L’UML è organizzato in un package, che contiene tutte le classi facenti parte del model, ovvero tutta la struttura del gioco.

NOTA: i metodi setter e getter non verranno specificati a meno di casi particolari.

**GAME:**

Attributi:

* Chat chat: oggetto che verrà utilizzato per mostrare la chat della partita
* ArrayList<Player> players: conterrà i player della partita (min 2, max 4)
* Board board: oggetto che rappresenta il tabellone di gioco
* Player firstPlayer, currentPlayer: attributi che tengono traccia del primo giocatore (con cui inizia ogni giro), e del giocatore corrente.
* ArrayList<CommonGoal> commonGoals: ArrayList che conterrà le due istanze di commonGoal

Metodi:

* getPlayers(): restituisce l'ArrayList<Player> della partita.
* getCommonGoal(): restituisce i due commonGoal che sono stati estatti a caso per la partita.
* addPlayer(): aggiunge un player alla partita.
* randomCommonGoals(): estrae randomicamente 2 commonGoals e li istanzia (i commonGoals istanziati non possono essere uguali).
* fillCommonGoalsArray(): è un metodo di supporto per il metodo randomCommonGoals, di fatti è stato messo con privacy private. (La sua funzione consiste nel creare un array di classi generics che corrispondono ai commonGoals).

**CHAT:**

Attributi:

* ArrayList<String> chat: contiene tutti i messaggi inviati dai giocatori sotto forma di String

Metodi:

* addChatLog(String message): aggiunge il nuovo messaggio all'array chat

**PLAYER:**

Attributi:

* int position: indica la posizione del giocatore nel giro
* String nickname: indica il nickname del player
* int points: indica i punti accumulati durante la partita dal giocatore
* List<Tile> temporaryTiles: contiene i Tiles che il giocatore sceglierà di turno in turno, ha capienza massima 3 e verrà svuotato subito dopo l'inserimento dei Tiles nella Shelf
* ArrayList<boolean> CommonGoalRedeemed: array di lunghezza massima 2: true se il giocatore ha già soddisfatto i requisiti del commonGoal nella rispettiva posizione. Le posizioni saranno uguali a quelle dei CommonGoal nella classe Game.
* Shelf shelf: indica la shelf del player
* PersonalGoal personalGoal: contiene il personalGoal del giocatore

Metodi:

* isCommonGoalRedeemed(int indice): restituisce true se l'attributo commonGoalRedeemed[indice] è true. Indica se il giocatore ha già soddisfatto i requisiti del commonGoal controllando all'indice 'indice' l'array commonGoalRedeemed
* addTile(Tile tiles): prende dalla board i tiles e li mette in temporaryTiles
* selectTile(int index): seleziona i tile da temporaryTiles e con l'ausilio del metodo insertTiles() nella classe Shelf li inserisce nella shelf del giocatore.
* tryPersonalGoal(int goalCode): genera un personalGoal casuale per il player stando prestando attenzione a non associare un personalGoal a più player.

**SAVEGAMESTATUS:**

Attributi:

* Game game: indica il game da salvare
* String filepath: indica dove salvare il game

Metodi:

* gameToJson: converte il game in un file Json
* saveGame: salva il game

**BOARD :**

Attributi:

* Tile board[9][9]: una matrice che rappresenta la board. Viene inizializzata in modo da rendere inaccessibili le caselle inutilizzate
* boolean endGame: indica se è presente o meno la tessera da un punto

Metodi:

* updateBoard: modifica una tile della board
* getTileFromBoard: restituisce una tile della board

**SHELF:**

Attributi:

* Tile shelf[6][5]: indica la shelf

Metodi:

* insertTiles: inserisce una tile nella shelf

**PERSONALGOAL:**

Attributi:

* Tile personalGoal[6][5]: una matrice simil-shelf che simula il personal goal
* List<Integer> usedCodes: i codici dei personal goal già assegnati

Metodi:

* cleanUsedCodes: resetta i codici usati

**COMMONGOAL:**

**(è implementato tramite strategy)**

Attributi:

* int redeemedNumber: il numero di volte che il commonGoal è stato assegnato

Metodi:

* checkRequirements: controlla se è stato soddisfatto il commonGoal mediante vari algoritmi
* increaseRedeemedNumber: aumenta il redeemed number

Non si consideri l’UML come definitivo, è soggetto a variazioni in caso l’implementazione del codice lo richieda.

Per qualsiasi chiarimento, rimaniamo a vostra disposizione. Mail del referente: luca.parsani@mail.polimi.it