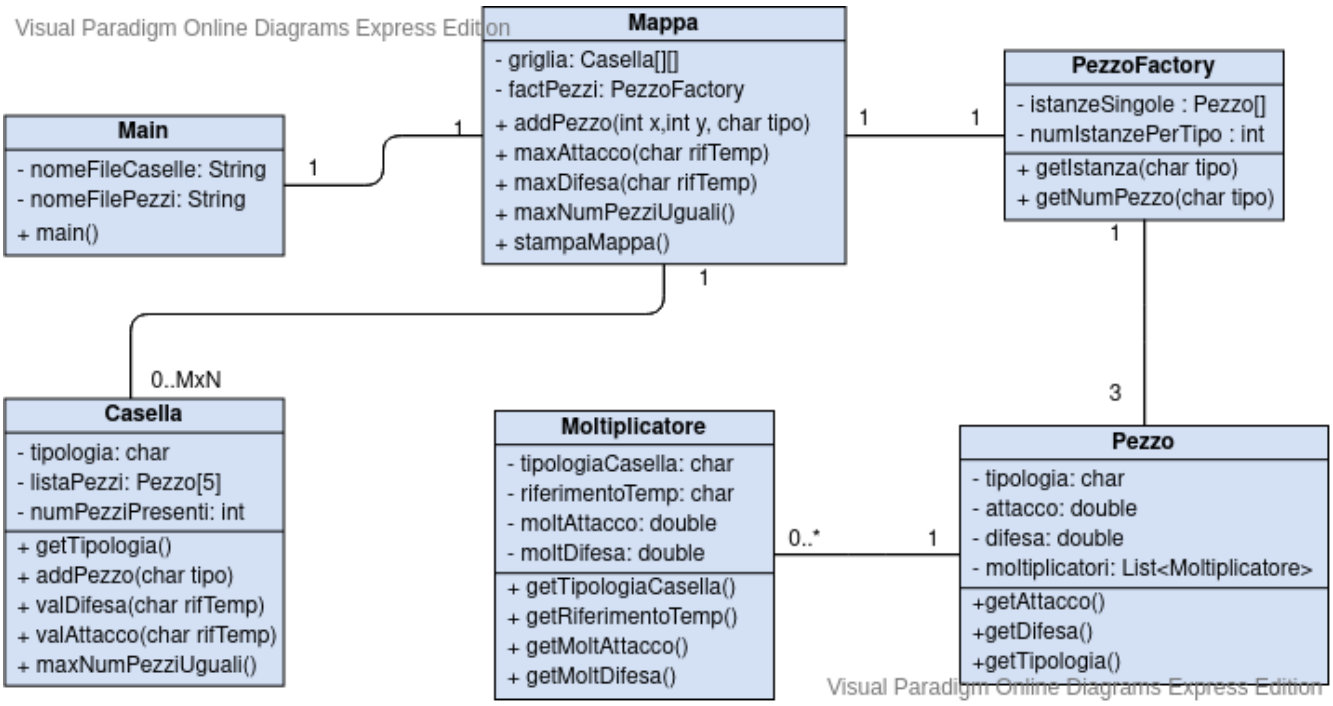
Documento di architettura, versione 1.0

Sviluppatore: **Cappellotto Lorenzo, matricola 1188257**, corso ingegneria informatica generale.

Questo documento e' un riassunto di come il sistema sara' strutturato e di quali sono le sue funzionalita' principali.



Moltiplicatore

E' un classe usata solamente per descrivere i moltiplicatori di attacco e difesa da attribuire alla tuple {tipologia-Casella, tipologia-Pezzo, riferimento-Temporale}. Per ridurli di numero, in caso ve ne siano di simili e' possibile inserire come tipologia-Casella o riferimento-Temporale il simbolo '*' per indicare "qualsiasi caso".

Questa classe e' strutturata in modo da permettere la specifica degli attributi solo al momento della creazione degli oggetti. Per ottenere questo risultato gli attributi dovranno essere privati e la classe sara' dotata solo di metodi get, ma non di metodi set.

Pezzo

Questa classe permette di descrivere un pezzo come richiesto dai committenti, ovvero oggetti dotati degli attributi {tipologia, attacco, difesa, moltiplicatori} dove i moltiplicatori sono una lista di oggetti e non una tabella data la loro quantita'.

Non verra' creata una classe specifica per ogni tipologia di pezzo, ma essa verra' usata come attributo per identificarli tra di loro.

Anche questa classe, come Moltiplicatore, specifichera' i dati solo al momento della creazione, non permettendo modifiche successive. Sara' quindi dotata di attributi privati e metodi get.

In particolare i metodi getAttacco() e getDifesa() restituiranno il valore di attacco o difesa dato l'ambiente in cui il pezzo si trova (tipologia-Casella + riferimento-Temporale).

PezzoFactory

Data la natura dei pezzi, in cui ogni oggetto della stessa tipologia ha tutti i rimanenti valori uguali, ho deciso di costruire una classe che utilizzi una combinazione dei pattern Factory e Singleton per permettere la specifica di un singolo Pezzo per ogni tipologia e la restituzione di essi in base alla singola tipologia di pezzo richiesta. (Pezzo e' una classe concreta e non un'interfaccia, ma ho associato la restituire di un'istanza in base a un parametro con il pattern Factory ugualmente).

Inoltre, all'interno di questa classe verranno memorizzate una lista di tipologie possibili e il numero di pezzi per ogni tipologia. Mentre la lista dovra' essere costante, il numero di pezzi verra' incrementato ogni volta che viene richiesto un getIstanza alla Factory.

Casella

Questa classe rappresenta ogni cella della mappa. Ogni oggetto dovra essere dotato degli attributi {tipologia-Casella, un vettore di pezzi}.

Metre la tipologia potra' essere specificata solo al momento della creazione dell'oggetto, i pezzi dovranno essere aggiunti successivamente attraverso il metodo addPezzo(Pezzo p), rispettando il limite massimo di 5 pezzi.

Inoltre, questa classe dovra' restituire i valori di attacco o difesa della casella, ovvero la somma dei valori di attacco o difesa di tutti i pezzi presenti all'interno della casella dato il riferimento-Temporale giorno/notte.

Infine, dovra' restituire la tipologia attraverso un metodo get e il massimo numero di pezzi dello stesso tipo presenti all'interno della casella.

Mappa

Questa classe rappresenta la mappa nel suo insieme. Dovra' essere dotata delle dimensioni M e N, di una griglia di caselle e di un istanza di PezzoFactory per permettere la creazione di pezzi.

Tutti gli attributi verranno specificati al momento della creazione della mappa e non potranno essere piu' modificati. Solamente i pezzi dovranno essere aggiunti successivamente attraverso il metodo addPezzo().

Inoltre, la classe Mappa dovra soddisfare tutte le richieste funzionali dei committenti e quindi:

- Ritornare le coordinate della casella avente massimo valore di attacco o difesa attraverso i metodi maxAttacco() o maxDifesa()
- Ritornare le coordinate della casella avente il massimo numero di pezzi della stessa tipologia attraverso il metodo maxNumPezziUguali()
- Ritornare il numero di pezzi presenti nella mappa per ogni tipologia attraverso getNumPezzi()
- Ritornare una matrice di caratteri rappresentante la tipologia delle caselle

Main

Questa parte del sistema non sara’ una classe come quelle sopra descritte ma piuttosto un punto di avvio del sistema.

Il metodo statico Main() si dovra’ occupare della gestione dei file e quindi la loro apertura e lettura. Inoltre, si occupera’ dell’ottenimento e della validazione dell’input.

Infine, attraverso un oggetto della classe Mappa prima descritta, si occupera’ di stampare a video tutte le funzionalita’ prima descritte nella classe Mappa.