Breve descrizione dei messaggi scambiati tra Client e Server:

Da Client a Server:

- **Message**: è una classe astratta rappresentante una mossa inviata da un giocatore (Client) al Server. Message estende diverse sottoclassi, ciascuna rappresentante una diversa tipologia di mossa.
  - *String sender*: nickname del giocatore che ha inviato il messaggio
  - *Action action*: tipo di mossa richiesta (Action è un'enum)
- **AddMeMessage**: questo è il primo messaggio che ogni client invia al server, specificando con quanti giocatori vuole giocare e se vuole utilizzare le regole complete o meno, in modo che GameManager possa poi inserirlo nella partita a lui più adatta.
  - *boolean completeRules*: utilizzare regole complete o no?
  - *int maxPlayers*: numero di giocatori con i quali si vuole giocare.
- **PlayCardMessage**: questo messaggio viene inviato quando un giocatore decide di giocare una delle proprie carte assistente.
  - *int priority*: intero che identifica quale carta giocare (visto che le carte sono univocamente numerate da 1 a 10).
- **MovementMessage**: questa classe astratta è necessaria ad estendere poi MoveStudentMessage e ExchangeStudentMessage. Questa classe astratta include gli attributi comuni di queste due classi.
  - *Location departureType*: specifica su che tipo di location (enum) si trova lo studente da spostare. Lo studente potrebbe infatti trovarsi all'ingresso della propria scuola (location==ENTRANCE), su un'isola (location==ISLAND), nella sala mensa (location==CANTEEN) o su una carta personaggio (CARD_EXCHANGE, CARD_ISLAND, CARD_CANTEEN).
  - *int departureId*: dato il tipo di location, identifica in quale location si trova lo studente. Ad esempio, se departureType==ISLAND, departureId potrebbe essere un intero da 0 a 11.
  - *Location arrivalType*: analogo a departureType, solo che specifica il tipo di location del posto nel quale vogliamo spostare lo studente.
  - *int arrivalId*: analogo a departureId.
- **MoveStudentMessage**: questo messaggio viene inviato dal Client quando l'utente vuole spostare due studenti.
  - *int studentId*: identifica quale studente l'utente vuole muovere.
- **MoveMotherNatureMessage**: messaggio inviato dal Client per mouvere Madre Natura.
  - *int movement*: numero di passi che Madre Nature deve eseguire.
- **Classe SelectCloudMessage**: ultimo messaggio inviato dal client prima della fine del suo turno per scegliere una nuvola.
  - *int cloudPosition*: identifica quale nuvola si vuole scegliere.
- **UsePowerMessage**: messaggio inviato dal client quando l'utente vuole attivare una carta personaggio.
  - *int characterCloud*: identifica quale carta si vuole attivare.
- **ChooseColorMessage**: classe astratta che contiene gli attributi comuni alle carte che prevedono la scelta di un colore.
  - *Color chosenColor*: colore selezionato dall'utente.
- **BlockColorMessage**: se la carta attivata prevede di scegliere un colore che verrà ignorato al prossimo movimento di Madre Natura, il client dovrà inviare questo tipo di messaggio DOPO aver inviato un messaggio UsePowerMessage.
- **PutBackMessage**: se la carta attivata prevede di scegliere il colore degli studenti che devono essere ritornati dalla sala mensa al sacchetto, il client dovrà inviare questo tipo di messaggio DOPO aver inviato un messaggio UsePowerMessage.
- **ChooseIslandMessage**: se la carta attivata prevede di scegliere un'isola, il client dovrà inviare questo tipo di messaggio DOPO aver inviato un messaggio UsePowerMessage.

- *int idIsland*: identifica l'isola scelta dall'utente.

- **BlockIslandMessage**: se la carta attivata prevede di scegliere un'isola da bloccare, il client dovrà inviare questo tipo di messaggio DOPO aver inviato un messaggio UsePowerMessage.
- **InfluenceIslandMessage**: se la carta attivata cambia le regole del calcolo dell'influenza sull'isola (ad esempio, ignorare le torri), il client dovrà inviare questo tipo di messaggio DOPO aver inviato un messaggio UsePowerMessage.
- **ExchangeStudentMessage**: se la carta attivata prevede di scambiare due studenti, il client dovrà inviare questo tipo di messaggio DOPO aver inviato un messaggio UsePowerMessage.
  - *int studentId1*: matricola studente da scambiare.
  - *int studentId2*: matricula student da scambiare.
- **PingMessage**: è un messaggio "vuoto" che il Client invia continuamente al Server (ogni 4 secondi). Infatti, per evitare che un Client possa abbandonare improvvisamente la partita lasciando il Server in attesa di una sua risposta che mai arriverà, bloccando l'intera partita e gli altri giocatori in quello stato, il Server si aspetta un messaggio proveniente da ciascun Client ogni 20 secondi. Se entro questo lasso di tempo non riceve alcun messaggio, significa che il Client è stato disconnesso, e perciò il Server termina la partita per tutti. Il messaggio di Ping è quindi necessario affinché il Client non interrompa la partita solo perché un giocatore impiega più di 20 secondi a pensare alla propria mossa.

Da Server a Client:

- **GameReport**: questo messaggio può avere x3 funzioni differenti
  - questo messaggio inviato dal Server ai giocatori contiene tutte le informazioni relative allo stato corrente del gioco: studenti sulle isole, studenti agli ingressi, studenti nelle sale mensa, carte disponibili… Ogni volta che il modello viene aggiornato, una copia di GameStatus viene inviata a tutti i client. Inoltre, a ciascun giocatore viene inviato un GameReport diverso, contenente SOLO le informazioni a lui destinate. Questo per evitare che, nel caso GameReport fosse lo stesso per tutti, un client malevolo possa accedere a dei campi a lui normalmente preclusi per leggere informazioni private destinate solo ad altri (come, ad esempio, le Carte Assistente a disposizione degli avversari).
  - Nel caso il Client chiedesse di effettuare una mossa illegale (ad esempio chiedesse di scegliere una nuvola anche se è il momento di giocare una carta), il Server restituirebbe un oggetto GameStatus (A LUI SOLTANTO!!!) con tutti i parametri nulli o vuoti. L'unica eccezione sarebbe il campo "error", che conterrebbe la stringa di testo che specifica quale errore è stato commesso. Il client infatti, ricevuto l'oggetto GameStatus, controlla per prima cosa il campo "error". Se questo contiene qualcosa, significa che un errore è stato commesso e che il modello non è stato modificato: il client, quindi, stamperà a video il solo messaggio d'errore. Se questo campo invece è vuoto, significa che la mossa era legale ed è stata correttamente applicata al modello, quindi il Client stamperà a video il nuovo modello.
  - Questa entità svolge inoltre il ruolo di "Pong": non appena al Server giunge un PingMessage dall'utente, quest'ultimo risponde con un GameReport contenente nel campo error il messaggio "PONG". In questo modo, quando il Client riceve queso GameRecord particolare, sa che è ancora connesso correttamente con il Server e non applica alcuna modifica alla propria grafica.