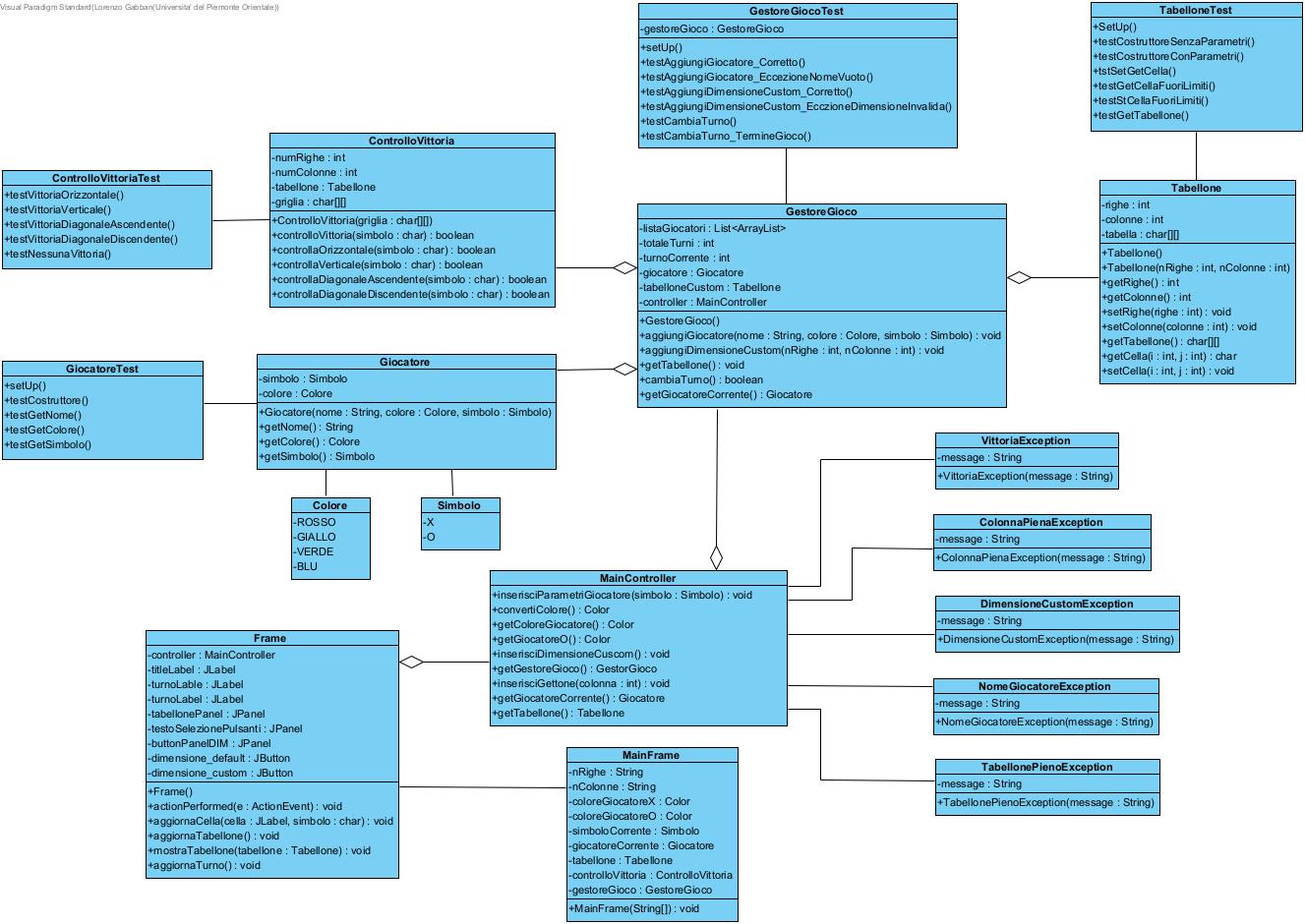
**Progetto Paradigmi di Programmazione**

****

VISTA:

* La classe **“Frame”** e’ la interfaccia principale del gioco di Forza4; la prima schermata che ci viene mostrata al suo avvio ci chiede di inserire il nome dei 2 giocatori e selezionare il loro annesso colore, in questa sezione sono presenti controlli tali per cui non e’ possibile lasciare vuoto il campo nomeGiocatore. In seguito possiamo selezionare il campo di gioco, se default o di dimensione custom, entrambe le scelte sono effettuate attraverso 2 pulsanti dedicati. Se si sceglie una dimensione custom sono presenti controlli tali per cui i parametri di tale dimensione possono esclusivamente essere compresi tra 4 e 10. Una volta scelto il tabellone si avvia il gioco effettivo: per ogni colonna e‘ presente un bottone alla cui pressione si inserisce il proprio gettone nella relativa colonna. Ad ogni inserimento da parte di un giocatore si controlla se e’ possibile inserire tale gettone e se la vittoria avviene con tale mossa. Il gioco termina alla vittoria di un giocatore o al riempimento del tabellone
* La classe **“MainFrame”** contiene il metodo principale del gioco, avvia la interfaccia grafica attraverso il richiamo del costruttore di Frame, si avvia il gioco attraverso il metodo main.

CONTROLLO:

* La classe “**MainController”** gestisce tutte le interazioni tra utente e gioco; e’ responsabile della comunicazione tra le componenti della vista e del modello. Nello specifico interfaccia il giocatore con i comandi di inserimento di un nuovo giocatore, inserimento della dimensione del tabellone se custom, inserimento del gettone ad ogni turno e un insieme di funzioni secondarie non visibili all’utente, quali gestione dei turni e dei parametri.

MODELLO

* La classe “**GestoreGioco**” e’ sicuramente la classe principale del package, richiama funzioni presenti nelle altre classi dello stesso e applica a livello logistico quanto richiesto dal Controller.
* La classe “**Giocatore**” contiene tutti i parametri e le funzioni relative alla creazione e manipolazione di giocatori, ogni giocatore viene definito con un suo nome, colore e simbolo. Si noti che il simbolo del giocatore non puo’ essere scelto da quest’ultimo ma viene inserito automaticamente.
* La classe “**Tabellone**” contiene tutti i parametri e le funzioni relative alla creazione e manipolazione del tabellone di gioco, logisticamente rappresentato da una matrice di caratteri, sono presenti 2 costruttori, rispettivamente utilizzati in base alla scelta utente di dimensione standard o custom. Permette di modificare singoli parametri e celle dello stesso.
* La classe “**ControlloVittoria**” viene invocata ad ogni inserimento di un gettone da parte del giocatore, in base al suo esito termina o meno la partita. Controlla in ogni direzione possibile la potenziale vittoria attraverso funzioni dedicate per ciascun senso (orizzontale, verticale, diagonale ascendente e discendente).

TEST

* La classe “**TabelloneTest**” definisce test per la classe Tabellone, relativi ad entrambi i tipi di costruttore, tentativi di richiamo di celle valide e non, metodi get.
* La classe “**GiocatoreTest**” definisce test per la classe Giocatore, relativi al costruttore e tutti i metodi get.
* La classe “**ControlloVittoriaTest**” definisce test relativi alla classe ControlloVittoria, relativi alla vittoria in ogni possibile direzione ed al caso in cui la vittoria non avvenga.
* La classe “**GestoreGiocoTest**” definisce test per la classe GestoreGioco, relativi alla aggiunta di un giocatore e di un tabellone custom, sia con parametri validi che non e metodi per la gestione dei turni.

ECCEZIONI

* La classe “**ColonnaPienaException**” viene invocata nel caso si cerchi di inserire un gettone in una colonna piena, non in grado di ospitare un ulteriore gettone, notifica all’utente l’impossibilita’ di terminare l’operazione richiesta.
* La classe “**DimensioneCustomException**” viene invocata nel caso il giocatore tentasse di creare un tabellone custom con righe e/o colonne superiori a 10 o inferiori a 4.
* La classe “**NomeGiocatoreException**” viene invocata nel caso si cercasse di lasciare vuoto il campo nomeGiocatore all’inserimento di un giocatore ad inizio partita.
* La classe “**TabellonePienoException**” viene invocata nel momento in cui non sono piu’ possibili ulteriori mosse, scatta nel momento in cui si concludono nRighe\*nColonne turni totali.
* La classe “**VittoriaException**” viene invocata nel momento in cui un giocatore vince mettendo in fila 4 gettoni in una qualsiasi direzione (orizzontale, verticale, obliquo), dopo la sua invocazione la partita termina.

|  |  |
| --- | --- |
| Test per le classi del modello | Fatto |
| Javadoc per i componenti pubblici del modello | Fatto |
| Il modello non fa alcuna lettura/scrittura da console o altra interazione con l’utente | Fatto |
| È stata usata l’ereditarietà ovunque appropriato | Fatto |
| È stato riutilizzato il codice ovunque appropriato | Fatto |
| Sono state utilizzate visibilità appropriate | Fatto |
| Sono state lanciate eccezioni ove appropriato | Fatto |
| Sono state usate classi annidate ove appropriato | Fatto |
| Specificare dove/come è stata fatta l’estensione di una classe o classe astratta | Estensione di Exception per la implementazione delle eccezioni gestite. |
| Specificare dove/come è stato usato il lancio/gestione di un’eccezione | Implementazione di eccezioni all’interno di un package dedicato, try effettuati in Frame e throws delle eccezioni all’intero di MainController e GestoreGioco. |
| Specificare dove è stata usata una classe annidata | Definizione delle classi annidate di tipo enumerativo Simbolo e Colore all’interno della classe Giocatore. |