Peer-Review 1: UML

Roberto Ferri, Lorenzo Fiamingo, Luca Musco, Lorenzo Rossi.

Gruppo 4

Valutazione del diagramma UML delle classi del gruppo 3.

Lati positivi

Lati positivi, punto per punto:

- `AssistantCard` come enumerazione.

Lati negativi

Lati negativi, punto per punto:

- `StudentHolder` è una generalizzazione non necessaria perché:
  - `Bag` eredita metodi che non gli servono e può essere semplicemente una struttura di dati all'interno di `GameManager`
  - `Cloud` eredita metodi che non gli servono e può essere semplicemente una struttura di dati all'interno di `GameManager`
- `Professor` non è necessaria perché non serve sapere in ogni momento quale giocatore controlla quale professore
  - ai fini del calcolo dell'influenza sull'isola il controllo del professore può essere valutato turno per turno
- `CharacterCard` è un'interfaccia che non viene implementata da nessuna classe.
- `StudentHolderObserver` è un'interfaccia ma viene estesa da delle classi. Non si capisce quindi se quelle classi siano interfacce (in questo caso queste interfacce non vengono mai implementate), o se la dicitura *extends* sia stata scambiata erroneamente con *implements*.
- In `Player` non è necessaria la variabile `towersLeft`, in quanto basta invece contare quante torri si trovino sulle isole.
- In `AssistantCard` non è possibile distinguere carte che hanno lo stesso valore e lo stesso numero di passi possibili che può effettuare madre natura. Inoltre dove e quando vengono assegnate le carte al `Player`? Come viene garantito che la stessa carta appaia più di una volta?

Confronto tra le architetture

- Uso degli *osservatori*
  - non ancora previsti nel nostro schema
- Metodo di implementazione della *fusione delle isole*
  - più comodo e più facile da gestire rispetto al nostro approccio basato sul rimuovere le isole che si fondono