Note UML Sequence Diagram

Ilaria Brunelli, Elena Brugnolo, Lisa Cardini, Giulia Pennella

Gruppo AM22

L'UML sequence diagram inviato è diviso in due parti per facilitarne la comprensione: la parte sinistra implementa la logica prima dell'inizio vero e proprio della partita, nella parte destra, invece, sono presenti le azioni eseguite durante la partita vera e propria.

Tra client e server vengono scambiati file json: quando devono essere scambiati dati viene costruita un'istanza della classe messaggio opportuna (ci sono varie classi messaggio che modellano i pacchetti di informazioni scambiati client-server), questa viene convertita in una stringa json e inviata. A destinazione la stringa json viene riconvertita in un oggetto Java che verrà gestito opportunamente (si identifica il tipo di messaggio tramite il contenuto dell'attributo type).

>> UML Sequence Diagram a sinistra

Inizialmente il client si connette al server il quale crea un thread dedicato con una nuova istanza della classe ClientHandler (che si occupa di gestire la singola connessione con il client lato server).

Il client chiede al server di giocare e ciò può avvenire in due modalità differenti (nel diagramma è indicato con un alt) a seconda che sia il first player (se vuole creare una nuova partita) o meno.

Se il giocatore è il first player, invia al ClientHandler un messaggio con il numero di giocatori della nuova partita e il suo nickname. ClientHandler a sua volta invoca un metodo di GamesHandler* che istanzia il Controller relativo alla partita passando al suo costruttore il codice della partita, il numero di giocatori necessari e il nickname del primo giocatore. Come messaggio di risposta, al client torna il codice identificativo della nuova partita, che potrà essere utilizzato dagli altri giocatori per accedervi. *La classe GamesHandler ha come attributi delle HashMap che tengono traccia dei vari controller instanziati (nel caso di più di una partita), dei ClientHandler associati a ogni singola partita e dei nickname dei giocatori disconnessi.

Se invece non si tratta del primo giocatore, nel messaggio inviato da client a server sarà presente, oltre al nickname del giocatore, anche il codice della partita a cui si vuole accedere. In questa parte di UML è presente un loop: il nuovo giocatore non potrà essere inserito nella lista giocatori a meno che il nickname scelto non sia diverso da tutti gli altri (nel caso ci sia corrispondenza con un nickname già esistente, viene lanciata un'eccezione e come messaggio di risposta uno che indica errore). Questo controllo viene effettuato dal controller all'interno del metodo "addPlayerNickname" chiamando "checkNickname". Il metodo "addPlayerNickname", inoltre, ogni volta che aggiunge un nuovo giocatore alla lista giocatori,

controlla se il numero di players corrisponde a quello dato dal first player: in caso affermativo, viene istanziato un nuovo Game passando al costruttore di quest'ultimo il codice della partita e il numero dei giocatori.

A questo punto, nell'UML è presente un loop: questo ciclo tiene conto del fatto che un giocatore potrebbe voler iniziare il setup del gioco prima che il numero dei giocatori sia effettivamente uguale a quello dichiarato dal primo giocatore. Questo viene gestito da un controllo fatto all'inizio del metodo "PreGame" di Game, il quale viene a sua volta chiamato dal metodo "setUpGame" del controller.

Quando viene raggiunto il corretto numero di giocatori, da uno di questi viene mandato il messaggio per iniziare il setup del gioco, tramite una catena di invocazioni di metodi (da ClientHandler a Controller e da controller a Game) viene chiamato "PreGame". Questo comporterà l'invio a tutti i client della startingCard, dedi commonGoals, delle due goalCards, delle tre carte iniziali e dei colori tra cui poter scegliere. Per fare questo vengono chiamati, sempre nel metodo "setUpGame" del controller, i metodi di game "SetStartingCard" (che ritorna una carta iniziale), "SetCommonGoals" (che ritorna un array contenente gli obbiettivi comuni), "GiveSecretGoals" (che ritorna un array di due carte obbiettivo) e "GiveCard" (che ritorna un Arraylist di tre carte).

A questo punto dal client partono una serie di messaggi per:

- Giocare la carta iniziale: viene chiamato il metodo "playStartingCard" di Controller (a cui vengono passati il nickname, la carta giocata e il lato scelto) che a sua volta chiama il metodo "PlaceFirstCard" di Game (a cui vengono passati la carta, l'id del giocatore e se la carta è stata posizionata su front o back);
- Scegliere l'obiettivo segreto: viene chiamato il metodo "chooseSecretGoals" del Controller che a sua volta chiama il metodo "setSecretGoals" di Game;
- Scegliere il colore: viene chiamato il metodo "choosePlayerColor" di Controller, che a sua volta chiama il metodo "chooseColor" di Game.

A questo punto, il first player può decidere di iniziare il gioco mandando un messaggio: verrà chiamato il metodo "startGame" di Controller che a sua volta chiamerà "StartGame" di Game.

>> UML Sequence Diagram a destra

Qui sono riportati i messaggi e i conseguenti metodi chiamati per le diverse azioni dei giocatori:

- Se il giocatore vuole pescare una carta, una volta ricevuto il messaggio, viene chiamato il metodo "takeCard" del Controller che a sua volta chiama il metodo "ChooseCard" di Game, a cui vengono passati l'id corrispondente al player e la carta;
- Se il giocatore vuole giocare una carta, ricevuto il messaggio viene chiamato il metodo "playCard" di Controller (passando il nickname, la carta, la posizione e il lato) che a sua volta chiama "PlayTheCard" di Game, passando gli stessi parametri (invece di nickname viene passato l'id del player);

- Se il giocatore vuole vedere lo score board, una volta ricevuto il messaggio dal client, viene chiamato il metodo “getScoreBoard” di Controller, che a sua volta chiama “ShowScoreBoard” di Game;
- Se il giocatore vuole vedere il gioco degli altri giocatori, viene chiamato il metodo “showOtherPlayerGame” di Controller che a sua volta chiama il metodo “ShowPlayerGame” di Game; come messaggio verranno inviate le carte piazzate sul piano di gioco del giocatore richiesto;
- Se il giocatore vuole vedere gli obbiettivi comuni, viene chiamato il metodo “getCommonGoals” di Controller, che chiamerà a sua volta il metodo “ShowCommonGoals” di Game;
- A fine partita, se il giocatore vuole sapere lo score finale, viene chiamato il metodo “finalScore” di Controller che a sua volta chiamerà il metodo “FinalScore” di Game;
- Se un utente vuole disconnettersi, il client invierà un messaggio che causerà l’invio di un messaggio di notifica a tutti gli altri client e successivamente la morte del thread corrispondente.