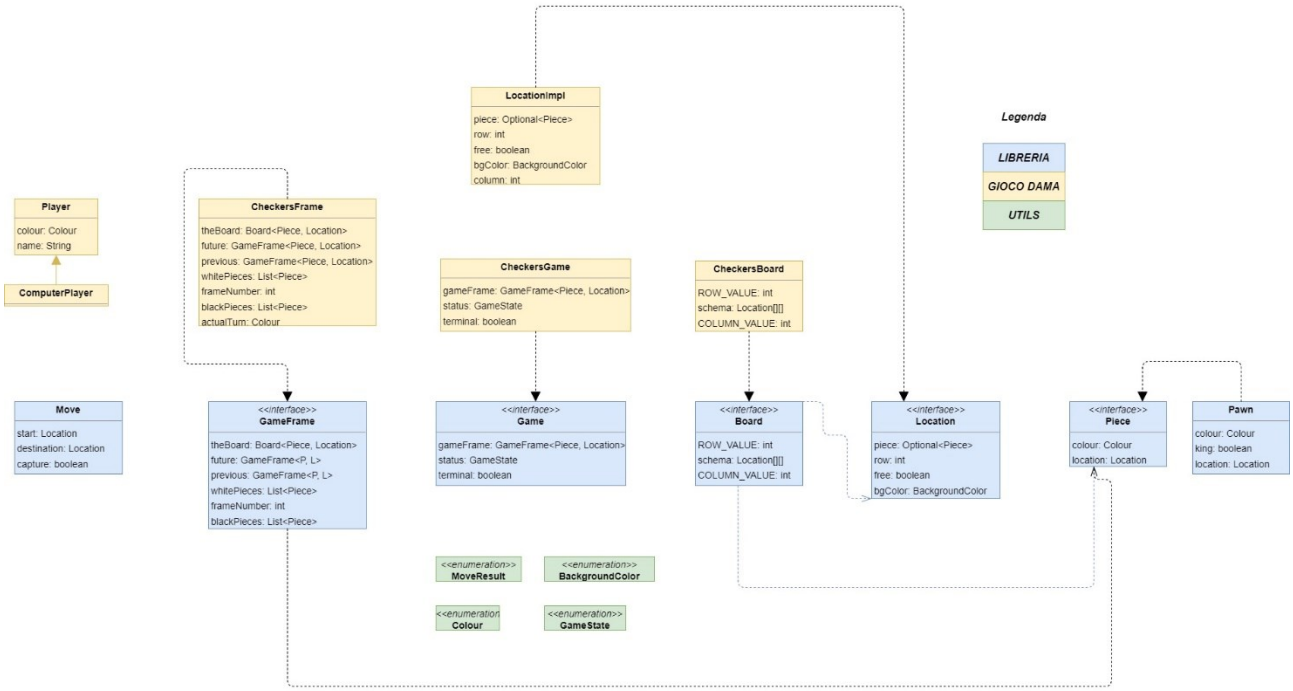
LORIS E. MINETTI 115346

RELAZIONE 'PROGETTO PROGRAMMAZIONE AVANZATA' – FEBB./MAR. 2023

Responsabilità principali:

- Definire un sistema di coordinate (che chiameremo per comodità griglia).
- Tenere traccia di oggetti all'interno di questa griglia, potendo conferire ad ogni coordinata un oggetto o meno, dunque un ente abile di esserci – non esserci - spostrarsi.
- Tenere traccia di tutte le possibili disposizioni di oggetti in relazione alla griglia.
- Definire il concetto di mossa.
- Definire il comportamento di ogni elemento, se questi potrebbero avere multipli comportamenti (esempio pedina e re nel gioco della Dama italiana)
- Tenere traccia la modalità di movimento per ogni elemento, all'interno della griglia, in un determinato momento.
- Tenere traccia in ogni momento lo stato della partita, ovvero come susseguirsi di disposizioni degli elementi nella griglia fino alla disposizione terminale della partita, la quale ha la responsabilità di terminarla.

UML class diagram



In questo diagramma non sono presenti i metodi relativi ad ogni classe. In ogni modo è visibile la dipendenza delle classi relative al gioco della Dama verso le classi di libreria.

Implementazione delle responsabilità con classi ed interfacce

Interfacce: queste definiscono i tasselli fondamentali per una libreria che permette di giocare a giochi per scacchiera.

Piece *:* interfaccia che definisce il concetto di pezzo ovvero, un elemento che si trova all'interno di una coordinata. Fornisce un set di comportamenti utili ad un pezzo qualsiasi, come recuperare il giocatore propietario di quel pezzo e la sua coordinata.

Location : definisce il concetto di posizione/coordinata all'interno di un sistema di coordinate. E' stato concepito in un contesto bi-dimensionale.

Board : fornisce un set di comportamenti fondamentali ovvero tutte quelle informazioni che una Location o un Piece possono avere trovandosi all'interno della Board: ad esempio scoprire i proprio vicini, scoprire in base ai propri vicini quali movimenti sono effettuabili, scoprire tutti i movimenti possibili per ogni giocatore… ha informazioni implicite che serviranno poi al gioco per poter effettuare movimenti e/o la gestione dei pezzi per ogni giocatore. Dunque è abile di far cooperare le "entità atomiche" che la compongono.

GameFrame : fornisce un set di comportamenti fondamentali per lo sviluppo della partita. Chiunque implementasse questa interfaccia ha le possibilità di gestire multiple informazioni in una partita basata su scacchiera, ed è strutturata in modo da poterlo fare per ogni istantanea di gioco, o meglio per ogni turno. Dunque operazioni come sapere tutte le mosse disponibili attualmente per un giocatore, o per uno specifico pezzo di quel giocatore.

Game : fornisce i comportamenti tipici che un gioco dovrebbe fare su se stesso, ovvero aggiornarsi, effettuare effettivamente una mossa tenendo sempre in considerazione ogni singolo GameFrame sul quale sta operando, e verificare ogni volta se può continuare o fermarsi per una vittoria o pareggio. È dunque un contesto nel quale poter gestire più GameFrame.

Classi:

LocationImpl*:* classe che definisce le caratteristiche di una coordinata. Oltre a ascissa ed ordinata, per poter essere localizzata in un sistema di coordinate, contiene anche alcuni indicatori per recuperare informazioni critiche in un contesto come un gioco di dama, come: un booleano che mostra se la Location è libera, un BackGroundColor che indica di che colore è la casella ed il pezzo che quella Location contiene, se ne contenesse uno

Move*:* Classe che definisce le caratteristiche atomiche del concetto di mossa, come spostamento di un oggetto da una Location ad un'altra, con la conseguente causa. Ovvero quello di togliere un pezzo da una location ed impostarlo in un'altra location, rendendo 'vuota' la location di partenza. Contiene un'informazione aggiuntiva che indica se una mossa sia "catturante" o meno. *[Move potrebbe sembrare una classe da inserire nella libreria, ma alcune sue caratteristiche in questo caso non la rendono adatta ad esempio per il gioco Reversi, poiché la caratteristica del movimento potrebbe essere comune solo ai giochi di Dama e Scacchi]

Player*:* classe che definisce il concetto e le caratteristiche di giocatore.

ComputerPlayer*:* classe che definisce un particolare tipo di giocatore, dunque estende un semplice Player, il quale obiettivo è quello di fare sempre una mossa randomica, finchè è in partita.

***CheckersGame**:* implementazione diretta del concetto di partita nel gioco della dama. Classe che gestisce l'azione scatenante per il passaggio da un GameFrame di gioco all'altro, da quello iniziale a quello terminale. Controllando sempre la condizione di termine di partita, effettua come azione principale una mossa. Ogni mossa renderà possibile il passaggio tra un turno e l'altro (ovvero tra un GameFrame e l'altro)

***CheckersGameFrame**:* Classe che contiene tutte le informazioni necessarie per gestire un singolo GameFrame nel gioco della Dama Italiana. Riesce a recuperare, in base alla disposizione dei pezzi, ogni possibile mossa di un pezzo ovviamente tenendo in considerazione le regole della Dama italiana.

***CheckersBoard**:* Classe che ha la funzione di creare le condizioni necessarie e l'ambiente preconfigurato per il gioco della dama italiana. Con tutte le funzionalità che l'inizializzazione del gioco richiede. Dunque il colore di background delle caselle e la disposizione iniziale.

***Pawn**:* pezzo specifico per il gioco della dama. Che porta con se un comportamento specifico di poter cambiare il suo stato da semplice pedina a Dama (o king).

Enumerazioni

BackGroundColor: {**DARK, LIGHT**} colore di background delle celle della damiera, per definizione

Colour: {**BLACK, WHITE**} colore distintivo per i giocatore

GameState: { **RUNNING, DRAW, PLAYER_1_WINS, PLAYER_2_WINS** } possibili stati del gioco durante la partita. Fondamentali per il proseguimento o l'interruzione del gioco.

MoveResult: { **OK, *START_LOCATION_EMPTY, START_LOCATION_OTHER_PLAYER, MOVE_NOT_VALID*** *}*

Per implementazioni future..

È possibile sfruttare la libreria già presente per sviluppare giochi basati su scacchiera. Ad esempio è possibile sviluppare un gioco come scacchi poiché la Board è già in grado di restituire molte informazioni riguardanti le caselle vicine o informazioni sulla casella stessa. In più sarebbe comodo astrarre maggiormente il concetto di Move rendendola un interfaccia (Moveable), con un unico metodo '**move()'** implementabile singolarmente con un comportamento customizzato per ogni pezzo degli scacchi, visto che i comportamenti sono diversi per ogni pezzo.

In più basterebbe cambiare la preconfigurazione iniziale della scacchiera ed impostare correttamente il primo GameFrame per far si che il gioco poi (con le giuste regole per ogni pezzo) possa iniziare ed arrivare a termine GameFrame dopo GameFrame.

Loris E. Minetti