Relazione per

“Programmazione ad Oggetti”



A cura di:

Marconi Riccardo  
Prencipe Michele  
Valdiserri Sara

Anno Accademico 2019/2020

**Indice**

[**1 ANALISI** **2**](#_Toc9490)

[1.1 Requisiti 2](#_Toc9491)

[1.2 Analisi e modello del dominio 3](#_Toc9492)

[**2 DESIGN** **5**](#_Toc9493)

[2.1 Architettura 5](#_Toc9494)

[2.2 DESIGN DETTAGLIATO 7](#_Toc9495)

[2.2.1 Parte gestita da: Marconi Riccardo 7](#_Toc9496)

[2.2.2 Parte gestita da: Prencipe Michele Pio 10](#_Toc9497)

[2.2.3 Parte gestita da: Valdiserri Sara 11](#_Toc9498)

[**3 SVILUPPO** **13**](#_Toc9499)

[3.1 Testing automatizzato 13](#_Toc9500)

[3.2 Metodologia di lavoro 14](#_Toc9501)

[3.2.1 Marconi Riccardo 14](#_Toc9502)

[3.2.2 Prencipe Michele Pio 14](#_Toc9503)

[3.2.3 Valdiserri Sara 14](#_Toc9504)

[3.2.4 Parte di lavoro in cooperazione 14](#_Toc9505)

[3.3 Note di sviluppo 15](#_Toc9506)

[**4 Commenti Finali** **16**](#_Toc9507)

[4.1 Autovalutazione e lavori futuri 16](#_Toc9508)

[4.1.1 Marconi Riccardo 16](#_Toc9509)

[4.1.2 Prencipe Michele Pio 17](#_Toc9510)

[4.1.3 Valdiserri Sara 17](#_Toc9511)

[4.2 Difficolta` e consigli - Note per i docenti 18](#_Toc9512)

[**A Guida utente** **19**](#_Toc9513)

**Capitolo 1**

# ANALISI

Il software realizzato si pone l’obiettivo di realizzare una versione del gioco per smartphone: ”2048”.

Il gioco consiste nello spostare, unire e creare varie tessere all’interno di una griglia composta di tessere, il valore di queste è costituito da potenze del due ed il giocatore potrà spostare tutte le caselle in base alla direzione scelta.   
Se due caselle con lo stesso valore si scontrano mentre si muovono nella stessa direzione, si uniscono in un’unica casella che avrà come valore la somma delle due tessere ed il rispettivo colore.   
La partita è vinta quando, continuando a far sommare le tessere, si riesce a crearne una con il valore 2048.

Se il giocatore non può più muovere le caselle, perché non ci sono spazi vuoti o non ci sono tessere adiacenti con lo stesso valore, la partita finisce registrando un punteggio pari alla somma di tutte le unioni tra le tessere.  
L’utente potrà scegliere, oltre alla classica modalità di gioco, una variante con vari moltiplicatori, ossia bonus che lo aiuteranno nel raggiungimento della vittoria oppure malus che aumenteranno la difficoltà di gioco, ed un’altra modalità che permette l’utilizzo di altre tre diverse dimensioni della griglia.

## 1.1 Requisiti

**Requisiti funzionali**

L’applicativo deve permettere all’utente:

* Di poter scegliere tramite menu la modalità che preferisce tra quelle disponibili;
* L’interazione con la griglia tramite comandi da tastiera;
* Di calcolare il punteggio dinamicamente durante il corso della partita permettendone il salvataggio finale;
* Visualizzare i record dei vari giocatori secondo una classifica;
* Possibilità di decidere la dimensione della griglia di gioco.

**Requisiti non funzionali**

* Musica e animazioni saranno oggetto di studi futuri.

## 1.2 Analisi e modello del dominio

**Descrizione del modello del Dominio**

Il software dovrà essere in grado di modellare il dominio di una partita generando le entità utili (caselle, griglia, bonus, malus) e di spostarsi tra le varie sezioni del menu.

Le caselle, ciascuna con il proprio valore e il relativo colore, saranno direzionabili tramite input da tastiera mediante l’uso delle lettere A, S, D, W a seconda della direzione in cui l’utente vorrà muoverle, mentre la griglia conterrà delle celle che potranno essere vuote o con delle caselle.   
Uno dei principali problemi che l’applicativo dovrà gestire è la creazione delle singole caselle in maniera random ad ogni turno, con i valori più bassi delle potenze del 2 (2 e 4) e la gestione delle interazioni tra le varie caselle durante gli spostamenti all’interno della griglia.   
La risoluzione di tale problema consisterà nel gestire e controllare correttamente tutti i comportamenti possibili, e non, di ogni singola casella durante gli spostamenti.

Inoltre, sarà disponibile la scelta tra diverse tipologie di grandezza della griglia e una differente modalità di gioco in cui sono presenti bonus e malus. Una volta terminata la partita l’utente potrà salvare il proprio punteggio ed esso, se abbastanza alto da entrare, sarà aggiunto ad una classifica dei dieci

punteggi più alti che sarà consultabile al menu principale.

I requisiti non funzionali riguardanti l’implementazione di effetti sonori e animazioni saranno oggetto di studi futuri.

**Capitolo 2**

# DESIGN

In questo capitolo verrà mostrata una visione architetturale e il relativo funzionamento.

## 2.1 Architettura

In questo progetto abbiamo scelto di usare il pattern architetturale MVC (Model-View-Controller).

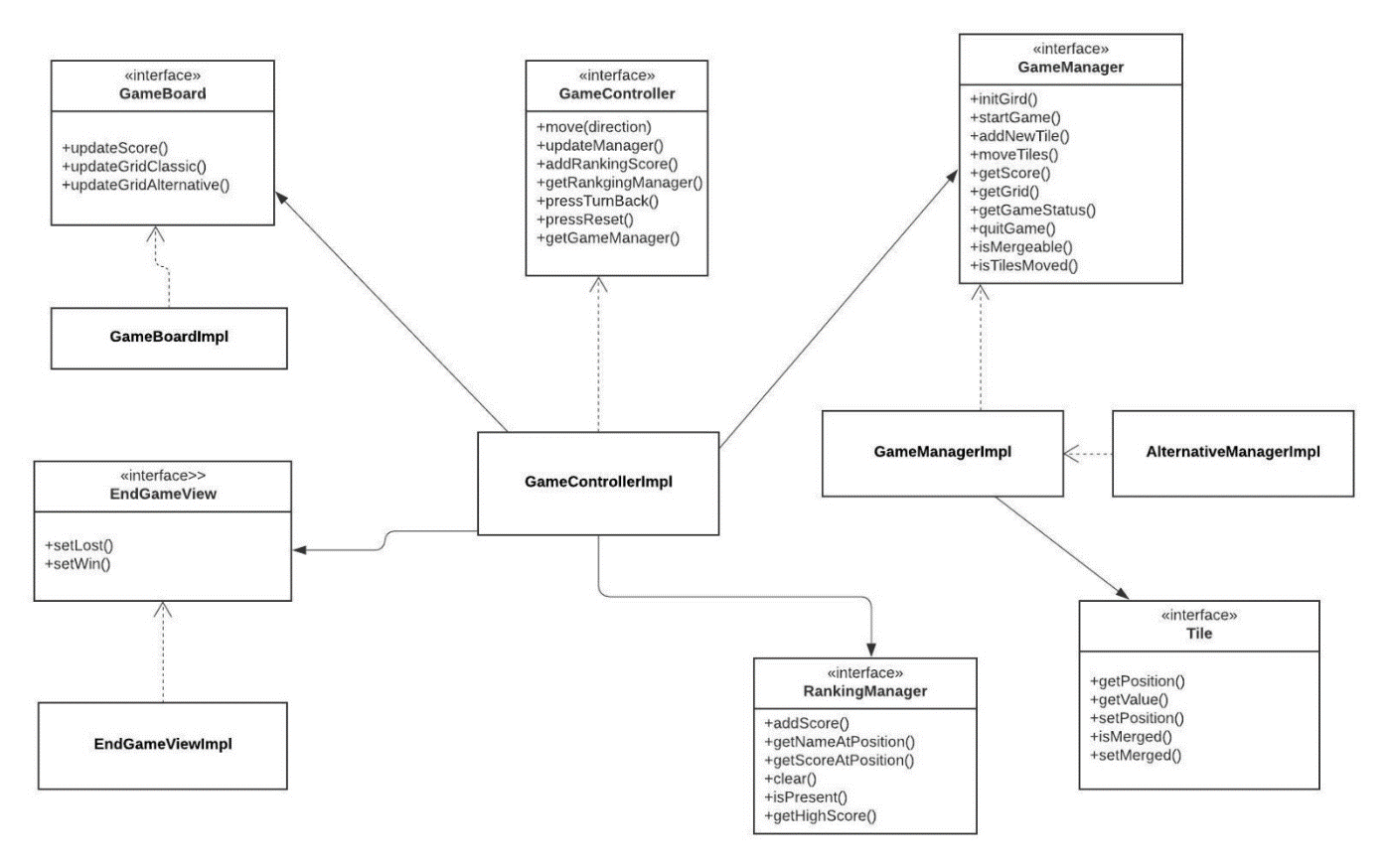
Tale scomposizione ci permette in un futuro di andare eventualmente a modificare o aggiungere nuovi elementi in maniera più rapida e intuitiva, inoltre ci ha consentito di separare la parte logica da quella grafica.

In particolare, il Controller crea gli oggetti necessari all’applicativo e si occupa di gestire le interazioni tra le varie entità, manipolando i dati del Model e notificando gli eventi scatenati dall’utente alla View.

Il Model si occupa della gestione della logica, nel nostro caso si occupa di implementare i comportamenti delle entità coinvolte.

La View e la rappresentazione grafica dei dati aggiornati.

Laddove possibile, abbiamo cercato di sviluppare delle interfacce al fine di separare al meglio la logica del dominio applicativo dalla sua implementazione.



## 2.2 DESIGN DETTAGLIATO

### 2.2.1 Marconi Riccardo

##### **ControllerScene**

MenuController, CustomSelectionController, HowToPlayController e ModeSelectionController sono tutte le classi che si occupano di gestire le varie scene, disegnate in formato FXML con Scene Builder, attraverso l’uso di bottoni.   
Ho deciso di creare un controller per ogni scena per poterle gestire meglio e per rendere più facile e veloce l’applicazione di modifiche alle varie classi, anche in un eventuale futuro nel qual caso lo si voglia.

MenuController gestisce le scelte dell’utente all’avvio dell’applicazione: permette di visualizzare il ranking, andare al menu di scelta della modalità di gioco e di poter visualizzare la schermata che delle regole del gioco.

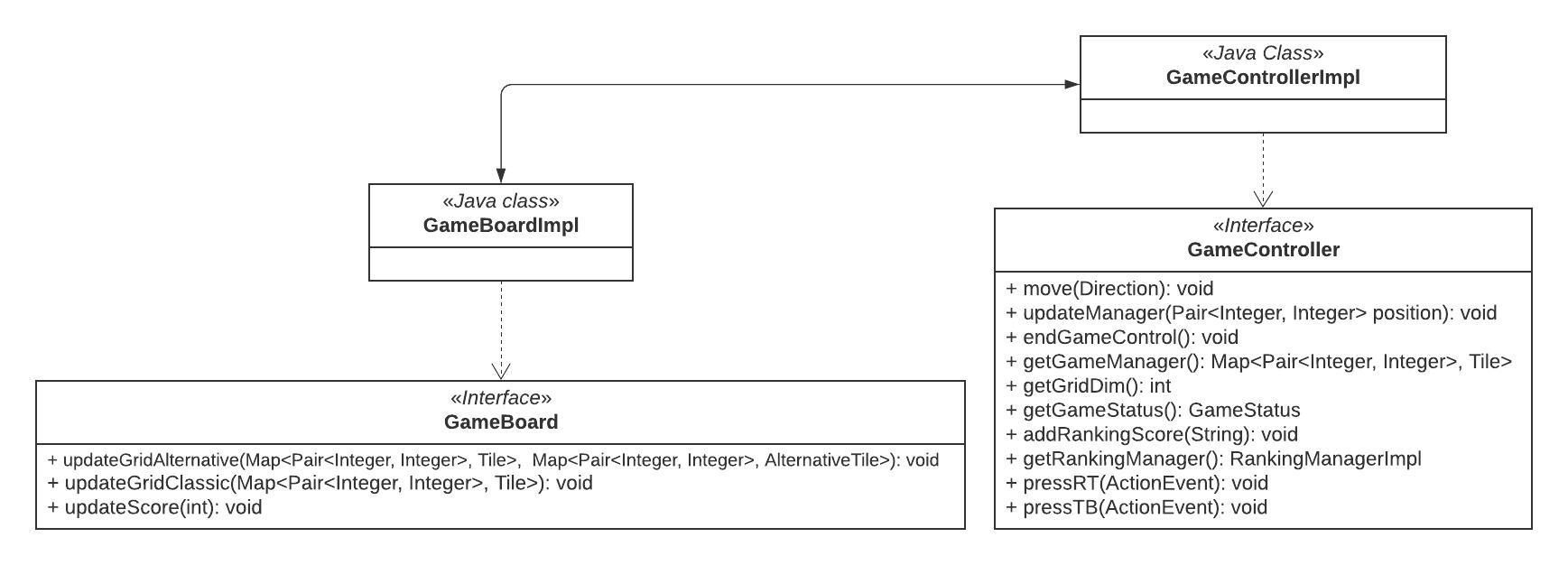
ModeSelectionController gestisce la scelta della modalità di gioco, infatti si potrà scegliere tra tre diverse possibilità: una partita classica, una con altre tre dimensioni selezionabili della griglia e l’ultima che permette di giocare una partita in modalità alternativa.

CustomSelectionController consente la scelta tra tre grandezze di griglia (5x5, 6x6, 7x7) e permette anche una scelta casuale.

HowToPlayController serve per poter tornare nel menu principale dopo che si sono visualizzate le regole di gioco.

**GameControllerImpl**Grazie a questa classe l’applicazione assume una conformazione architetturale del tipo MVC (model-view-controller).   
GameController inizialmente crea il model (a seconda della modalità di gioco), successivamente crea una nuova view passandole la griglia di gioco del model, la quale verrà disegnata e aggiunta alla scena.   
Inoltre, ascolta gli input da tastiera dell’utente facendo muovere la griglia del model secondo la direzione indicata, il controller riceve la griglia aggiornata che immediatamente passa alla view per aggiornare la GUI.   
Oltre agli input da tastiera il controller ascolta anche la pressione dei pulsanti “reset” e “turn back” che consentono rispettivamente di ricominciare una nuova partita e tornare al menu precedente.   
Quando la partita finisce, che sia una vittoria o una sconfitta, richiama la EndGameView che imposterà la scena di fine gioco.   
Il giocatore potrà salvare la partita con un nickname che verrà poi passato insieme al punteggio e alla modalità al RankingManager che gestirà la classifica.

##### **GameBoardImpl**

GameBoardImpl interpreta la View e non è altro che la rappresentazione grafica del modello con i vari elementi della schermata di gioco.   
Alla sua inizializzazione genera in automatico tutti gli elementi grafici della scena come lo sfondo, il titolo, i bottoni ed infine la griglia di gioco, che a seconda della taglia le sue celle vengono ingrandite o rimpicciolite.   
I metodi updateGridClassic e updateGridAlternative servono per aggiornarla a seconda della modalità di gioco: le vengono passate dal controller una mappa di coppie posizione-Tile, successivamente la view aggiunge nelle celle le Tile corrispondenti con i metodi addTile e addAlternativeTile.

##### Figura 2.2: Rappresentazione UML di Controller e View del pattern MVC.

##### **AlternativeManagerImpl**

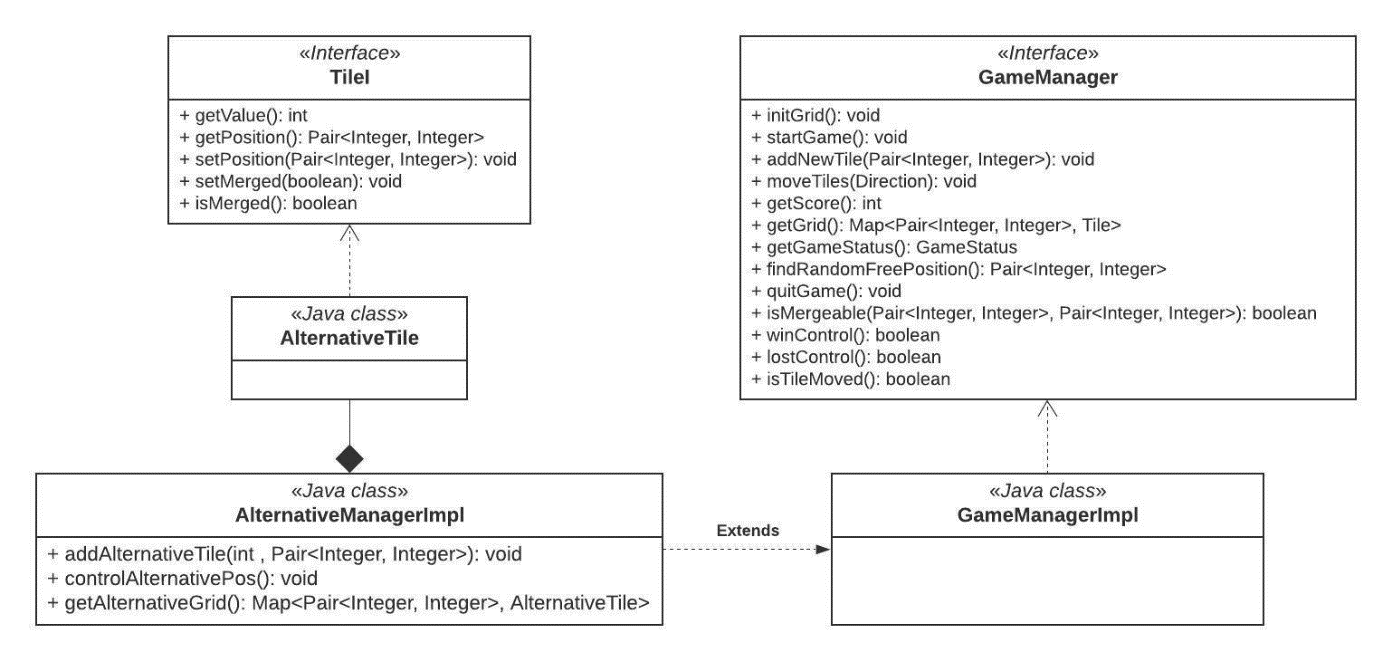
AlternativeManagerImpl viene inizializzata dal controller quando si vuole giocare una partita in modalità alternativa.   
Questa classe estendendo GameManager ha il vantaggio di ereditare i suoi metodi e quindi poterli utilizzare.   
Oltre a gestire la griglia di gioco grazie ai metodi di GameManager questa classe genera una nuova griglia di AlternativeTile, il metodo addAlternativeTile aggiunge a questa griglia un AlternativeTile, in una posizione non sia occupata da una Tile nella griglia delle Tile.   
Il metodo controlAlternativePos quando viene richiamato controlla le posizioni delle AlternativeTile e delle Tile: se due si trovano nella stessa posizione allora verrà generata una nuova Tile con il valore della vecchia moltiplicato per quello dell’AlternativeTile.  
Valendo solo 0 e 2 le AlternativeTile equivalgono ad un bonus che moltiplica per due, o a un malus che elimina qualsiasi cella.   
Infine, questa classe ritorna la griglia delle AlternativeTile che verranno poi disegnate dalla view.

Figura 2.3: Rappresentazione UML di AlternativeManagerImpl e delle componenti collegate.

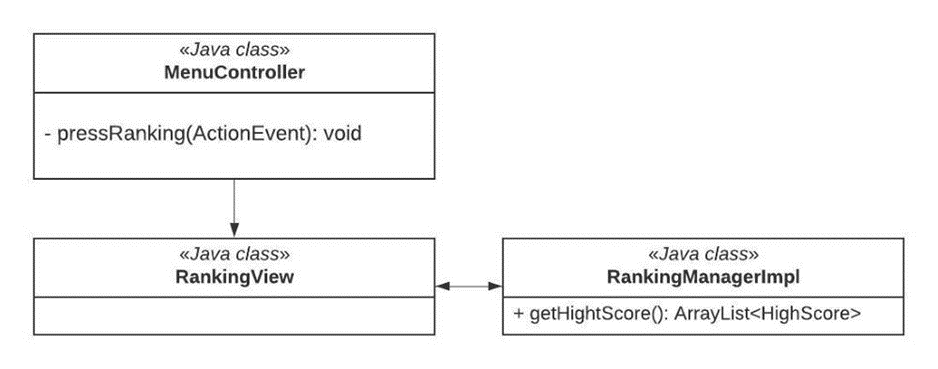
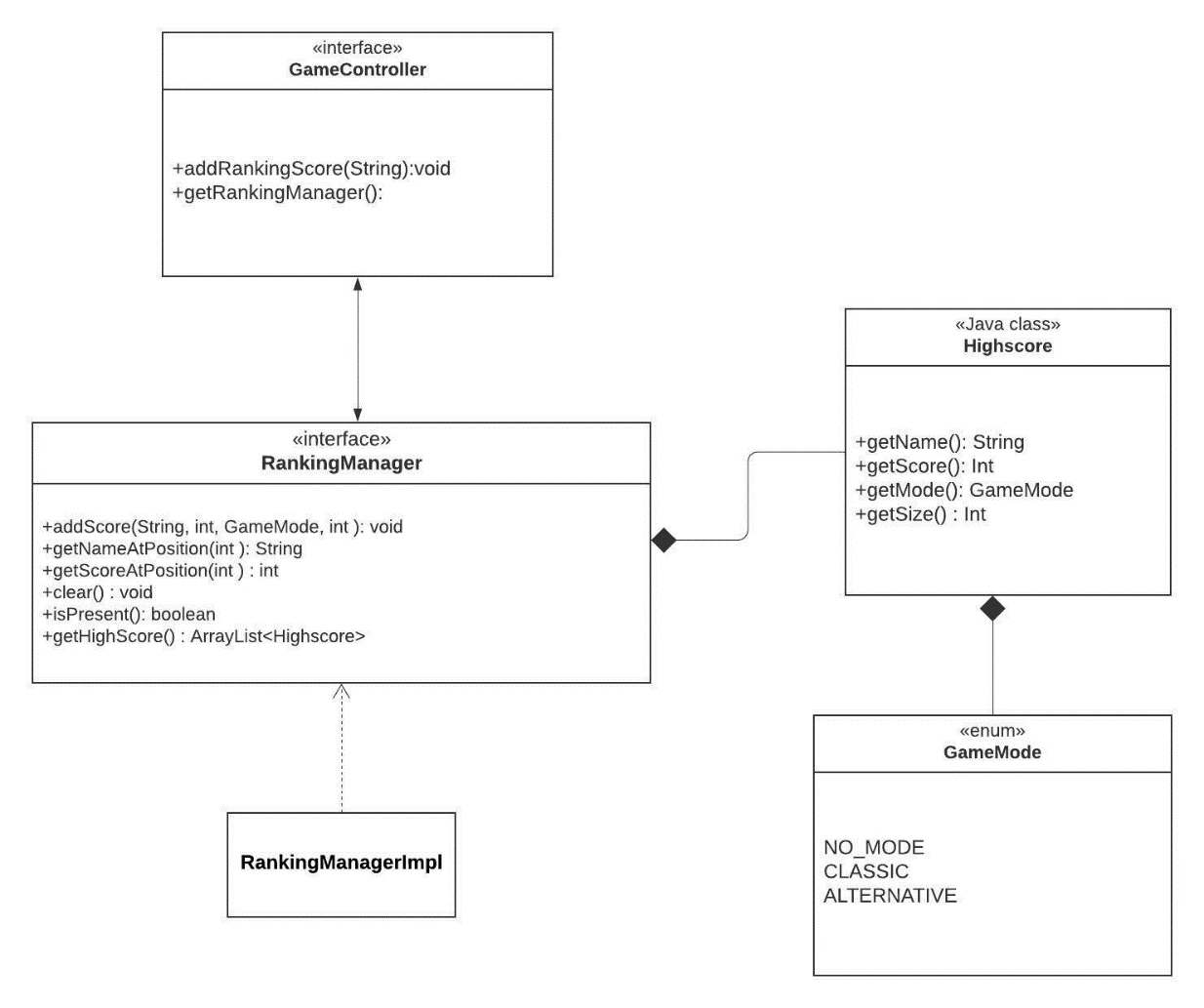
**RankingView**  
RankingView disegna la classifica delle dieci partite salvate con punteggio più alto, inserendo in ogni riga di una griglia nome, punteggio, modalità di gioco e la taglia della griglia di ogni singola partita e disegna anche un pulsante per tornare al menu principale. Questa classe ottiene una lista di tutti gli HighScore da RankingManager, e li va a scrivere nelle celle della griglia.

Figura 2.4: Schema UML di RankingView con componenti collegate.

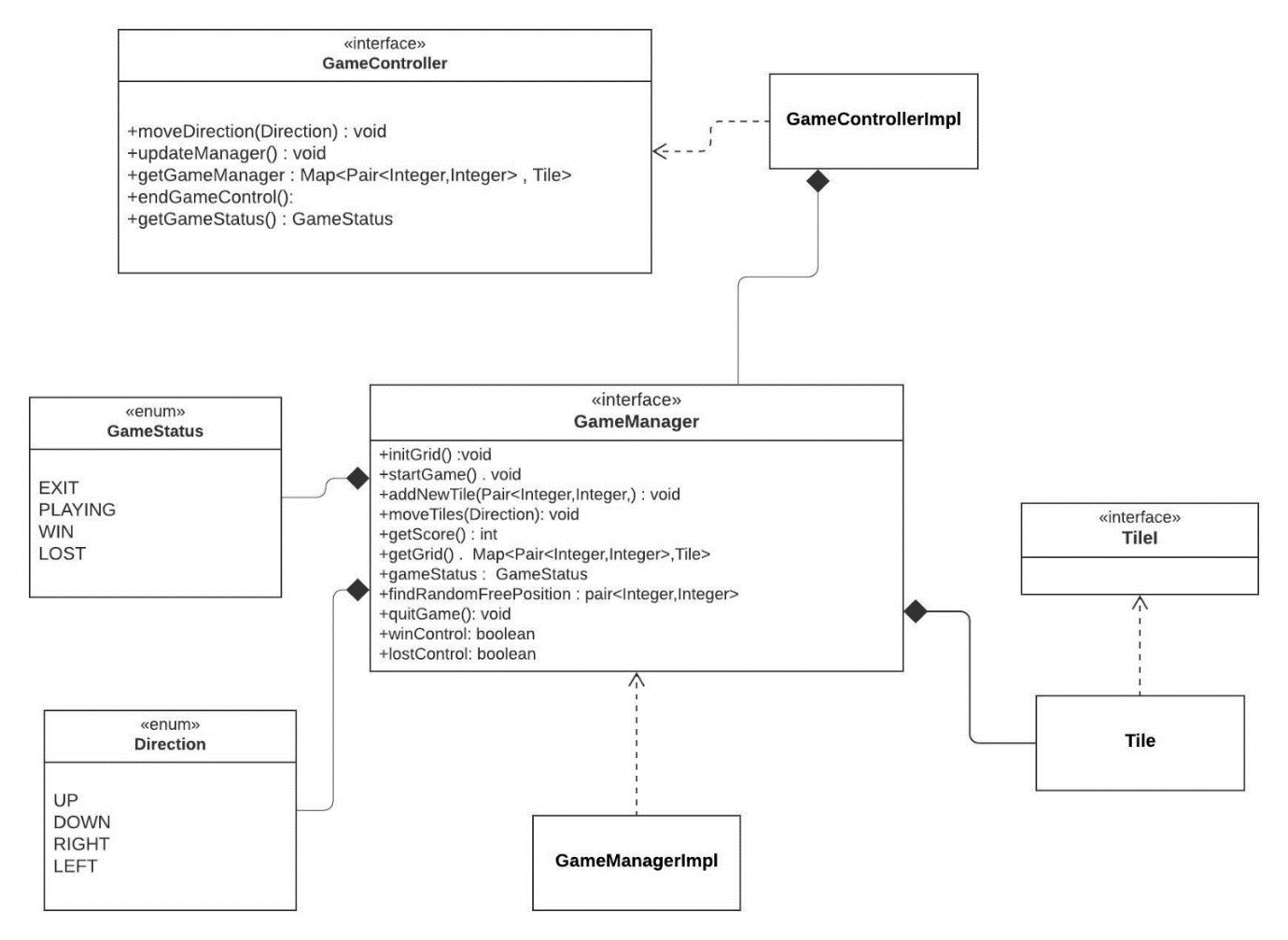
**2.2.2 Prencipe Michele Pio**

##### **Ranking**

 In questo ambito si vogliono memorizzare i dati relativi ai punteggi dei giocatori che finiscono una partita. Implementando l’interfaccia RankingManager vengono usati vari metodi sia per la creazione e l’aggiornamento del file sia per il caricamento da quest’ultimo dei punteggi precedenti.   
In modo tale da salvare i punteggi di tutti i predecessori in maniera duratura nel tempo e si possano recuperare i vecchi risultati anche se il gioco viene chiuso.   
La classe utilizza anche un oggetto di tipo HighScore che associa un punteggio, una modalità, una grandezza della griglia e una posizione in classifica ad un determinato giocatore.   
L’ordinamento delle posizioni nella classifica finale viene gestito tramite un Comparator.   
In futuro si potrebbe anche gestire differentemente la comparazione di Score in quanto si potrebbe aggiungere anche una variante a tempo del gioco.

**GameManager**

In questo ambito si sviluppa la logica di gioco composta da movimenti delle Tile, unioni, aggiornamento del punteggio e vari check sulla vittoria e sulla sconfitta. Il movimento non è banale infatti richiama diversi controlli alla fine dei quali una tessera raggiunge la propria posizione corretta oppure si è unita con un’altra tessera per formarne una nuova col valore successivo.   
Alla fine del movimento la griglia sarà aggiornata con le posizioni nuove per ogni tessera e verrà aggiunta una nuova Tile in una posizione random libera.   
Quando il turno è terminato si procede con l’aggiornamento del punteggio determinato da tutte le caselle create dopo un’unione.  
Lo scopo di GameManager è quello di tener sempre aggiornata la griglia composta da vari elementi, che verrà manipolata dal Controller, il quale riceve eventi da parte dell’utente come la direzione in cui le tessere si devono muovere. Il GameManager restituisce al Controller anche uno stato di avanzamento della partita che potrà essere:

PLAYING  
WIN  
LOST  
EXIT

### 2.2.3 Valdiserri Sara

##### **Tile**

Questa classe rappresenta l’entità che sta alla base di tutto il gioco sviluppato nell’applicazione.

Essa infatti descrive come è composta e quali sono le caratteristiche dell’entità della singola casella.

Viene richiamata in primis ogni volta che viene creata una nuova griglia all’inizio di ogni gioco creando così le prime due caselle che avranno come valore due o quattro, ovvero le potenze più basse del due.

Successivamente verrà richiamata ogni volta che vi è un movimento delle caselle in maniera da poterne aggiungere una nuova.

La scelta del valore che avrà la nuova casella è gestita in maniera random dal metodo newRandomtile.

L’entità della casella è costituita dalla sua posizione all’interno della griglia espressa attraverso una coppia di coordinate e un valore numerico delle potenze del due.

Il valore di ogni casella non è immutabile ma cambia nel momento in cui due caselle con la stessa potenza del due si trovano vicine orizzontalmente o verticalmente e viene effettuato dall’utente uno dei due movimenti possibili lungo tale asse, in questo caso le due caselle si fondono in una sola che avrà come valore la potenza del due successiva a quella del valore delle due caselle.

##### **TileView**

Essa rappresenta la grafica dell’entità Tile, ovvero come viene visualizzata ogni singola casella.

Per ogni tipologia di griglia scelta tale classe dovrà essere in grado di adattare la dimensione delle caselle a quelle della griglia.

Ogni casella visualizzerà al suo interno il valore della potenza del due contenuto in essa.

Ad ognuna delle potenze del due è associato un diverso colore.

##### 

##### Figura 2.7: schema UML che rappresenta la gestione dell’entità Tile e di come sono collegate tra di loro le varie parti di essa.

##### **EndGameView**

EndGameView va ad aggiungersi all’implementazione delle altre scene del gioco.   
Essa rispetto alle altre rappresenta la parte finale del gioco, infatti verrà richiamata dall’applicativo solo nel momento in cui il giocatore vince la partita, raggiungendo il valore 2048, oppure nel momento in cui non potranno più essere effettuati movimenti e si avrà il classico ’Game Over’.

Per capire in quale di questi due casi l’utente ha finito il gioco all’interno della classe vi sono due diversi metodi e in base a quale di esso viene richiamato verrà espresso se l’utente ha vinto o ha perso.   
Viene riconosciuto quale dei due metodi chiamare in base allo stato del gioco, ovvero se esso è impostato su WIN o LOST.

Altre caratteristiche, come i bottoni che consentono di tornare alla scena precedente (che sia ricominciare da capo la partita o cambiare modalità tornando indietro al menu) rimangono uguali per entrambi i casi. In tale scena ci si occupa inoltre di far fare all’utente il ’login’, l’utente potrà inserire il proprio username decidendo così di salvare su file il proprio punteggio ottenuto in tale partita.

Tale punteggio associato al relativo username potrà in seguito essere consultato nella classifica all’intero della Ranking.

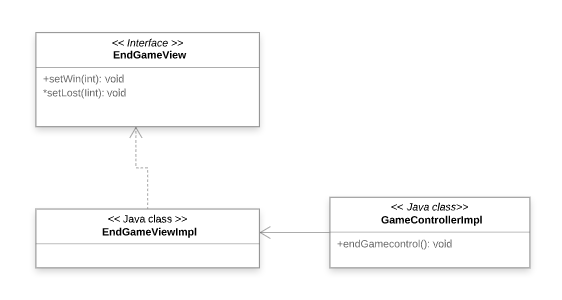
****

Figura 2.8: schema UML della rappresentazione della schermata di fine gioco.

**Capitolo 3**

# SVILUPPO

## 3.1 Testing automatizzato

Il testing automatizzato viene fatto in JUnit per verificare il corretto funzionamento di ogni classe, specialmente delle classi appartenente al Model e di calcoli randomici che non prevedono l’iterazione corrente.

Durante lo sviluppo dei metodi delle varie classi ne abbiamo verificato il corretto funzionamento.

In particolare, abbiamo effettuato i test relativi ai metodi della classe GameManager, ovvero la classe che gestisce le principali interazioni tra le entità`. I test effettuati riguardano la creazione di una nuova griglia, il movimento delle caselle, l’aggiornamento dello score, l’unione tra due caselle e i casi di vittoria/sconfitta.

#### *NOTE* : (*Valdiserri Sara*)

Inizialmente era stata adottata la suddivisione dei test in Test Case e Test Suite dove ogni Test Case `e associata ad una singola classe, ovvero quella che si vuole testare, mentre la Test Suite la possiamo immaginare come l’insieme di tutte le Test Case. La classe extends Test Suite generalmente `e individuata dal nome ”AllTest”, e contiene i riferimenti alle singole Test Case. Successivamente si `e deciso di mostrare la classe a cui principalmente ci siamo dedicati per quanto riguarda la parte dei test e quindi `e stato optato per non suddividere i Test nella modalità prima descritta.

## 3.2 Metodologia di lavoro

Abbiamo mantenuto senza troppe modifiche la suddivisione dei lavori iniziale, suddividendo chiaramente tutti i vari compiti.

Di seguito vi `e la suddivisione per ogni componente:

### 3.2.1 Marconi Riccardo

* Creazione e gestione menu principale, con relativa scelta delle modalità di gioco; gestione finestra d’inizio gioco;
* gestione dell’unione di due caselle (merge tra due Tile).

### 3.2.2 Prencipe Michele Pio

* Gestione movimento delle caselle;
* gestione e punteggio della classifica finale.

### 3.2.3 Valdiserri Sara

* creazione di nuove caselle (entità Tile);
* gestione vittoria/sconfitta (gestione metodi per controllare se si `e in uno di questi due casi e conseguentemente apertura e gestione della schermata di fine gioco).

### 3.2.4 Parte di lavoro in cooperazione

In alcune parti del progetto è stato necessario cooperare.

Abbiamo utilizzato DVCS Git legato ad un repository su BitBucket.  
Ogni componente ha lavorato in locale caricando volta per volta su tale repository il codice modificato.

* Fase finale in cui è stato effettuato anche il controllo di eventuali errori;
* interazione tra le varie parti sviluppate dai singoli componenti del gruppo, al fine di rendere l’applicativo funzionante nel suo complesso.

## 3.3 Note di sviluppo

È stato fatto uso della libreria JavaFX da tutti i membri del gruppo, in quanto utile a realizzare il progetto nella sua interezza.

#### Marconi Riccardo

Nella mia parte di progetto ho utilizzato il software Scene Builder da Eclipse per realizzare i file FXML rappresentanti le finestre dell’applicazione dei vari di menu in cui ci si può spostare, mentre ho realizzato la view delle partite tramite codice utilizzando la libreria JavaFX.   
Per comprendere come poter utilizzare al meglio questi strumenti mi sono documentato su siti come Oracle, Stack Overflow, forum online e le slide fornite dal corso.

#### Prencipe Michele Pio

*Stream*: utilizzati principalmente per le iterazioni sulle collezioni di dati, soprattutto la griglia, in quanto permettono di manipolarle e filtrarle con semplicità garantendo una scrittura del codice compatta.

*Lamda expression*: utilizzate in combinazione agli stream per avere un codice compatto e intuitivo.

*Optional*: utilizzati principalmente per verificare la presenza di Tile o meno in una posizione.

Onestamente, per alcuni metodi in cui ero particolarmente bloccato ho preso spunto da alcuni codici su Github e StackOverflow cercando comunque di reinterpretarli e inserirli nel nostro contesto. Anche le slide fornite durante il corso si sono dimostrate particolarmente utili per il settaggio dell’ambiente di sviluppo e per alcuni metodi non banali.

**Valdiserri Sara**

Nella mia parte per quanto riguarda la scelta dell’utilizzo del software per la scena finale ho deciso di non optare per la creazione di un FXML, in quanto essa è una scena che si va ad aggiungere alle altre solamente nel caso di una partita terminata, in caso contrario tale scena non verrà creata.

L’intero progetto è stato inoltre realizzato facendo uso della libreria JavaFX.

**Capitolo 4**

# COMMENTI FINALI

## 4.1 Autovalutazione e lavori futuri

### 4.1.1 Marconi Riccardo

Nella realizzazione del progetto ho riscontrato diverse difficoltà, una di queste è stata l’utilizzo del software SceneBuilder mai utilizzato prima, ma tramite personali ricerche e approfondimenti sono riuscito a capirlo e utilizzarlo in modo a me soddisfacente.   
Altra difficoltà è stato capire inizialmente il giusto utilizzo di Git, software fondamentale il quale approfondirò sicuramente in futuro per poterne imparare le molte altre complesse funzionalità delle quali è dotato, ma già dai primi frequenti utilizzi e grazie al lavoro di squadra è risultato più facile da utilizzare per le nostre più semplici necessità.   
Un aspetto che personalmente ho trovato sfavorevole è stato lavorare solamente a distanza con i membri del gruppo, senza potersi confrontare personalmente ma solo tramite chiamata vocale o messaggi, questo ci ha rallentato molto nel prendere decisioni o comprendere e risolvere alcune problematiche riscontrate durante lo sviluppo.   
Questo progetto mi ha aiutato molto a comprendere come lavorare e comunicare con i membri di un gruppo, inoltre ha aumentato le mie capacità poiché ho dovuto apprendere diverse cose per poterlo svolgere al meglio.   
Sono consapevole che alcune classi potrebbero essere implementate in un modo migliore, ma sono comunque soddisfatto di ciò che ho realizzato, soprattutto per aver rispettato il tempo limite prestabilito di consegna e per aver superato le difficoltà riscontrate.

### 4.1.2 Prencipe Michele Pio

Lo sviluppo di questa applicazione è stato molto arduo ed intenso in quanto era la prima volta che approcciavo ad un progetto di gruppo di questa portata, ma alla fine posso ritenermi abbastanza soddisfatto del risultato ottenuto.   
Tuttavia, la mia parte di codice potrebbe non essere perfetta e magari migliorabile in futuro, ma penso di aver raggiunto tutti gli obiettivi che mi ero prefissato per questo progetto.   
Infatti, l’aver portato a termine l’applicazione e l’aver rapportato le mie idee con quelle degli altri membri della squadra, ha aggiunto al mio bagaglio personale un’esperienza non banale che potrà tornarmi utile in un futuro sia in ambito lavorativo che per altri progetti simili.   
Il gruppo è stato abbastanza disponibile, e io ho cercato di ricambiare con consigli riguardanti alcune scelte implementative.   
Ho capito da questa esperienza che la programmazione non è solo scrittura di codice, ma anche e soprattutto progettazione senza la quale sia il codice potrebbe risultare scollegato.   
Inoltre, ho cercato di studiare anche le librerie come JavaFX usate dagli altri membri del gruppo al fine di ottimizzare l’interazione tra le varie classi.

Per quanto riguarda le modifiche future da apportare al progetto mi piacerebbe inserire sia le animazioni sia gli effetti sonori e migliorare alcune parti del mio codice.   
Tuttavia, non penso di andare avanti in questo progetto ma preferirei cimentarmi in altri a me utili.

### 4.1.3 Valdiserri Sara

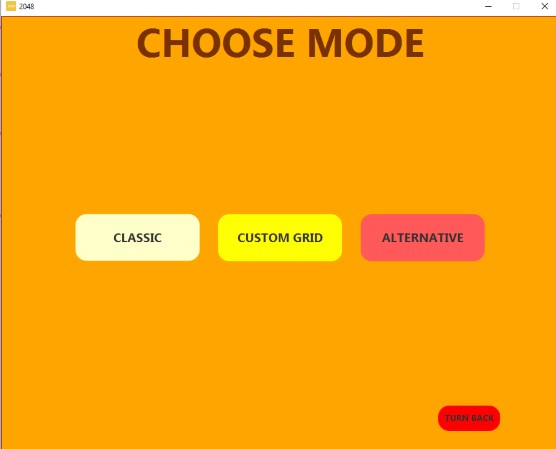
Realizzare questo progetto è stato utile non solo a poter crescere a livello di conoscenze di programmazione ma anche a capire e comprendere le difficoltà che stanno dietro quello che molte persone utilizzano nella loro quotidianità dandolo per scontato.   
Inizialmente la difficoltà più grande è stata imparare a fare uso di un linguaggio di programmazione a me nuovo, in quanto prima di questo corso non l’avevo mai approcciato.   
Sono consapevole che il codice da me sviluppato potrebbe non essere perfettamente corretto ma sono complessivamente contenta di quello che ho prodotto e di come esso mi sia servito a comprendere sempre di più il linguaggio Java.   
Un’altra difficoltà si è posta a noi nel momento in cui l’unico modo di poter cominciare è stato attraverso l’uso della tecnologia stessa, ammetto che lavorare e cooperare insieme in presenza, a parer mio, è sempre più efficace; però tale periodo e tale situazione ci ha anche dimostrato l’importanza di come il nostro codice debba essere scritto in maniera da essere compreso da altri informatici.   
Tale metodologia di lavoro penso sia molto utile per il nostro futuro nel momento in cui andremo a lavorare, e ci troveremo a doverci rapportare con altri informatici.

## 4.2 Difficoltà e consigli - Note per i docenti

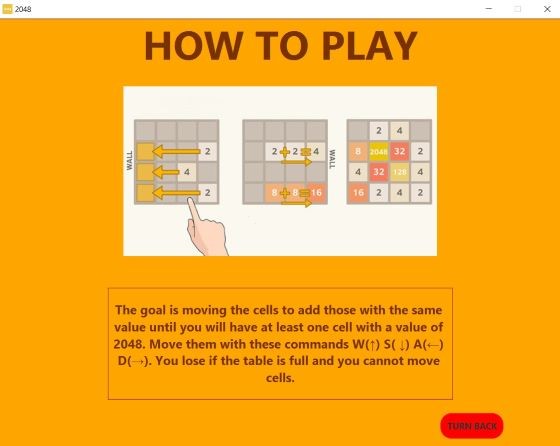
Per via dell’emergenza di quest’anno ci siamo riuniti sempre via web. Questo ha reso più difficoltosa l’organizzazione delle riunioni per via degli stravolgimenti delle routine quotidiane.   
Questa difficoltà ha accomunato tutti i membri del gruppo.

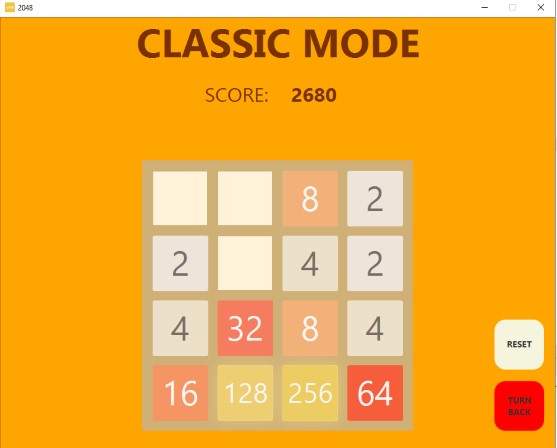
**Appendice A**

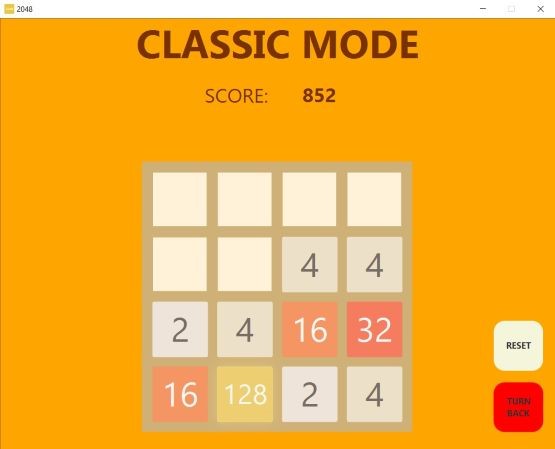
# Guida utente

Per iniziare la partita è necessario selezionare dalla schermata iniziale la modalità di gioco cliccando su MODE SELECTION.  
Una volta che si è aperta tale schermata compare un menu attraverso il quale verrà scelta l’effettiva modalità.

Schermata di selezione della modalità.

La figura mostra come giocare.





Alcuni esempi di possibili partite.