Peer-Review 1: UML

Arturo, Luca, Lorenzo
Gruppo 42

4 aprile 2022

Valutazione del diagramma UML delle classi del gruppo 52.

1 Lati positivi

- L'idea di utilizzare una **classe astratta per Tile** è buona, tuttavia potrebbe aver senso mettere la `List<Pawn>` presente in entrambe le classi figlie (`Cloud.cloudStudent` e `Island.IslandStudent`) all'interno del padre.
- `Bag` come **Singleton** (se non sono previste partite multiple).

2 Lati negativi e consigli

- **L'UML pare incompleto** in più punti (`getColor()` in `Pawn` non ritorna nulla, non è chiaro cosa faccia `activateEffect()` di `Character_Card`, etc..).
- **Inconsistenze nella naming convention**. Ci sembra di capire che usiate principalmente camelCase, tuttavia ci sono alcuni nomi che non sono consistenti con questa decisione. In particolare, usate la `Pascal_Snake_Case` convention per `School_Board`, `Color_Tower`, `Color_Pawn`, `Mother_nature`, `Mother_nature.get_instance()` ed `Assistant_Card`.
- **Movimento di Madre Natura**: Come fa Madre Natura a sapere (in `move`) quando ricominciare dalla posizione 0?
  Es. La posizione iniziale è 8 e ci sono 10 isole. Facendo `move(3)`, come faccio a sapere che finirò in posizione 1?
- **Funzionamento di TileFactory** al variare del numero di giocatori. In particolare, come fa a sapere quante Cloud creare (visto che dipende dal numero di giocatori)?
- **MergeIslands()** non avrebbe più senso **come metodo di Game?** In questo modo non sarebbe necessario passare islands alla funzione. Inoltre, assumiamo che dall'UML sia stata omessa la specifica di size ed il suo corrispettivo getter in Island.
- Non è molto chiara l'utilità della **classe Coin**.
  Probabilmente un intero sarebbe più che sufficiente.

- Non è chiara l'utilità della **classe Mother_nature**.
  Probabilmente un booleano all'interno di ogni isola sarebbe più che sufficiente.
- Non vi è alcun riferimento a `Professor` nelle altre classi, quindi potrebbe risultare superfluo.
- Visto che alcuni componenti all'interno di `Game` (`Bag` e `Mother_nature`) sono singleton, potreste considerare di rendere anche `Game` un singleton.

2.1 Possibili refusi

- Non è specificato alcun valore di ritorno per `getColor()`.
- Perchè `School_Board.initTower()` vuole un `Color_Pawn` e non un `Color_Tower`?
- `Value` e `Steps` in `Assistant_Card` sono attributi, eppure hanno la maiuscola.
- Manca l'attributo `size` in `Island`: come faccio a sapere da quante isole è composto un arcipelago?
- `removeEntranceStudent()` in `Tower`.
- Manca una 'b' in `numberOfStudentColor()` in `Player`.

3 Confronto tra le architetture

L'architettura del gruppo GC52 fa un ottimo uso del pattern Singleton. Sfortunatamente, questa idea è di difficile applicazione nel nostro caso, in quanto abbiamo deciso di optare per l'implementazione della FA: "Partite multiple".