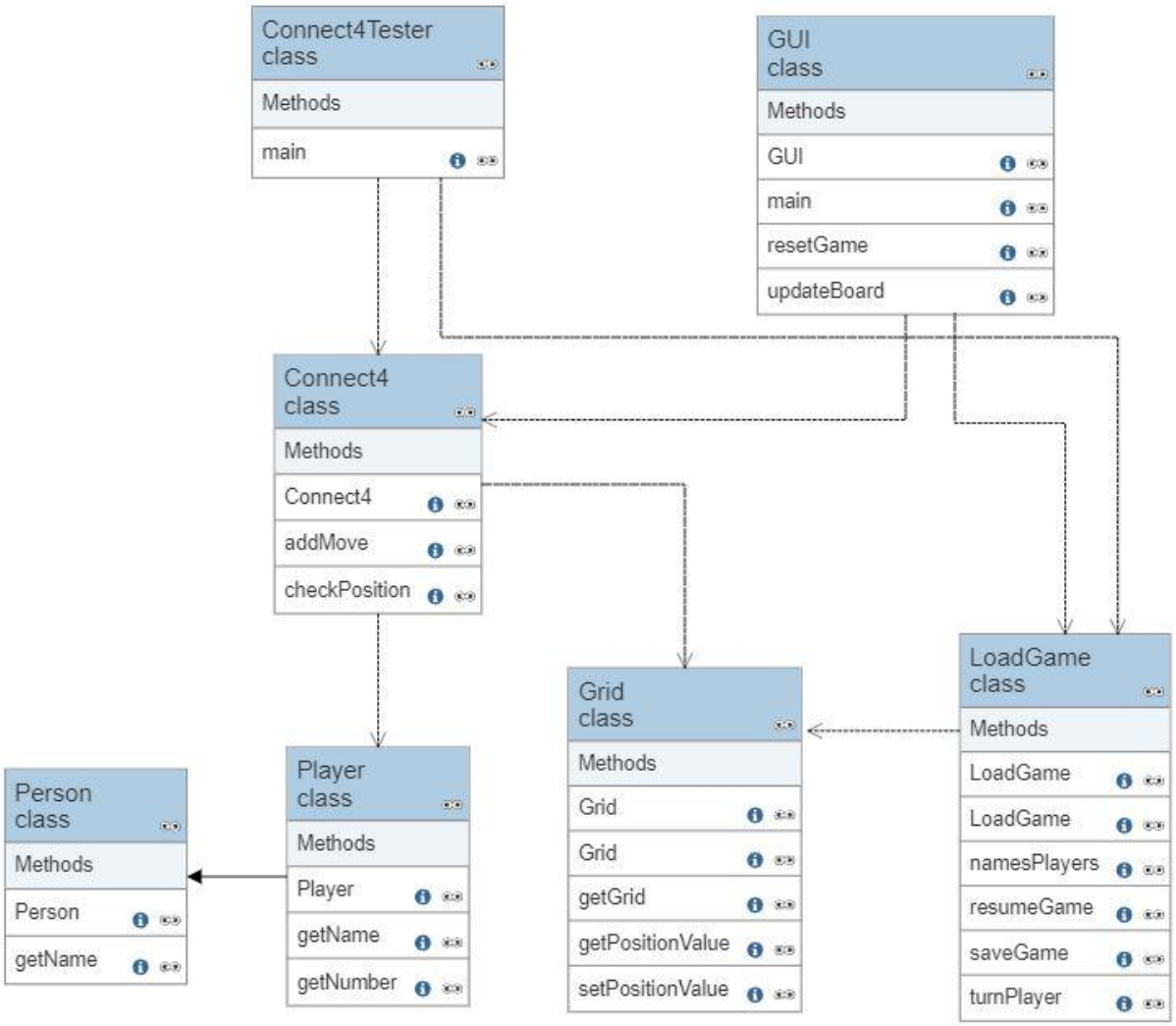
Nella fase iniziale del progetto, man mano che pensavamo le varie classi / metodi, abbiamo stampato l’output a video, questo per creare la struttura principale e per debug. La GUI è stata creata successivamente basata sul funzionamento corretto dei componenti creati.

Nel nostro gruppo di 2 persone, abbiamo organizzato lo sviluppo del programma in modo incrementale: Ogni X giorni (3-4 di solito) parlavamo dell’ architettura e idee di implementazione, assegnandoci il lavoro che ogni membro compiva nel periodo fino al prossimo incontro. La suddivisione delle task tra i membri viene spiegata più nel dettaglio nelle sezioni specifiche.

Descrizione delle classi

In questa classe descriviamo le classi utilizzate con i loro metodi e attributi. E anche incluso un diagramma UML per spiegare il rapporto tra le classi create.

Diagramma UML



Classe Connect 4

Descrizione:

Questa classe si occupa della logica principale del gioco, ovvero contiene metodi e attributi per l'avanzamento, il controllo e la fine del gioco.
Principi applicati: Class Cohesion, Minimizing Side Effects, Name Consistency

Attributi:

- ArrayList<Player> **giocatori**
  Un array di oggetti player usato per mantenere le informazioni dei giocatori (nome e colore).

- Griglia **grill**
  Un oggetto di gioco Griglia .

Metodi:

- **Connect4** (ArrayList<Player> players, Grid grid)
  Costruttore per la classe Forza4, crea un oggetto Forza 4 prendendo come argomenti oggetti (players) dalla classe Player, e (grid) dalla classe Grid.

- **addMove** (int col, int value, int turno)
  Argomenti: (col) colonna della griglia grid in cui inserire il colore (value).
  Da notare che il colore è espresso con un (value) intero per semplicità, ed ha value 1 o 2, nella GUI verrà convertito nel colore relativo al Giocatore 1 o Giocatore 2.
  L'intero (col) è accettato se compreso tra 0 e 6, ovvero il numero totale delle colonne della griglia.
  (turno) rappresenta il turno corrente del gioco, serve per chiamare checkPosition.
  Questo metodo semplicemente inserisce una pedina in una colonna scelta
  Tramite un ciclo while si controlla quanto la colonna (col) della griglia sia piena, in modo da inserire la pedina nella casella più in basso. Se la colonna è completamente piena, l'operazione di inserimento non viene eseguita (quindi è necessario chiamare il metodo su una colonna non piena).
  Una volta inserito il value nella griglia, chiama il metodo checkPosition.

- **checkPosition** (int row, int column, int turn)
  Argomenti: (row, column) coordinate della matrice in cui è stata posizionata l'ultima pedina, (turn) turno corrente del gioco.
  Controlla che lo stato della griglia di gioco sia in uno stato finale, ovvero che una verticale, orizzontale, diagonale o anti-diagonale abbiano 4 pedine consecutive dello stesso colore. Ritorna True in quest'ultimo caso, altrimenti False.

Classe Grid

Descrizione:

Classe dedicata alla griglia di gioco 6x7 di Forza 4
Principi applicati: Class Cohesion, Separating Accessors and Mutators, Minimizing Side Effects, Names Consistency

Attributi:

- int [][] **grid**
  La matrice di interi su cui si svolge il gioco.

Metodi:

- **Grid** ()
  Costruttore per la classe Griglia, semplicemente crea un oggetto grill come int[6][7]
- **Grid** (Grid g)
  Costruisce un oggetto grid inizializzandolo con un oggetto grid in input (usato al momento di caricare la partita)

- **getGrill** ()
  Stampa la matrice di interi.
  Utilizzato per la rappresentazione di output a video.

- **getPositionValue** (int row, int column)
  Argomenti: (row, column) rappresentano la casella della matrice grill
  Metodo ritorna il valore della casella corrispondente.

- **setPositionValue** (int row, int column, int value)
  Argomenti: (row, column) rappresentano la casella della matrice grill
  (value) valore da inserire.
  Modifica l'oggetto grill, inserendo (value) nella casella grill [row , column]

Classe Player

Descrizione:

Classe che descrive le proprietà dei giocatori di Forza 4
Principi applicati: Class Cohesion, Separating Accessors and Mutators, Minimizing Side Effects, Names Consistency

Attributi:

- String **name**
  Nome del giocatore da registrare.

- int **number**
  numero intero, può valere 1 o 2. Assegnato al giocatore per rappresentare il suo colore.
  (usato un intero per semplicità nella logica, convertito in colore effettivo nella GUI)

Metodi:

- **Player** (String name, int n)
  Costruttore per la classe Player, semplicemente assegna nome e numero agli attributi.

- **getName** ()
  Il metodo restituisce il nome dell'oggetto player.

- **getNumber** ()
  Il metodo restituisce il numero dell'oggetto player.

Classe Person

Descrizione:

Classe che rappresenta una persona. La classe è in relazione di ereditarietà con Player
Principi applicati: Class Cohesion, Minimizing Side Effects, Names Consistency, Inheritance

Attributi:

- String **name**

Nome della persona

Metodi:

- **Person** (String n)
  Costruisce un oggetto Person dato un nome

- String **getName** ()
  Ritorna il nome dell' oggetto Person

Classe LoadGame

Descrizione:

Classe che si occupa di salvare e caricare una partita , specificando il nome in entrambi i casi
Principi applicati: Class Cohesion, Separating Accessors and Mutators, Names Consistency

Attributi:

- Grid **grid**
- String **pathFileName**

Metodi:

- **LoadGame** (Grid g)
  Costruisce una griglia grid partendo da una grid esistente.

- **LoadGame** (String pathFileName)
  Costruisce una griglia da editare per la partita da caricare

- **saveGame** (int turn,String player1, String player2, String gameName)
  Salva la partita in in un file. In particolare salva i valori della griglia, il turno e i giocatori della partita

- **resumeGame** ()
  Carica la partita da un file. Sostituisce i valori della griglia salvati nella nuova partita
- **turnPlayer** ()
  Sostituisce il turno salvato nella nuova partita

- **namesPlayers** ()
  Sostituisce i nomi dei giocatori salvati nella nuova partita

Classe GUI (main)

Descrizione:

Classe dell' Interfaccia grafica per l'utente (descrizione più dettagliata in sezioni successive). La classe agisce da main.

Principi applicati: Class Cohesion

Attributi:

V Attributi per Interfaccia Grafica V

JFrame **frame**
JTextField **g1FieId**
JTextField **g2FieId**
JInternalFrame **internalFrame**
JLabel l**blNewLabel_1**
JLabel **turni_label**
JLabel[][] **slots**
JButton[] **buttons**

V Attributi per Logica V

final int **xsize** = 6
final int **ysize** = 7
ArrayList<Player> **giocatori**
Griglia **grill**
Forza4 **game**
int **turno**

Metodi:

- **main** (String[] args)
  Il main. Lancia l'applicazione e crea una nuova finestra GUI

- **GUI**()
  IL costruttore, chiama initialize()

- **initialize**()
  Crea oggetti Griglia, ArrayList<Player> e Forza 4 delle rispettive classi, gestisce i turni e tutti i componenti del frame GUI

- **actionPerformed**(ActionEvent e) – Relativo a bottone "Conferma Nomi"
  Metodo chiamato al click sul relativo bottone: inserisce nell'oggetto ArrayList<Player> **giocatori** i nomi scelti e il colore

- **actionPerformed**(ActionEvent e) – Relativo a bottone "Resetta Partita"
  Metodo chiamato al click sul relativo bottone: Inizializza una nuova partita con una interfaccia "pulita" e nuovi oggetti.

- **getInternalFrameVisible**()
- **setInternalFrameVisible**(boolean visible)
  I metodi per la visibilità del frame interno

- **updateBoard**()

Metodo che sincronizza l'oggetto griglia (parte logica) con la rappresentazione grafica (JLabel[][] **slots).**
Per ogni valore 1 o 2 nelle caselle della matrice griglia, modifica il background del corrispettivo Jlabel nella matrice di Jlabel, con colore rosso se valore 1, altrimenti giallo.

- **resetGame** ()
  Metodo che resetta la partita corrente con una nuova, re-inizializza tutti le variabili della logica e la GUI.

Classe Connect4Tester (ex-main)

Descrizione:

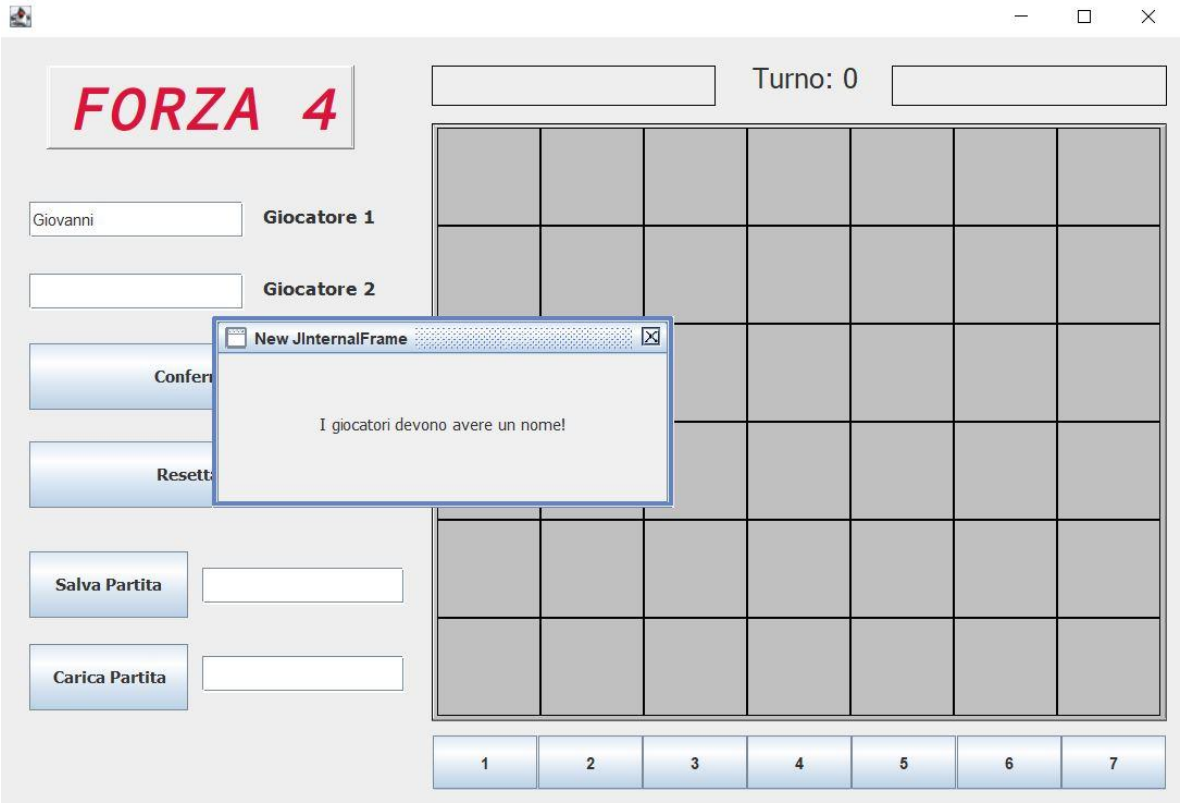
Classe utilizzata nelle fasi iniziali di progettazioni (precedente alla creazione della GUI), permette di testare il programma stampando l'output a video.
L'utilizzo di questa classe ci ha permesso di debuggare il codice

Attributi:

Scanner **in**= new Scanner(System.in)
ArrayList<Player> **players**
Grid **grid**
int **turn**

Descrizione delle funzionalità

In questa sezione descriviamo il flusso del nostro programma e come le funzionalità richieste sono state implementate, con utilizzo di screenshot del programma stesso (GUI).



Inizializzazione

Una volta avviato il programma (con GUI) partirà una fase di “preparazione” del gioco, in cui verrà richiesto ai giocatori di inserire i propri nomi (come da requisito).
Se questa condizione non è soddisfatta, ad esempio se uno dei due giocatori (o entrambi) ha come nome una stringa vuota, il programma non avanzerà e non si potrà avviare una regolare partita.

In questa fase non è possibile salvare una partita (dato che non c’è una partita in corso).

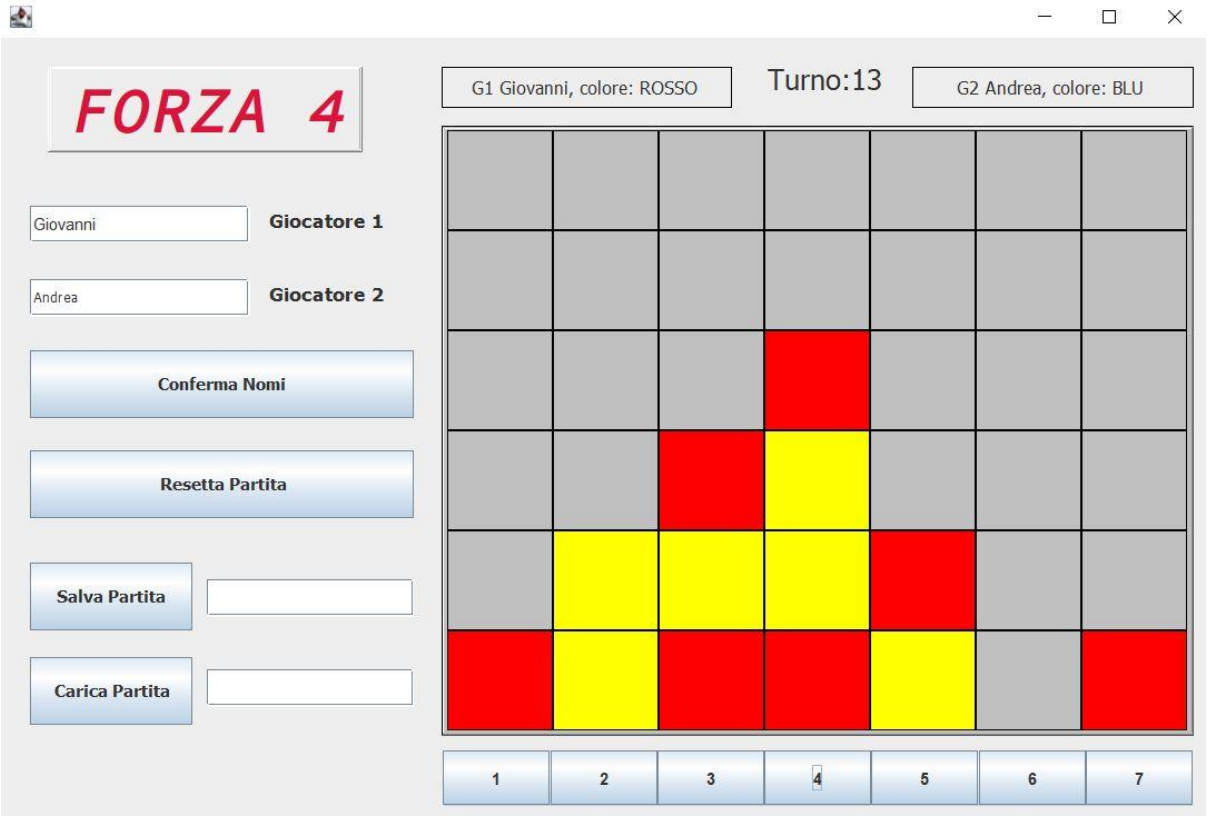
Inoltre in questa fase ed in quelle successive è possibile caricare una partita fornendo il path di una partita salvata in precedenza. Per “caricare una partita” si utilizza un file .txt da cui si prendono informazioni circa il nome dei giocatori, i colori assegnati ai giocatori, il turno e lo stato della griglia). Il file .txt in questione deve essere scritto in precedenza dalla funzione “salva partita”

Una volta inseriti i nomi correttamente e confermato (con l'apposito bottone nella GUI), il programma assegna ai Giocatori due colori diversi (Rosso e Giallo) che rappresentano il colore delle pedine che utilizzeranno successivamente.

Da adesso la partita è pronta per cominciare, ed è regolata da un contatore "Turno" per alternare i turni tra i 2 giocatori (come da requisito).

Da questa fase in poi è possibile salvare la partita corrente assegnandogli un nome arbitrario (il che salverà i dati della partita nel file .txt)

In geneale "salvare" o "continuare" una partita senza un nome del file / path non consentirà l'operazione.



Svolgimento partita

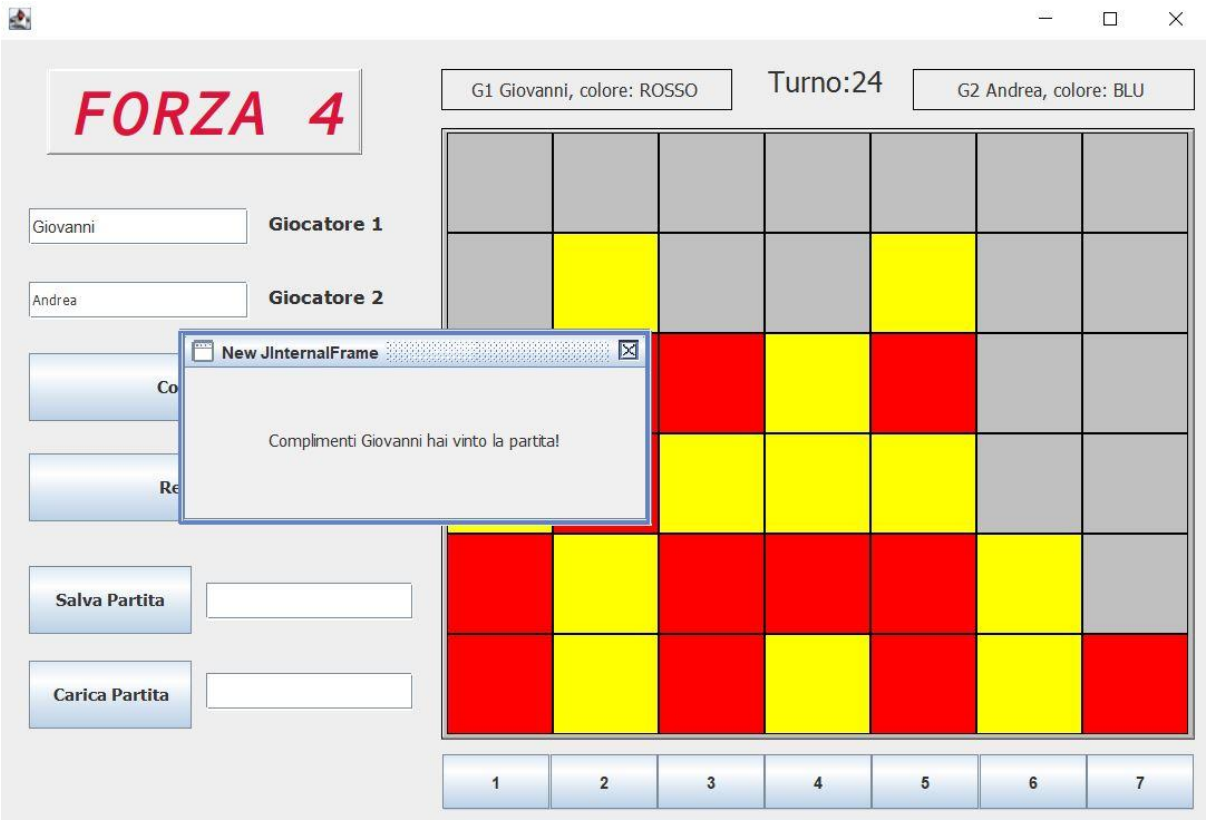
In questa fase i giocatori inseriscono a turno una pedina del colore assegnatogli in precedenza, selezionando una colonna. (nella GUI vengono utilizzati i bottoni col numero della rispettiva colonna per compiere questa azione).

Ad esempio se Giovanni nel turno 13 ha inserito una pedina rossa in una colonna, nel turno 14 toccherà ad Andrea che inserirà una pedina gialla in una colonna scelta.

All'inserimento di una pedina in una colonna (non piena), la pedina stessa si andrà a collocare nello spazio sopra allo stack di pedine che già si trovano in quella colonna, così da aumentare lo spazio occupato di quella colonna di 1, con il colore inserito.

Se una colonna C viene riempita in un turno X, nei turni successivi non sarà concessa l'azione di mettere una pedina nella colonna piena C.

I giocatori possono visualizzare lo stato del gioco su una matrice 6X7, che si aggiorna ogni volta che una pedina viene inserita.



Termine Partita

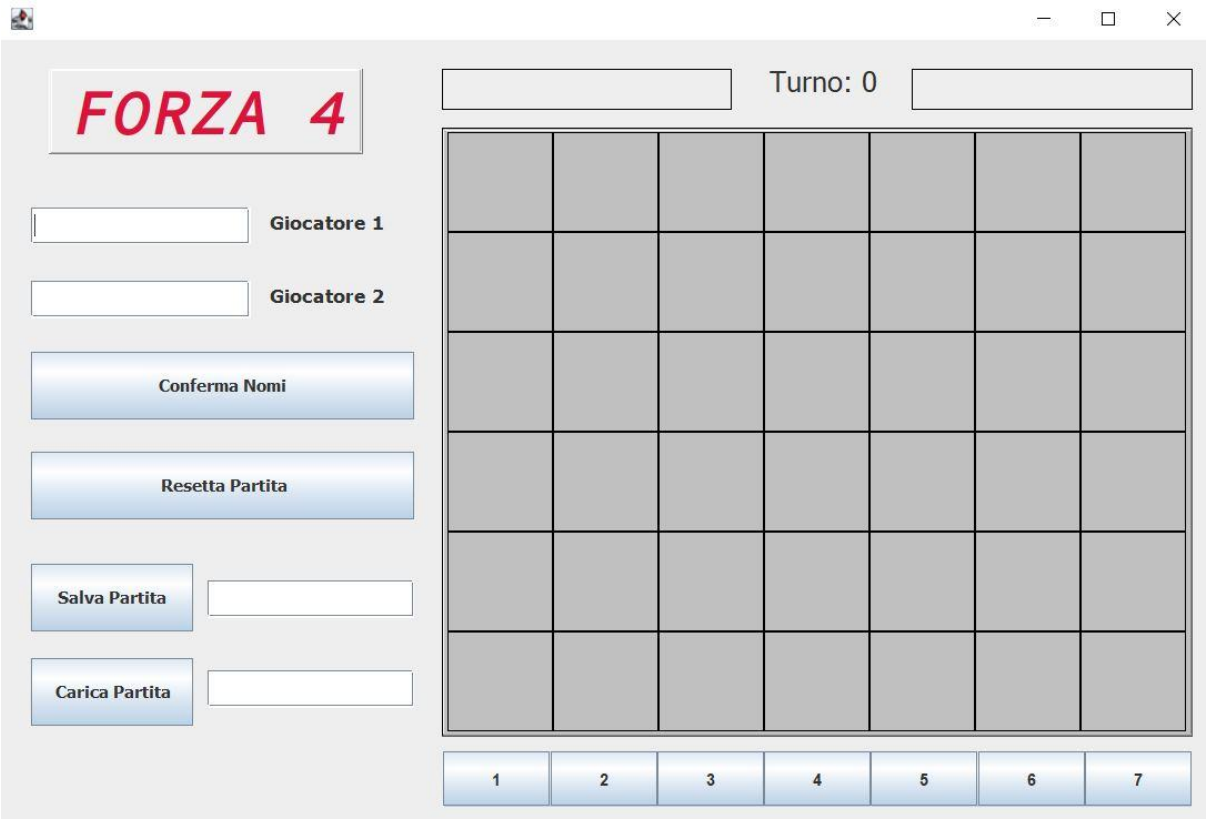
Lo scopo dei due giocatori è di allineare 4 pedine dello stesso colore per vincere.

Il programma rileva lo stato della griglia ad ogni inserimento per controllare che questa condizione sia verificata, ovvero se 4 pedine (Rosse o Gialle) sono allineate in orizzontale, verticale, diagonale e contro-diagonale.

Se si raggiunge quest' ultimo caso, Il programma segnala la vittoria del relativo giocatore (con opportuno textBox nella GUI). In caso contrario, ovvero se nessun giocatore riesce a vincere la partita, essa terminerà nell'ultimo turno, quando tutta la griglia è riempita e quindi non sono consentite più mosse. Questo caso viene segnalato come pareggio.

P.S. nella GUI ad ogni azione non concessa sarà visualizzato il textBox con la motivazione dell'errore causato.

Manuale della GUI



Descrizione e Utilità Interfaccia

Per questo progetto sono stati usati componenti della libreria javax.swing e java.awt.
All'avvio del programma con GUI, la finestra che si aprirà si può immaginare divisa concettualmente in parte destra e sinistra.

- Parte destra: essa è adibita alla partita vera e propria e contiene elementi collegati allo sviluppo della partita quando è in corso

    - Griglia (al centro): luogo dove le pedine dei giocatori vengono mostrate al momento dell'inserimento e durante il resto della parita. Realizzata con matrice di JLabel, modificando il colore di background quando viene inserita una nuova pedina

    - Bottoni (in basso): strumento per i giocatori per dare il comando di inserire le pedine nella griglia. Ci sono 7 JButton in totale, ognuno corrisponde alla colonna in cui il giocatore vuole inserire la pedina del suo colore (es. bottone "1" -> inserimento pedina in colonna 1)

- Turno (in alto): contatore del turno corrente della partita, ulteriore informazione per i giocatori sull'andamento della partita (es. partita pareggia al turno 42)

- Etichette (in alto): due JLabel segnano i nomi dei giocatori e i relativi colori, assegnati nella fase "pre-partita", utile ai giocatori per sapere che sono ingaggiati nella partita e il colore a loro assegnato

- Parte sinistra: essa è adibita per funzioni "meta-partita", ovvero operazioni che non contribuiscono direttamente all'avanzamento della partita

    - Titolo

    - Campi di testo (a sinistra): JTextField dove i giocatori devono inserire i propri nomi prima di iniziare la partita

    - JLabel "Giocatore 1" e "Giocatore 2" (a sinistra): indica ai giocatori di inserire i nomi nella casella corrispondente a sinistra

    - Bottone "Conferma Nomi" (al centro a sinistra): JButton usato per confermare i nomi dei giocatori una volta inseriti nei campi di testo. Rende visibile il contenuto in Etichette

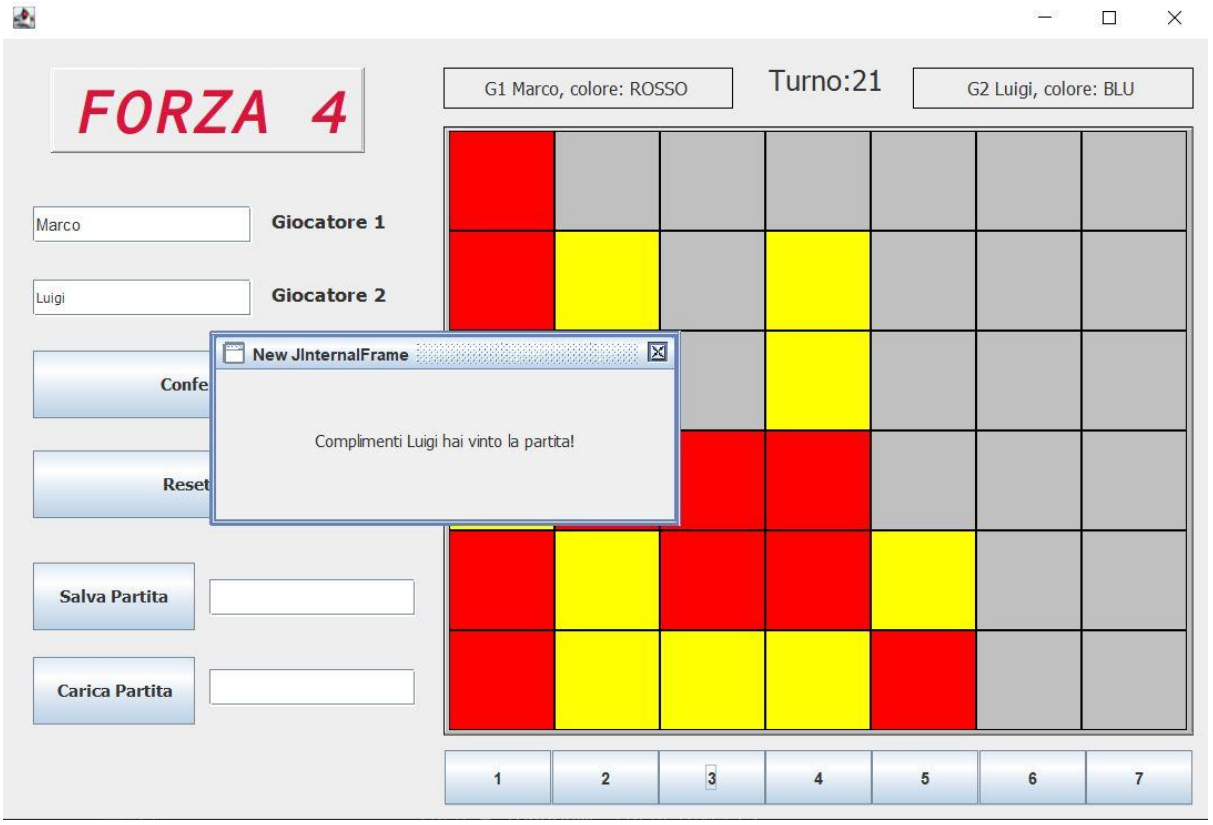
    - Bottone "Resetta Partita" (al centro): usato per "pulire" la GUI e inizializzare tutti le variabili della logica, così da poter iniziare una nuova partita da 0

    - Bottone "Salva Partita" (in basso a sinistra): usato per salvare la partita, specificando un nome nel textField alla sua destra

    - Bottone "Carica Partita" (in basso a sinistra): usato per caricare una partita, specificando un path nel textField alla sua destra

    - Campi di testo (a sinistra in basso): JTextField dove poter inserire i nomi o path del file per poi caricare o salvare una partita

- Finestra Extra: Un JInternalPanel a comparsa che contiene un Jlabel. Essa appare al momento di un errore o un'azione non permessa al giocatore, con il relativo testo di errore. Chiudibile.



Referenti di sviluppo

Insieme abbiamo discusso e ideato l'architettura principale del programma, qui sotto vengono elencati i principali ambiti di sviluppo di ogni membro
(Da notare che è presente lavoro cooperativo su alcuni di questi ambiti)

Dino Livio ha principalmente lavorato su:

- Connect4Tester
- GUI
- Relazione Progetto
- checkPosition ()

Daniele Federici ha principalmente lavorato su:

- Connect 4
- Grid
- Player / Person
- LoadGame

Il codice delle classi è stato opportunamente commentato in inglese, dal corrispettivo membro che ci ha lavorato.

File README

Come utilizzare il programma: compilare ed eseguire direttamente la classe GUI che funge da main del programma senza parametri aggiuntivi. Apparirà la schermata di gioco.

Come giocare a “Forza 4”:

1) Scrivere i nomi dei giocatori negli slot a sinistra e confermare con apposito bottone.
2) Inizia il gioco!, Giocatore 1 seleziona la colonna (con il bottone corrispondente in basso) in cui si desidera inserire la pedina.
3) Passare la mossa al Giocatore 2, che sceglierà una colonna e così via.
4) Il Giocatore 1 o 2 vince quando 4 pedine del suo colore sono adiacenti (in verticale, orizzontale e diagonale).