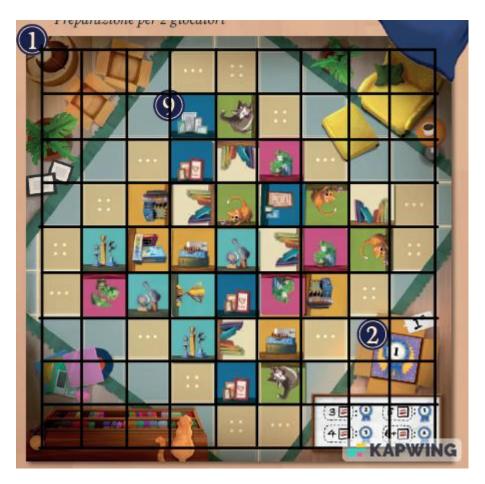
## NOTA INTEGRATIVA DESCRIZIONE MODEL DI MYSHELFIE

## Componenti del gioco

- Plancia Soggiorno (1x)
- Libreria (4x)
- Sedia del 1° Giocatore (1x)
- Sacchetto (1x)
- CARTE
  - Obiettivo Comune (12x)
  - o Obiettivo Personale (12x)
- TESSERE
  - o Punteggio (8x)
  - o Fine Partita (1x)
  - o OGGETTO (132x)
    - Gatti (22x)
    - Libri (22x)
    - Giochi (22x)
    - Cornici (22x)
    - Trofei (22x)
    - Piante (22x)

## Entità tra i componenti del gioco e motivazioni sulla loro scelta

<u>Plancia Soggiorno,</u> denominato GameBoard, è il tabellone di gioco. Implementato come matrice 9x9 (come in figura) di Tessere Oggetto. Accoppiata a questa entità è l'attributo validGrid in Game (vedere nel seguito).



<u>Libreria</u> Allo stesso modo anche le librerie di ciascun giocatore per ogni momento del gioco conterranno informazioni importanti per essere ignorate, dal momento che il loro completamento determinerà la fine della partita. Dispone del metodo isFull(), pubblico, che determina se la Libreria è piena.

<u>Tessere Oggetto</u> La posizione delle tessere oggetto in ogni momento della partita è di fondamentale importanza per lo svolgimento del gioco: di fatto sono la preoccupazione principale di ciascun giocatore e il loro spostamento tra Plancia e Librerie determina sostanziali cambiamenti al gioco, come la fine della partita. Ha un attributo Category che rappresenta il tipo di Tessera Oggetto tra i sei del gioco.

<u>Carte Obiettivo Personale</u> Esse sono di un numero finito, immutabile per tutta la durata della partita e perciò enumerabili. Inoltre, la loro importanza è altamente rilevante siccome determinano scelte di strategia e tattica di ciascuno dei giocatori, modificando quindi lo svolgersi di una partita in modo differente. Ha un attributo di tipo stringa personalObjCardDescription, una cardGrid matrice di Tessere Oggetto che contiene le posizioni dei tipi di Tessere Oggetto necessarie per triggerare l'obiettivo della carta. Sia la descrizione che la matrice verranno riempiti da un Reader JSON (vedere nel seguito)

<u>Carta Obiettivo Comune</u> Rappresentati da commonObjCard, ha come attributi objPoints, vettore di interi che contiene le astrazioni delle Tessere Punteggio in base al numero di giocatori, nextPoints di tipo intero che è un riferimento alla posizione del vettore objPoints relativa ai prossimi punti da assegnare, un riferimento a StrategyCheck che implementa il pattern Strategy mediante l'utilizzo di una classe astratta, playersDone lista di Player che registra i giocatori che hanno già conseguito l'obiettivo e quindi non possono prendere ulteriori punti dalla carta. Come metodi doCheck(),

pubblico, che esegue, delegando a StrategyCheck il controllo del raggiungimento dell'obiettivo della carta. In caso fosse raggiunto provvede a erogare i punti e registrare il giocatore. Il metodo defineStrategy, privato, è utilizzato al momento della costruzione della Carta per definire quale tipo di controllo si deve effettuare.

<u>Sacchetto</u> Questo componente svolge il ruolo di mazzo. È un contenitore delle Tessere Oggetto. Detiene una lista di Tessere Oggetto, un metodo drawitem() che estrare casualmente una Tessera Oggetto dal Sacchetto e un metodo setItemCards() che aggiunge una Tessera Oggetto al Sacchetto.

<u>Carta Fine Partita</u> Questa carta non ha risvolti di gioco: è una sorta di bonus che aggiunge un punteggio pari ad uno al primo giocatore che riempie la propria libreria. Viene assegnato dal controller al raggiungimento delle condizioni.

<u>Sedia del 1° giocatore</u> Questo componente è un segnaposto. Determina il primo giocatore della partita da cui parte poi il giro dei turni. Viene assegnato dal controller e inglobato in Player mediante l'attributo isFirstPlayer.

<u>Carte Punteggio</u> Questo componente assegna un punteggio in relazione al completamento delle carte Obiettivo Comune. Viene inglobata nelle Carte Obiettivo Comune, le quali assegneranno un punteggio diverso a seconda dell'ordine con cui i giocatori completano l'obiettivo.

<u>Giocatore</u> Rappresentato da Player ha come attributi nickname (stringa) e clientID (intero) che identificano il giocatore, score (intero) che è il suo punteggio, un riferimento alla sua libreria con myShelf, alla Carta Obiettivo Personale con myPersonalObjCard, selectItems che è la lista di Tessere Oggetto che il giocatore ha raccolto dalla Plancia e si accinge a depositare nella propria Libreria e l'attributo booleano isFirstPlayer. Come metodi:

- playerChoice(), pubblico, che rappresenta la scelta del giocatore durante il proprio turno
- pickItems(), privato, che raccoglie quelle Tesser Oggetto selezionate dal giocatore e le ripone nell'attributo selectItems
- putItemInShelf(), privato, che deposita le Tessere Oggetto in selectItems nella Libreria
- addPoints(), pubblico, che aggiunge al giocatore i punti che ha conseguito con la propria strategia
- setPersonalObjCard(), pubblico, che istanzia la Carta Obiettivo Personale del giocatore
- setMyShelf(), pubblico, che instanzia la Libreria del giocatore

<u>Game</u> rappresenta l'entità del gioco MyShelfie di cui un'istanza verrà creata dal Controller. Detiene una lista di giocatori, il numero di giocatori, una Plancia Soggiorno, un Sacchetto, una lista con le due Carte Obiettivo Comune e un riferimento al giocatore in turno. Dispone di una "validGrid", ossia una matrice di interi che, accoppiata alla Plancia Soggiorno conserva le informazioni sulle celle sempre non raggiungibili della Plancia e su quelle giocabili a seconda del numero di giocatori in partita.

(0 --> INVALID: sempre non giocabile, 1 --> PLAYABLE: giocabile con questo numero di giocatori, 2 --> OCCUPIED: cella giocabile e occupata da una Tessera Oggetto).

## METODI DI GAME:

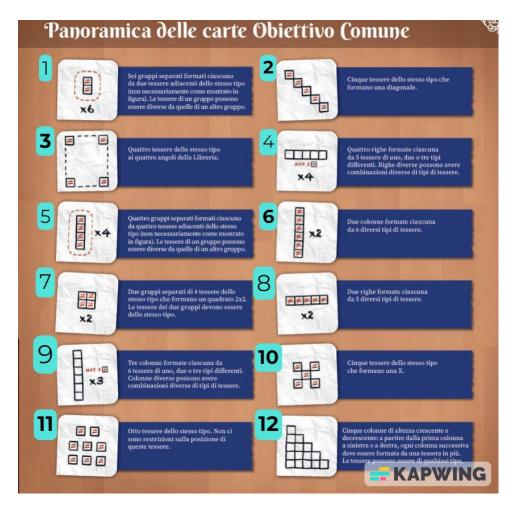
- o createPlayers(): metodo privato che crea i giocatori per questa partita
- o createGaBoard(): metodo privato che crea la Plancia Soggiorno per questa partita
- o createBag(): crea il Sacchetto per questa partita

- setGridForTwo(): metodo privato che crea una Plancia Soggiorno per una partita di due giocatori. Viene usata come base per creare le Plance delle partite con più di due giocatori.
- o generatePersonalObjCards(): metodo pubblico che crea le Carte Obiettivo Personale e le distribuisce ai giocatori in partita
- generateCommonObjCards(): metodo pubblico che crea le due Carte Obiettivo Comune per questa partita
- refillGameBoard(): metodo pubblico che riempie la GameBoard qualora si verificassero le condizioni espresse dalle regole del gioco che impongono un riempimento della Plancia Soggiorno
- o setCurrentPlayer(): metodo pubblico che impone il giocatore di turno

<u>PersonalOBJReader</u> E' l'entità preposta alla lettura dal file Json delle Carte Obiettivo Personale. Ha come attributo file (il nome del file Json) e come metodo readFromFlle(), pubblico, che esegue la lettura del suddetto file e restituisce la lista di PersonalObjCard.

<u>StrategyCheck</u> E' l'applicazione del design pattern Strategy. Ha l'attributo type, protected, di tipo intero che identifica uno dei dodici tipi di Carta Obiettivo Comune in modo da eseguire il corretto metodo di controllo "obiettivo raggiunto". Questo controllo viene preso in carico dal metodo check(), overridato nelle sue sottoclassi. Le sottoclassi (RowsColumnsCard, ThreeDifferentTypes, CornerDiagonals, GroupCards, XCards) hanno l'implementazione del metodo check() il quale a seconda del valore di type effettua una chiamata all'apposito metodo di controllo.

- RowsColumnsCard: controllo sulla sesta e ottava carta
- ThreeDifferentTypes: controllo sulla quarta e la nona carta
- CornersDiagonals: controllo sulla seconda, terza e dodicesima carta
- GroupCards: controllo sulla prima, quinta, settima e undicesima
- XCards: controllo sulla decima carta



<u>Tipi di Tessere Oggetto</u> Sono i sei tipi di Tessere Oggetto. Sono rappresentati da Category, un'enumerazione