# Peer-Review 1: UML

Kevin Ziroldi, Matteo Volpari, Samuele Villa, Tommaso Uberti Foppa

Gruppo GC07

Valutazione del diagramma UML delle classi del gruppo GC54.

## Lati positivi

- Introdurre due classi concrete DiagonalDisposition e LDisposition permette di usare algoritmi più specifici per verificare i due tipi di condizioni.
  Questa cosa risulta efficiente soprattutto dato che il layout è stato memorizzato in una matrice molto grande.
- Gestione delle Condition con design pattern strategy.
- Nell'UML compaiono già classi atte a realizzare funzionalità aggiuntive.
- Si nota una suddivisione rigorosa del Model e del Controller, in accordo con il design pattern MVC.

## Lati negativi

### Alcuni problemi generali

Per nessuno dei metodi sono indicati i parametri, il che rende più complesso capire il loro funzionamento.
Nell'UML compaiono delle associazioni tra oggetti che in realtà non sono in relazione tra loro; per esempio, CommonBoard, PlayerBoard e Player hanno una associazione con Chat.

### Problemi legati a singole classi:

- Sono presenti attributi ridondanti, ovvero salvati più volte in classi diverse.
  Il riferimento alla CommonBoard è presente sia nella classe Model, che in tutti i Player; era sufficiente salvarlo una volta nel Model.
  PlayerBoard memorizza il nickname, anche se il Player ha il riferimento alla PlayerBoard.
  Sia nella classe Model che nella classe GameController sono presenti gli attributi players e gameID, che rappresentano la stessa cosa e ottenibili tramite getter del Model.
- Esiste una unica classe enumerata ResourceType che contiene sia le risorse che gli oggetti di gioco, ma sarebbe stato più corretto dividerle.
  Ad esempio, la classe Card ha un attributo permanentResource che dovrebbe contenere solamente risorse, e non oggetti di gioco.
- In GoldCard ci sono due attributi, coveredEdges e resourceRequiredToScore, ma nessuna carta oro li ha entrambi, solo uno o nessuno. Sarebbe stato meglio spostarli nella Condition, così da evitare attributi inutili per ogni GoldCard.
- Nelle ObjectiveCard, i layout sono massimo di 3x3 carte, non è necessario utilizzare una intera PlayerBoard, che avrà moltissime posizioni vuote.
- Le carte obiettivo hanno due possibili tipi di condizioni: layout o risorse.
  Nelle ObjectiveCard viene salvato solamente il layoutToMatch, mentre nel caso la condizione riguardi le risorse sul campo, questa non viene memorizzata.

## Confronto tra le architetture

- Abbiamo apprezzato la soluzione di introdurre una classe ModelView immutabile, che possa essere inviata alle view.
- Noi abbiamo memorizzato nel campo da gioco direttamente le carte, al posto degli angoli. Questo ci ha permesso di memorizzate ogni carta una sola volta, evitando di memorizzarne varie copie.