# Contents

# Relazione

Michele Leuti - 1872217

July 2022

## 1 Introduzione

Forza4 consiste è un gioco in cui vi sono un massimo di due giocatori. A turno il giocatore inserirà una pedina nella colonna scelta. L'obiettivo del gioco è quello di ottenere 4 pedine dello stesso colore adiacenti.

Il programma simula in tutto il gioco Forza4 a partire dalla creazione della tabella, creazione dei giocatori a cui è possibile assegnare un nome ed una pedina, logica e regole del gioco, vincita, pareggio, e gestione dei casi illegali.

Qui di seguito sono riportate dettagliatamente la descrizione delle classi, le funzionalità principali del programma e il manuale della GUI.

Il progetto è caricato anche sul mio profilo github:
**https://github.com/michele1777777/Forza4**

## 2 Diagramma UML

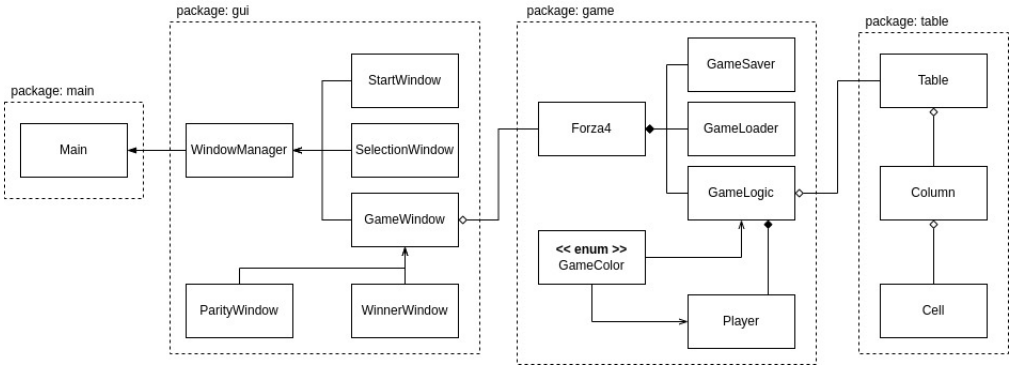



Figure 1: Diagramma UML

# 3 Descrizione delle classi

```
LEGENDA:
    o PACKAGE
        - CLASS
          ATTRIBUTES
            * METHODS
               ° PARAMETHERS
```

- main: il main package.
  - **Main**: La classe principale del programma tramite cui è possibile avviarlo.

    ATTRIBUTI: LA CLASSE NON HA ATTRIBUTI.
    COSTRUTTORI: LA CLASSE NON HA UN COSTRUTTORE.

    METODI:

    * *main*: il metodo main del programma Main, con la quale si avvia il programma.
  - **Forza4Test**: La classe test del programma.

    ATTRIBUTI: LA CLASSE NON HA ATTRIBUTI.
    COSTRUTTORI: LA CLASSE NON HA UN COSTRUTTORE.

    METODI:

    * *main*: il metodo main che effettua i test.

- table: il package che contiene le classi che compongono la table.
  - **Cell**: Rappresenta una cella della tabella.

    `ATTRIBUTI:`

    1. GameColor color: il colore della cella.

    `COSTRUTTORI:`

    * *Cell( )*: Costruttore della classe Cell che costruisce un oggetto cella vuota(senza colore).
    * *Cell(GameColor color)*: Costruttore della classe Cell che costruisce un oggetto cella piena(colorata).
      · color = il colore con cui colorare la cella.

    `METODI:`

    * *setColor(GameColor color)*: Imposta il colore della cella secondo il colore preso in input.
      · color = il colore con cui colorare la cella.
    * *getColor( )*: Restituisce il colore della cella.
  - **Column**: Rappresenta una colonna della tabella.

    `ATTRIBUTI:`

    1. Cell[ ] cells: le celle della colonna.
    2. int height: l'altezza della colonna.
    3. nextCell: l'indice della colonna che tiene traccia della prima cella disponibile per l'inserimento.

    `COSTRUTTORI:`

    * *Column(int height)*: Costruttore della classe Column.
      · height = l'altezza della colonna.

    `METODI:`

    * *getCell(int i)*: Restituisce la i-esima cella della colonna.
      · i = l'indice della cella.
    * *getCells( )*: Restituisce tutta la celle della colonna come array di celle.
    * *getHeight( )*: Restituisce l'altezza della colonna.
    * *getLastCellIndex( )*: Restituisce l'indice dell'ultima colonna in cui è stato effettuato un inserimento.
    * *insert(GameColor color)*: Effettua un inserimento nella prima posizione disponibile della colonna.
      · color = il colore con cui colorare la cella.

– **Table**: Rappresenta la tabella di gioco.

`ATTRIBUTI:`

1. COlumn[ ] columns = le colonne della tabella.
2. int height = l'altezza della tabella.
3. int lenght = la lunghezza della tabella.

`COSTRUTTORI:`

* *Table(int height, int lenght)*:
  Costruttore della classe Table.
  - height = l'altezza della tabella.
  - lenght = la lunghezza della colonna.

`METODI:`

* *getCell(int row, int column)*: Restituisce la sella della riga row e della colonna column.
  - row = l'indice della riga della cella.
  - column = l'indice della colonna della cella.
* *getColumn(int column)*: Restituisce la colonna di indice column specificato.
  - column = l'indice della colonna.
* *getColumns( )*: Restituisce tutte le colonne della tabella come un array di colonne.
* *getHeight( )*: Restituisce l'altezza della tabella.
* *getLenght( )*: Restituisce la lunghezza della tabella.
* *getDiagonal(int i)*: Restituisce la i-esima diagonale della tabella.
  - i = l'indice della diagonale.
* *getAntiDiagonal(int i)*: Restituisce la i-esima antidiagonale della tabella.
  - i = l'indice della anti-diagonale.

- game: il package che contiene tutte le classi che riguardano il gioco.
  - **Player**: La classe che rappresenta il giocatore.

    `ATTRIBUTI:`

    1. GameColor color: il colore del giocatore.
    2. String name: il nome del giocatore.

    `COSTRUTTORI:`

    * *Player(String name, GameColor color)*: Costruttore della classe Player.
      · name = il nome del giocatore.
      · color = il colore assegnato al giocatore.

    `METODI:`

    * *getColor( )*: Restituisce il colore del giocatore.
    * *getName( )*: Restituisce il nome del giocatore.
  - **GameColor (enum)**: Ogni colore è mappato grazie all'enum GameColor ad un colore della classe Color e ad un carattere che lo rappresenta.

    `ATTRIBUTI:`

    1. RED = il colore rosso in rgb.
    2. GREEN = il colore verde in rgb.
    3. BLUE = il colore blu in rgb.
    4. YELLOW = il colore giallo in rgb.
    5. WHITE = il colore bianco in rgb.
    6. BLACK = il colore nero in rgb.
    7. EMPTY = un valore nullo.
    8. Color color = un oggetto della classe Color.

    `COSTRUTTORI:`

    * *GameColor(int r, int g, int b)*: Costruttore dell'enum GameColor.
      · r = il valore rosso in rgb del colore.
      · g = il valore verde in rgb del colore.
      · b = il valore blu in rgb del colore.

    `METODI:`

    * *get( )*: Restituisce l'oggetto Color del GameColor corrispondente.
    * *match(String color)*: Restituisce il colore GameColor corrispondente alla stringa passato in input.
      · color = la stringa che contiene informazioni sul colore.

– **GameLogic**: La classe che gestisce tutte le regole del gioco.

`ATTRIBUTI:`

1. Table table = la tabella di gioco.
2. Player[ ] players = Un array di player che contiene il giocatore 1 (in posizione 0) e il giocatore 2(in posizione 1).
3. int currentPlayer = indice dell'array players che tiene conto del giocatore corrente.
4. int[ ] moves = un array di interi in cui ad ogni elemento corrisponde l'indice della colonna in cui è stata fatta la mossa.
5. int movesCounter = un indice che scorre l'array moves.

`COSTRUTTORI:`

* *GameLogic(Table table, Player p1, Player p2)*:
  Costruttore della classe GameLogic che si utilizza per una nuova partita, si occupa di scegliere randomicamente il player iniziale, e di inizializzare tutti i parametri che si utilizzano durante lo svoglimento del gioco.
  - table = la tabella di gioco.
  - p1 = il giocatore 1.
  - p2 = il giocatore 2.
* *GameLogic(Table table, int initialPlayer, Player p1, Player p2)*:
  Costruttore della classe GameLogic che si utilizza per ricreare la partita precedentemente salvata, a differenza del primo costruttore infatti si prende in input anche il player che ha iniziato la partita.
  - table = la tabella di gioco.
  - initialPlayer = un indice che tiene conto chi ha iniziato la partita.
  - p1 = il giocatore 1.
  - p2 = il giocatore 2.

`METODI:`

* *chooseInitialPlayer( )*: Decide randomicamente il giocatore che deve iniziare la partita.
* *getInitialPlayer( )*: Restituisce il giocatore che ha iniziato la partita.
* *getCurrentPlayer( )*: Restituisce il giocatore che deve compiere la mossa di gioco corrente.
* *getCellColor(int row, int column)*: Restituisce il colore della cella della riga row in colonna column.
  - row = l'indice di riga della cella.
  - column = l'indice di colonna della cella.

* *getMoves( )*: Restituisce le mosse compiute di gioco come un array di interi, in cui ogni elemento rappresenta l'indice della colonna in cui è stata compiuta la mossa.
* *checkCellsLinearly(Cell[ ] cells)*: Restituisce true se nell'array di celle preso in input vi sono 4 celle adiacenti dello stesso colore.
  - cells = l'array di celle da analizzare.
* *checkWin(int column)*: Controlla se la mossa compiuta è una mossa vincente, ovvero controlla chiamando ogni volta checKCellsLinearly la riga della cella, la colonna della cella, la sua diagonale e la sua antidiagonale.
  Torna true se il checkCellLinearly su almeno una di esse ha restituito true, altrimenti restituisce false.
  - column = l'indice della colonna in cui è stata compiuta la mossa.
* *checkParity( )*: Controlla se tutte le colonne della tabella sono piene, in tal caso restituisce true, altrimenti false.
* *makemove(int column)*: Si occupa di compiere la mossa di gioco facendo un inserimento nella colonna presa in input. Richiama il checkWin e il checkParity, per controllare se la mossa è una mossa vincente o la partita è finita in pareggio. Restituisce 1, se la partita è vinta, 2 se la partita è terminata in pareggio, altrimenti 0 se la partita non è finita.
  - column: la colonna in cui deve effettuare un inserimento.

– **GameSaver**: La classe che si occupa di salvare la partita corrente.

`ATTRIBUTI: LA CLASSE NON HA ATTRIBUTI.`
`COSTRUTTORI: LA CLASSE NON HA UN COSTRUTTORE.`

`METODI:`

* *saveToFile(String path)*: Si occupa di salvare le informazioni di gioco in un file.
  - path: il nome del percorso del file di salvataggio.

– **GameLoader**: La classe che si occupa di caricare la partita da un file.

`ATTRIBUTI: LA CLASSE NON HA ATTRIBUTI.`
`COSTRUTTORI: LA CLASSE NON HA UN COSTRUTTORE.`

`METODI:`

* *LoadFromFile(String path)*: Si occupa di caricare le informazioni di gioco da un file, e impostare la partita secondo quest'ultime.
  - path = il nome del percorso del file di caricamento.

– **Forza4**: La classe che gestisce il gioco Forza4.

`ATTRIBUTI:`

1. GameLogic gameLogic = Un oggetto GameLogic.
2. Table table = La tabella di gioco.
3. Player p1 = il giocatore 1.
4. Player p2 = il giocatore 2.

`COSTRUTTORI:`

* *Forza4(Player p1, Player p2)*: Il primo costruttore della classe Forza 4 che prende in input i due giocatori e costruisce il gioco con una tabella predefinita (7x6).
  · p1 = il giocatore 1.
  · p2 = il giocatore 2.
* *Forza4(int lenght, int height, Player p1, Player p2)*: Il secondo costruttore della classe Forza4 che prende in input oltre ai due giocatori anche la lunghezza e l'altezza della tabella di gioco.
  · lenght = la lunghezza della tabella.
  · height = l'altezza della tabella.
  · p1 = il giocatore 1.
  · p2 = il giocatore 2.
* *Forza4(int lenght, int height, int initialPlayer, Player p1, Player p2)*: Il terzo costruttore della classe forza 4 che permette di impostare anche il giocatore che ha iniziato la partita.
  · lenght = la lunghezza della tabella.
  · height = l'altezza della tabella.
  · initialPlayer = il giocatore che ha iniziato la partita.
  · p1 = il giocatore 1.
  · p2 = il giocatore 2.

`METODI:`

* *getTableLenght( )*: Restituisce la lunghezza della tabella.
* *getTableHeight( )*: Restituisce l'altezza della tabella.
* *getPlayer1( )*: Restituisce il giocatore uno.
* *getPlayer2( )*: Restituisce il giocatore due.
* *getCurrentPlayer( )*: Restituisce il giocatore corrente.
* *getCurrentPlayerColor( )*: Restituisce il colore del giocatore corrente.
* *getInitialPlayer( )*: Restituisce il giocatore che ha iniziato la partita.
* *getCellColor(int row, int column)*: Restituisce il colore della cella della riga row e della colonna column.

* *getColumnLastIndex(int column)*: Restituisce l'indice dell'ultima colonna in cui è stato effettuato un inserimento.
* *getMoves( )*: Richiama il getMoves della classe GameLogic(Vedi il metodo getMoves( ) della classe GameLogic).
* *makeMove( )*: Richiama il makeMove della GameLogic(Vedi il metodo makeMove( ) della classe GameLogic).
* *runGame( )*: Consente di eseguire il programma da terminale.

- gui: Il package dell'interfaccia grafica.
  - **StartWindow**: La classe che gestisce la finestra di avvio del gioco. Estende la classe JFrame e implemente l'interfaccia ActionListener.

    `ATTIBUTI:`

    1. Dimension WINDOW_DIM = le dimensioni del frame.
    2. JPanel background = il background del frame.
    3. JPanel titlePanel = il JPanel che contiene la JLabel del titolo.
    4. JLabel titleLabel = la JLabel che contiene l'immagine del titolo.
    5. JPanel imagePanel = il JLabel che contiene la JLabel dell'immagine della finestra di avvio.
    6. JLabel imageLabel = la JLabel che contiene l'immagine della finestra di avvio.
    7. JPanel buttonPanel = il JPanel che contiene i bottoni start, load game e quit.
    8. JButton newGameButton = il bottone che permette di iniziare una nuova partita.
    9. JButton loadGameButton = il bottone che permette di caricare una partita precedente
    10. JButton quitButton = il bottone che permette di uscire dal programma.

    `COSTRUTTORI:`

    * *StartWindow( )*: Il costruttore della classe StartWindow

    `METODI:`

    * *createBackground( )*: Crea il background e lo aggiunge al frame.
    * *createTitlePanel( )*: Crea il JPanel che contiene la JLabel del titolo e lo aggiunge al background.
    * *createImagePanel( )*: Crea il JPanel che contiene la JLabel dell'immagine e lo aggiunge al background.
    * *createButtonPanel( )*: Crea il JPanel che contiene i bottoni start, load game e quit e lo aggiunge al background.
    * *createNewGameButton( )*: Crea il bottone che permette di iniziare una nuova partita.

* *createLoadGameButton( )*: Crea il bottone che permette di caricare una partita precedente.
* *createQuitButton( )*: Crea il bottone che permette di uscire dal programma.
* *ActionPerformed(ActionEvent e)*: Il metodo sovrascritto dell'interfaccia actionListener che permette di far compiere le azioni ai bottoni.
  - e = L'oggetto ActionEvent.
* *createPanel(Border border, Color color, int x, int y, int width, int height)*: Crea il pannello secondo le informazioni fornite in input.
  - border = il bordo del pannello.
  - color = il colore di sfondo del pannello.
  - x = la coordinata x del pannello.
  - y = la coordinata y del pannello.
  - width = la larghezza del pannello.
  - height = l'altezza del pannello.
* *createImage(String path)*: Crea un oggetto image con la path presa in input e lo restituisce.
  - path = il percorso dell'immagine.
* *createImageIcon(String path)*: Restituisce un oggetto ImageIcon.
  - path = il percorso dell'immagine.
* *setup( )*: Crea il frame.

– **SelectionWindow**: La classe che gestisce la finestra di selezione dei nomi dei giocatori, dei colori e delle dimensioni della tabella.

`ATTRIBUTI:`

1. Dimension WINDOW_DIM = le dimensioni del frame.
2. JPanel background = il background del frame.
3. JPanel titlePanel = il JPanel che contiene la JLabel del titolo.
4. JLabel titleLabel = la JLabel che contiene l'immagine del titolo.
5. JPanel p1InfoPanel = il JPanel che contiene le informazioni del giocatore 1.
6. JTextField p1NameField = il campo di testo per il nome del giocatore 1.
7. Colorgroup p1ColorGroup = L'oggetto che raggruppa i bottoni a scelta singola dei colori del giocatore 1(JRadioButton).
8. JRadioButton p1RedColor = bottone del colore rosso.
9. JRadioButton p1YellowColor = bottone del colore giallo.
10. JRadioButton p1GreenColor = bottone del colore verde.
11. JRadioButton p1BlueColor = bottone del colore blu.
12. JPanel p2InfoPanel = il JPanel che contiene le informazioni del giocatore 2.
13. JTextField p2NameField = il campo di testo per il nome del giocatore 2.
14. Buttongroup p2ColorGroup = L'oggetto che raggruppa i bottoni a scelta singola dei colori del giocatore 2(JRadioButton).
15. JRadioButton p2RedColor = bottone del colore rosso.
16. JRadioButton p2YellowColor = bottone del colore giallo.
17. JRadioButton p2GreenColor = bottone del colore verde.
18. JRadioButton p2BlueColor = bottone del colore blu.
19. String Color1Selected = Una stringa che salva le informazioni del colore scelto dal giocatore 1.
20. String Color2Selected = Una stringa che salva le informazioni del colore scelto dal giocatore 2.
21. JPanel tableDimensionPanel = Il pannello in cui vi sono i bottoni per la scelta delle dimensioni della tabella.
22. ButtonGroup tableDimension = L'oggetto che raggruppa i bottoni a scelta singola delle dimensioni della tabella.
23. JRadioButton tableDimension0 = il primo bottone a scelta singola delle dimensioni della tabella(5x4).
24. JRadioButton tableDimension1 = il secondo bottone a scelta singola delle dimensioni della tabella(6x5).
25. JRadioButton tableDimension2 = il terzo bottone a scelta singola delle dimensioni della tabella(7x6).

26. JRadioButton tableDimension3 = il primo bottone a scelta singola delle dimensioni della tabella(8x7).
27. String DimensionSelected = Una stringa che tiene conto della dimensione selezionata.
28. JPanel buttonPanel = il JPanel che contiene i bottoni start game e main menu.
29. JButton mainMebuButton = il bottone che permette di tornare alla finesta principale.
30. JButton startButton = il bottone che permette di avviare la partita con le informazioni fornite.

`COSTRUTTORI:`

* *SelectionWindow( )*: Il costruttore della classe SelectionWindow.

`METODI:`

* *addComponent( )*: Aggiunge i componenti al frame.
* *createBackground( )*: Crea il background e lo aggiunge al frame.
* *createTitlePanel( )*: Crea il JPanel che contiene la JLabel del titolo e lo aggiunge al background.
* *createP1infoPanel( )*: Crea il JPanel dove si possono inserire le informazioni del giocatore 1 e le aggiunge al background.
* *createP2infoPanel( )*: Crea il JPanel dove si possono inserire le informazioni del giocatore 2 e le aggiunge al background.
* *createP1ColorGroup( )*: Crea tutti i JRadioButton della scelta dei colori del giocatore 1 e li aggiunge al ButtonGroup.
* *createP2infoPanel( )*: Crea il JPanel dove si possono inserire le informazioni del giocatore 2 e le aggiunge al background.
* *createP2ColorGroup( )*: Crea tutti i JRadioButton della scelta dei colori del giocatore 2 e li aggiunge al ButtonGroup.
* *createLabel(String label)*: Crea la JLabel con il titolo preso in input.
  - label = il titolo della JLabel.
* *createNameField( )*: Crea un campo di testo.
* *createDimensionPanel( )*: Crea il JPanel che contiene i bottoni a scelta singola per le dimensioni della tabella.
* *createTableDimensionGroup( )*: Crea i JRadioButton delle dimensioni della tabella e il aggiunge al ButtonGroup.
* *createRadioButton(String buttonName)*: Restituisce un JRadioButton il cui nome è dato in input.
  - buttonName = il titolo del bottone.
* *createButtonsPanel( )*: Crea il JPanel che contiene i bottoni start e main menu.

* *createMainMenuButton( )*: Crea il bottone main menu che permette di tornare alla finestra principale.
* *createStartButton( )*: Crea il bottone start che permette di iniziare la partita.
* *setGame( )*: Permette di passare tutte le informazioni acquisite per creare un oggetto Forza4 che verrà passato al frame di gioco.
* *getChoosenPlayerColor(String color)*: Restituisce il colore scelto dal giocatore.
  - color: la stringa che contiene le informazioni sul colore scelto.
* *ActionPerformed(ActionEvent e)*: Il metodo sovrascritto dell'interfaccia actionListener che permette di far compiere le azioni ai bottoni.
  - e = L'oggetto ActionEvent.
* *createPanel(Border border, Color color, int x, int y, int width, int height)*: Crea il pannello secondo le informazioni fornite in input.
  - border = il bordo del pannello.
  - color = il colore di sfondo del pannello.
  - x = la coordinata x del pannello.
  - y = la coordinata y del pannello.
  - width = la larghezza del pannello.
  - height = l'altezza del pannello.
* *createImage(String path)*: Crea un oggetto image con la path presa in input e lo restituisce.
  - path = il percorso dell'immagine.
* *createImageIcon(String path)*: Restituisce un oggetto ImageIcon.
  - path = il percorso dell'immagine.
* *setup( )*: Crea il frame.

– **gameWindow**: La classe che gestisce la finestra di gioco.

`ATTRIBUTI:`

1. static Forza4 game = L'oggetto Forza 4.
2. Dimension WINDOW_DIM = le dimensioni del frame.
3. JPanel background = il background del frame.
4. int tableX = la coordinata x iniziale del JPanel della tabella.
5. int tableY = la coordinata y iniziale del JPanel della tabella.
6. JPanel versusPanel = il JPanel che contiene i nomi dei due sfidanti.
7. JPanel tablePanel = ilJPanel che contiene la tabella.
8. JLabel[ ][ ] table = la tabella di gioco come matrice di JLabel.
9. JPanel insertPanel = il JPanel che contiene i bottoni di inserimento.
10. JButton[ ] insertButtons = Un array di JButton ogniuno dei quali effettua un inserimento.
11. JPanel buttonsPanel = il JPanel che contiene i bottoni save game e main menu.
12. JButton saveButton = il bottone di salvataggio.
13. JButton mainMenuButton = il bottone che permette di tornare al menu iniziale.
14. JPanel gameTurnsPanel = il JPanel che contiene la JLabel delle informazioni sul turno corrente.

`COSTRUTTORI:`

* *GameWindow( )*: Il costuttore della classe game window.

`METODI:`

* *addComponent( )*: Aggiunge i componenti al frame.
* *createBackground( )*: Crea il JPanel del background e lo aggiunge al frame.
* *createVersusPanel( )*: Crea il JPanel che contiene i nomi dei due sfidanti.
* *createTablePanel( )*: Crea il JPanel che contiene la tabella di gioco.
* *createTable(int L, int H)*: Crea la grafica della tabella di gioco secondo le dimensioni prese in input.
  - L = la lunghezza della tabella.
  - H = l'altezza della tabella.
* *createInsertButtonsPanel( )*: Crea il JPanel che contiene i bottoni di inserimento.
* *createButtons( )*: Crea i bottoni di inserimento.

* *createButtonsPanel( )*: Crea il JPanel che contiene i bottoni save game e main menu.
* *createSaveButton( )*: Crea il bottone save button.
* *createMainMenuButton( )*: Crea il bottone che permette di tornare al menu iniziale.
* *createGameTurnsInfoPanel( )*: Crea il JPanel che contiene la JLabel delle informazioni del turno corrente.
* *changeCellColor(JLabel cell, GameColor color)*: permette di cambiare l'immagine della JLabel secondo il colore preso in input.
  - cell = la JLabel che rappresenta la cella.
  - color = il colore della cella.
* *ActionPerformed(ActionEvent e)*: Il metodo sovrascritto dell'interfaccia actionListener che permette di far compiere le azioni ai bottoni.
  - e = L'oggetto ActionEvent.
* *createPanel(Border border, Color color, int x, int y, int width, int height)*: Crea il pannello secondo le informazioni fornite in input.
  - border = il bordo del pannello.
  - color = il colore di sfondo del pannello.
  - x = la coordinata x del pannello.
  - y = la coordinata y del pannello.
  - width = la larghezza del pannello.
  - height = l'altezza del pannello.
* *createImage(String path)*: Crea un oggetto image con la path presa in input e lo restituisce.
  - path = il percorso dell'immagine.
* *createImageIcon(String path)*: Restituisce un oggetto ImageIcon.
  - path = il percorso dell'immagine.
* *setup( )*: Crea il frame.

– **WinnerWindow**: La classe che gestisce la finestra della vittoria.

`ATTRIBUTI:`

1. static Forza4 game = L'oggetto Forza 4.
2. Dimension WINDOW_DIM = le dimensioni del frame.
3. JPanel background = il background del frame.
4. JPanel winnerInfoPanel = il JPanel che contiene le informazioni sul vincitore.
5. JPanel buttonsPanel = il JPanel che contiene i pulsanti new game, main menu e quit.
6. JButton newGameButton = il bottone che permette di iniziare una nuova partita.
7. JButton mainMenuButton = il bottone che permette di tornare al menu principale.
8. JButton quitButton = il bottone per terminare il programma.

`COSTRUTTORI:`

* *WinnerWIndow( )*: Il costruttore della classe WinnerWindow.

`METODI:`

* *addComponent( )*: Aggiunge i componenti al frame.
* *createBackground( )*: Crea il JPanel del background e lo aggiunge al frame.
* *createWinnerInfoPanel( )*: Crea il JPanel che contiene le informazioni sul vincitore.
* *createButtonsPanel( )*: Crea il JPanel che contiene i bottoni new game, main menu e quit.
* *createNewGameButton( )*: Crea il bottone new game.
* *createMainMenuButton( )*: Crea il bottone che permette di tornare al menu iniziale.
* *createQuitButton( )*: Crea il bottone che permette di terminare il programma.
* *ActionPerformed(ActionEvent e)*: Il metodo sovrascritto dell'interfaccia actionListener che permette di far compiere le azioni ai bottoni.
  - e = L'oggetto ActionEvent.
* *createPanel(Border border, Color color, int x, int y, int width, int height)*: Crea il pannello secondo le informazioni fornite in input.
  - border = il bordo del pannello.
  - color = il colore di sfondo del pannello.
  - x = la coordinata x del pannello.
  - y = la coordinata y del pannello.

    - width = la larghezza del pannello.
    - height = l'altezza del pannello.
- *createImage(String path)*: Crea un oggetto image con la path presa in input e lo restituisce.
    - path = il percorso dell'immagine.
- *createImageIcon(String path)*: Restituisce un oggetto ImageIcon.
    - path = il percorso dell'immagine.
- *setup( )*: Crea il frame.

– **ParityWindow**: La classe che gestisce la finestra del pareggio.

`ATTRIBUTI:`

1. Dimension WINDOW_DIM = le dimensioni del frame.
2. JPanel background = il background del frame.
3. JPanel buttonsPanel = il JPanel che contiene i pulsanti new game, main menu e quit.
4. JButton newGameButton = il bottone che permette di iniziare una nuova partita.
5. JButton mainMenuButton = il bottone che permette di tornare al menu principale.
6. JButton quitButton = il bottone per terminare il programma.

`COSTRUTTORI:`

* *ParotyWIndow( )*: Il costruttore della classe ParityWindow.

`METODI:`

* *addComponent( )*: Aggiunge i componenti al frame.
* *createBackground( )*: Crea il JPanel del background in cui viene inserita una scritta che informa che la partita è terminata in pareggio.
* *createButtonsPanel( )*: Crea il JPanel che contiene i bottoni new game, main menu e quit.
* *createNewGameButton( )*: Crea il bottone new game.
* *createMainMenuButton( )*: Crea il bottone che permette di tornare al menu iniziale.
* *createQuitButton( )*: Crea il bottone che permette di terminare il programma.
* *ActionPerformed(ActionEvent e)*: Il metodo sovrascritto dell'interfaccia actionListener che permette di far compiere le azioni ai bottoni.
  - e = L'oggetto ActionEvent.
* *createPanel(Border border, Color color, int x, int y, int width, int height)*: Crea il pannello secondo le informazioni fornite in input.
  - border = il bordo del pannello.
  - color = il colore di sfondo del pannello.
  - x = la coordinata x del pannello.
  - y = la coordinata y del pannello.
  - width = la larghezza del pannello.
  - height = l'altezza del pannello.
* *createImage(String path)*: Crea un oggetto image con la path presa in input e lo restituisce.

    - path = il percorso dell'immagine.
* *createImageIcon(String path)*: Restituisce un oggetto ImageIcon.
    - path = il percorso dell'immagine.
* *setup( )*: Crea il frame.

– **WindowManager**: La classe che gestisce tutte le finestre dell'interfaccia grafica.

`ATTRIBUTI:`

1. static StartWindow startMenu = L'oggetto StartWindow.
2. static SelectionWindow selectionMenu = L'oggetto SelectionWindow.
3. static GameWindow gameWindow = L'oggetto GameWindow.
4. static WinnerWindow winnerWindow = L'oggetto winnernWindow.
5. static ParityWindow parityWindow = L'oggetto ParityWindow.

`COSTRUTTORI: LA CLASSE NON HA UN COSTRUTTORE.`

`METODI:`

* *switchToStartWindow( )*: Apre la finestra del menu principale e chiude tutte le altre aperte.
* *switchToSelectionWindow( )*: Apre la finestra del menu di selezione e chiude tutte le altre aperte.
* *switchToGameWindow( )*: Apre la finestra di gioco e chiude tutte le altre aperte.
* *switchToWinnerWindow( )*: Apre la finestra della vittoria.
* *switchToGameWindow( )*: Apre la finestra del pareggio.
* *closeWindows( )*: Chiude tutte le finestre aperte.
* *quit*: Chiude il programma.

# 4 Descrizione delle funzionalità

## 4.1 creazione nuova partita

Tramite la schermata principale è possibile creare una nuova partita. Il programma aprirà una schermata nella quale è possibile creare i due giocatori:

- scelta del nome del giocatore;
- scelta del colore del giocatore.

Di default al giocatore 1 viene assegnato il rosso, al giocatore 2 il giallo. Il programma gestisce tutti i casi illegali che possono capitare:

- Ad uno o a entrambi i giocatori non viene assegnato un nome.
  In tal caso comparirà un pop-up di avviso che avviserà l'utente di assegnare un nome al giocatore a cui non lo è stato assegnato o, anche, di assegnarli ad entrambi.
- I giocatori hanno lo stesso nome.
  In tal caso comparirà un pop-up di avviso che avviserà l'utente che i giocatori devono avere un nome diverso.

Il programma inoltre non permette all'utente di selezionare lo stesso colore per entrambi i giocatori. Cambierà automaticamente dell'altro giocatore al colore successivo.

È possibile inoltre selezionare le dimensioni della tabella secondo un set predefinito:

```
5x4
6x5
7x6 (DEFAULT)
8x7
```

## 4.2 salvataggio di una partita

Tramite un JFileChooser viene scelto il path e il file name con cui salvare il file. Il programma scrive in un file le informazioni relative alla partita corrente, secondo il seguente formato:

```
ALTEZZA, LUNGHEZZA
NOME DEL GIOCATORE 1, COLORE DEL GIOCATORE 1
NOME DEL GIOCATORE 2, COLORE DEL GIOCATORE 2
GIOCATORE CHE HA INIZIATO LA PARTITA
LISTA MOSSE DI GIOCO
```

### 4.2.1 Colore del giocatore

Ogni colore è mappato grazie all'enum GameColor ad un colore della classe Color e ad un carattere che lo rappresenta. Esempio:

- Red - "R" - (255,0,0)
- Green - "G" - (20,148,68)

Per cui se il giocatore 1 ha selezionato il colore rosso, il programma scriverà nel file "R".

### 4.2.2 Player iniziale

Nel caso in cui il giocatore che ha iniziato la partita è il giocatore 1 verrà scritto "0" all'interno del file, altrimenti se ha iniziato il giocatore 2, verrà scritto "1".

### 4.2.3 Mosse di gioco

La lista delle mosse di gioco può contenere al massimo H x L mosse corrispondenti al numero totale di celle. Ogni indice della lista corrisponde ad un turno e di conseguenza potrà contenere uno dei seguenti valori:

- Nel caso in cui il turno sia stato svolto allora conterrà un numero, corrispondente all'indice della colonna in cui è stato effettuato un inserimento.
- Nel caso in cui il turno non sia stato svolto, conterrà il carattere "-".

Esempio file di salvataggio:

```
5,4
Michele,R
Gabriele,Y
0
0, 1, 2, 2, 1, 4, 3, 3, 2, -, -, -, -, -, -, -, -, -, -, -
```

Il programma utilizza una propria estensione dei file di salvataggio ".f4g".

### 4.3 caricamento partita esistente

Tramite un JFileChooser che filtra i file in base all'estensione ".f4g" (estensione dei file di gioco), possiamo navigare tra le cartelle del computer per cercare la partita da caricare.

Una volta letto il file, il cui formato è descritto nel paragrafo precedente, vengono estrapolate le informazioni riguardo i giocatori, le dimensioni della tabella e le mosse di gioco.

Vengono quindi creati i due oggetti player, la tabella e l'oggetto forza4, dopodiché il programma leggerà le mosse della partita salvata precedentemente e le effettuerà nuovamente sul nuovo oggetto Forza4 in modo da ricreare la partita.

## 4.4 svolgimento di una partita

### 4.4.1 descrizione tabella

Una tabella di lunghezza L e di altezza H possiede la seguente struttura:

- L colonne indicizzate da 0 a L - 1;
- H righe indicizzate da 0 a H - 1;
- L + H - 1 diagonali indicizzate da 0 a L + H - 2.
- L + H - 1 antidiagonali indicizzate da 0 a L + H - 2.

### 4.4.2 Inizio del gioco

il programma deciderà randomicamente chi dei due player inizia la partita.

### 4.4.3 Inserimento

Per l'inserimento si utilizza l'indice delle colonne ognuna delle quali presenta il parametro **nextCell**, che tiene traccia della prossima cella disponibile per l'inserimento successivo nella stessa.

Nella componente grafica, al di sopra di ogni colonna vi è un bottone, il quale una volta cliccato, effettuerà un inserimento nella colonna corrispettiva(Esempio: il 0-esimo bottone genererà un inserimento nella 0-esima colonna).

### 4.4.4 gestione dei casi speciali

Poiché la tabella possiede degli indice ben definiti, il programma si occuperà di gestire eventuali inserimenti illegali:

- Inserimento su una colonna non esistente(con indice inferiore a 0 o superiore a L - 1).
- inserimento su una colonna piena (il cui parametro nextCell corrisponde ad H).

### 4.4.5 controllo vincita e pareggio

Ogni volta viene effettuato un inserimento il programma controlla se la presenza di una mossa vincente, ossia la presenza di quattro celle adiacenti dello stesso colore allineate orizzontalmente, verticalmente o diagonalmente.

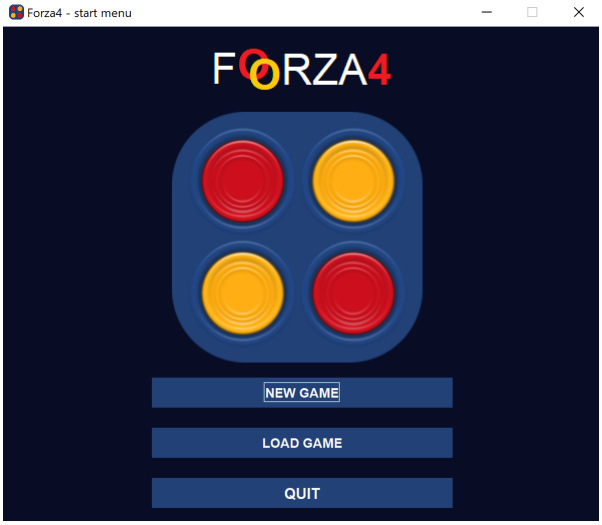
In particolare ogni volta viene creato un array di celle rappresentante la riga, la colonna, la diagonale o antidiagonale che si deve analizzare. Dopodiché il programma controllerà per ogni i-esima cella dell'array di celle se il colore delle 3 celle successive è uguale alla stessa, fermandosi alla quart'ultima.

Nel caso in cui non sia stata trovata alcuna mossa vincente verrà controllato se la tutte le colonne della tabella si sono riempite e in tal caso il programma dichiarerà che la partita è terminata in pareggio.

Altrimenti il gioco proseguirà normalmente richiedendo l'inserimento successivo.

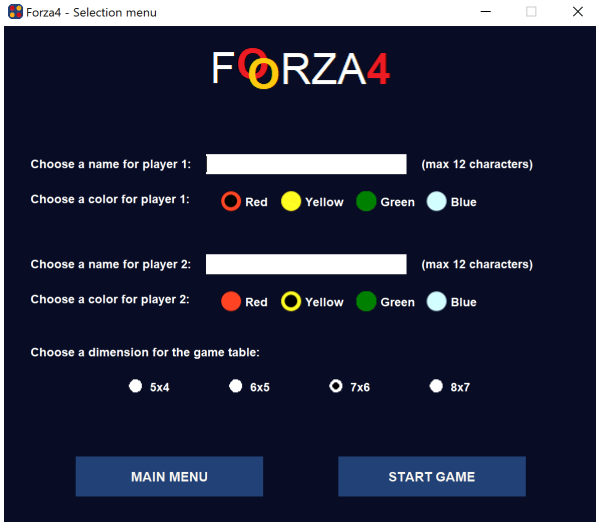
# 5 Manuale della GUI

## 5.1 Schermata iniziale




- **Creazione nuova partita**
  Premere il bottone "NEW GAME" per iniziare una nuova partita.
- **Caricare una partita precedente**
  Premere il bottone "LOAD GAME" per caricare una partita precedente.
- **Uscire dal gioco**
  Premere il bottone "QUIT" per terminare il programma.

## 5.2 Schermata di selezione

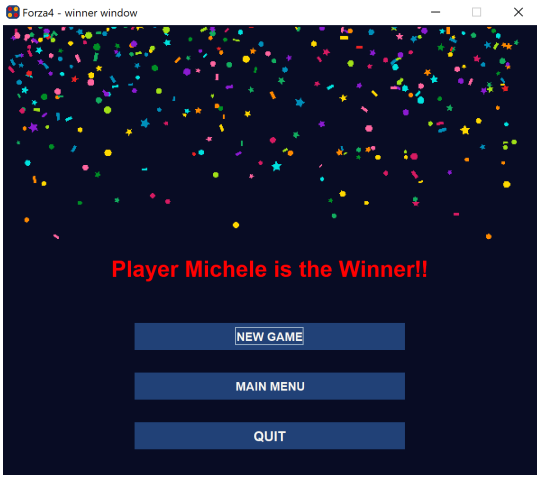


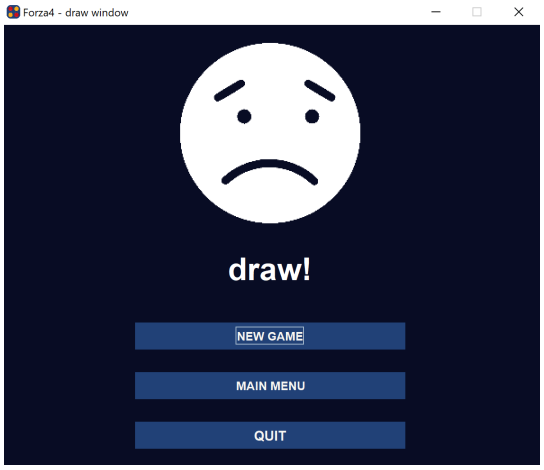

- **Campi di testo**
  Permettono di scegliere il nome dei giocatori.
- **Selezione dei colori**
  Quattro JRadioButton che permettono al giocatore di scegliere il colore tra uno dei seguenti: rosso, giallo, verde e blu.
- **Iniziare la partita**
  Premere il bottone "START GAME" per iniziare una nuova partita con le informazioni date in input.
- **Tornare al menu principale**
  Premere il bottone "MAIN MENU" per tornare al menu principale.

## 5.3 Schermata della partita


- **Nomi e colori dei giocatori**
  Nella parte superiore della schermata sono visibili il nome del giocatore e il colore da lui scelto.
- **Informazioni sul turno**
  Nella parte inferiore della tabella è visibile le informazione sul giocatore che deve effettuare la mossa.
- **Tabella di gioco**
  Nella parte centrale della schermata sarà visibile la tabella di gioco, verrà aggiornata ogni qual volta un giocatore compie una mossa.
- **Salvare la partita**
  Premere il bottone "SAVE GAME" per salvare la partita corrente.
- **Tornare al menu principale**
  Premere il bottone "MAIN MENU" per tornare al menu principale.

## 5.4 Schermata di vittoria o pareggio




- **Informazioni sul vincitore**
  Se vi è un vincitore nella parte centrale della schermata sono visibili le informazioni sul giocatore che ha vinto. Altrimenti si aprirà la schermata di pareggio.

- **Creazione nuova partita**
  Premere il bottone "NEW GAME" per iniziare una nuova partita.

- **Tornare al menu principale**
  Premere il bottone "MAIN MENU" per tornare al menu principale.

- **Uscire dal gioco**
  Premere il bottone "QUIT" per terminare il programma.