PROTOCOLLO DI RETE – GRUPPO AM07

Quando il server viene lanciato questi si mette in ascolto dei client che cercano di connettersi attraverso l'apertura di un thread SocketServer che si occupa appunto di accettare la richiesta di connessione da parte del client e di istanziare per ciascun client un ClientHandler che si frapporrà nella comunicazione tra ConnectionSocket (lato client) e server.

Il ClientHandler si occupa di gestire la connessione del client al server, inviando messaggi (di servizio o update) in risposta al client e utilizzando i messaggi ricevuti dal client in modo da avanzare nelle diverse fasi di gioco, modificando lo stato del gioco.

Il ClientHandler in ordine chiederà al client il nickname scelto e le sue preferenze di gioco, notificando il server delle scelte, che si occuperà di registrare il nuovo giocatore associando al nickname scelto il relativo ClientHandler, e in base alle preferenze da lui espresse deciderà se aggiungerlo ad un gioco già esistente o se crearne uno nuovo.

Il server è in grado di gestire partite multiple, andando a salvare i diversi match attivi in un ArrayList contenente i GameController di ciascuno di essi; per ogni giocatore vengono salvati all'interno di una HashMap (con il nickname del player come chiave) l'ID che identifica il match, l'ID del player all'interno della partita e il ClientHandler ad esso associato.

Il GameController è il responsabile dell'invocazione dei metodi pubblici di Game che modificano lo stato del gioco, previo controllo della bontà della mossa eseguita.

A seguito della modifica del Model viene chiamata in causa la VirtualView, la quale si occuperà di mandare le informazioni notificatele dal Model a tutti player della partita.

La disconnesione è così gestita: quando si verifica il logout o un errore da parte del client, il ClientHandler notificherà il server della disconnessione, il quale manderà un messaggio di QUIT a tutti i giocatori collegati alla stessa partita del player che ha abbandonato (ove ce ne fossero), dopodichè si occuperà di rimuovere tutte le informazioni associate ai giocatori e alla partita appena terminata.

Il client per connettersi al server istanzia un ConnectionSocket che, tramite il metodo startConnection, dopo aver mandato la richiesta all'indirizzo e la porta indicati dall'utente, lancia due thread: PingSender e Listener.

ClientPingSender si occupa di mandare al server ogni secondo un messaggio di PING in modo tale che la connessione resti attiva anche a seguito di inattività da parte dell'utente; inoltre, una mancata ricezione da parte del server di un messaggio di PING dopo un intervallo di 10 secondi dal precedente comporta la chiusura della connessione anche a seguito della caduta improvvisa della rete.

ClientListener ascolta i messaggi provenienti dalla rete e notifica il ClientController riguardo al messaggio appena ricevuto, eventualmente gestendo disconessioni.

Il ClientController in base alla tipologia del messaggio notificatogli dal ClientListener deciderà quale metodo invocare; esso potrà essere un metodo di aggiornamento delle informazioni provenienti dal Model oppure un metodo della View che chiederà all'utente di eseguire la sua mossa (in quest'ultimo caso verrà lanciato un thread facente parte di una ThreadPool che eseguirà il task assegnatogli).

Infine, il ClientController si occupa anche di mandare al server le informazioni sulla mossa eseguita dall'utente tramite un PlayerMoveMessage, sfruttando il metodo send del ConnectionSocket.

N.B.: il campo "genericValue" del PlayerMoveMessage è sfruttato per differenti scopi, tutti inerenti a un elemento che l'utente ha selezionato (può assumere infatti il valore dell'indice di un colore, di un'isola, di una carta…).

PROTOCOLLO DI RETE – GRUPPO AM07

MESSAGGI:

I messaggi utilizzati estendono la classe astratta NetworkMessage che offre un metodo per ricavare il MessageType associato ad ogni messaggio.

Ecco tutti i messaggi utilizzati:

LoginMessage:

```
{
        "MessageType": "LOGIN",
        "nickname": "String"
}
```

GamePreferencesMessage:

```
{
        "MessageType": "GAME_SETUP_INFO",
        "numberOfPlayers": "int",
       "expertMode": "Boolean"
},
```

GameCreation_UpdateMsg:

```
{
        "MessageType": "START_GAME",
        "numPlayers": "int",
        "nicknames": "ArrayList<String>",
        "gameMode": "GameMode",
        "clouds": "ArrayList<HashMap<RealmColors,Integer>>",
        "studentsOnCharacter": "ArrayList<HashMap<RealmColors,Integer>>",
        "entrances": "ArrayList<HashMap<RealmColors,Integer>>",
         "isleStudents": "ArrayList<HashMap<RealmColors,Integer>>",
        "whereMNId": "int",
        "numTowers": "ArrayList<Integer>",
        "money": "int",
        "generalReserve": "int",
        "towerColors": "ArrayList<TowerColors>",
        "characterNames": "ArrayList<String>",
        "characterCost": "ArrayList<Integer>",
        "characterDenyCards": "ArrayList<Integer>",
        "characterCardsDescription": "ArrayList<String>",
        "squads": "ArrayList<Squads>"
},
```

PROTOCOLLO DI RETE – GRUPPO AM07

ServiceMessage:

```
{
        "MessageType": "MATCH_JOINED / END_GAME",
        "message": "String",
        "playerID": "int"
}
```

ServiceMessage:

```
{
        "MessageType": "QUIT / KO / OK / GAMEPHASE_UPDATE / UNAVAILABLE_USERNAME",
        "message": "String",
        "playerID": "-1"
}
```

ServiceMessage:

```
{
        "MessageType": "PING / LOGOUT / USERNAME_ACCEPTED",
        "message": "none",
        "playerID": "int"
},
```

PlayerMoveMessage:

```
{
        "MessageType": "COLOR_VALUE / MOVE_STUDENT_TO_ISLE / MOVE_STUDENT_TO_DINING /
        PLAY_ASSISTANT_CARD / PLAY_CHARACTER_CARD / ACTIVATE_ATOMIC_EFFECT /
        MOVE_MOTHERNATURE / CHOOSE_CLOUD / GAMEPHASE_UPDATE",
        "playerID": "int",
        "genericValue": "int"
},
```

PROTOCOLLO DI RETE – GRUPPO AM07

AssistCard_UpdateMsg:

```
{
        "MessageType": "ASSISTANTCARD_UPDATE",
        "playerID": "int",
        "turnOrderPlayed": "int",
        "movementMNPlayed": "int",
        "turnOrders": "ArrayList<Integer>",
        "movementsMN": "ArrayList<Integer>",
},
```

StudentToDining_UpdateMsg:

```
{
        "MessageType": "STUDENTTODINING_UPDATE",
        "playerID": "int",
        "entrance": "HashMap<RealmColors,Integer>",
        "dining": "HashMap<RealmColors,Integer>"
},
```

StudentToIsle_UpdateMsg:

```
{
        "MessageType": "STUDENTTOISLE_UPDATE",
        "playerID": "int",
        "entrance": "HashMap<RealmColors,Integer>",
        "isleID": "int",
        "isleStudents": "HashMap<RealmColors,Integer>"
},
```

Professor_UpdateMsg:

```
{
        "MessageType": "PROFESSOR_UPDATE",
        "professors": "ArrayList<HashMap<RealmColors,Integer>>"
},
```

Money_UpdateMsg:

```
{
        "MessageType": "MONEY_UPDATE",
        "playerID": "int",
        "playerMoney": "int",
        "generalMoneyReserve": "int"
},
```

PickFromCloud_UpdateMsg:

```
{
        "MessageType": "CLOUDCHOICE_UPDATE",
        "playerID": "int",
        "entrance": "HashMap<RealmColors,Integer>",
        "cloudID": "int",
        "emptyCloud": "ArrayList<HashMap<RealmColors,Integer>>",
},
```

DenyCard_UpdateMsg:

```
{
        "MessageType": "DENYCARD_UPDATE",
        "grandmaIndex": "int",
        "cardCost": "int”
        "denyCardsOnCard": "int",
        "isleID": "int”,
},
```

FillCloud_UpdateMsg:

```
{
       "MessageType": "FILLCLOUD_UPDATE",
        "clouds": "ArrayList<HashMap<RealmColors,Integer>>",
},
```

MNMovement_UpdateMsg:

```
{
        "MessageType": "MNMOVEMENT_UPDATE",
        "totalIsles": "int",
        "students": "ArrayList<HashMap<RealmColors,Integer>>",
        "towerColors": "ArrayList<TowerColors>",
        "whereMNId": "int",
        "denyCards": "ArrayList<Booleans>",
        "numberOfIsles": "ArrayList<Integer>",
        "numberOfTowers": "ArrayList<Integer>"
},
```

CharacterCard_UpdateMsg:

```
{
        "MessageType": "CHARACTERCARD_UPDATE",
        "characterCardID": "int",
        "cardName": "CharacterCardsName",
        "cardCost": "int",
        "playerID": "int",
        "generalReserve": "int",
        "playerMoney": "int",
        "denyCards": "int",
        "studentsOnCharacter": "HashMap<RealmColors,Integer>"
},
```

EffectActivation_UpdateMsg (MONK / JESTER):

```
{
        "MessageType": "EFFECTACTIVATION_UPDATE",
        "characterCardIndex": "int",
        "cardCost": "int",
        "denyCardsOnPlace": "int",
        "studentsOnCard": "HashMap<RealmColors,Integer>",
         "id": "int",
        "studentsInPlace": "ArrayList<HashMap<RealmColors,Integer>>"
},
```

EffectActivation_UpdateMsg (FARMER):

```
{
        "MessageType": "EFFECTACTIVATION_UPDATE",
        "professors": "ArrayList<HashMap<RealmColors,Integer>>"
},
```

EffectActivation_UpdateMsg (HERALD):

```
{
        "MessageType": "EFFECTACTIVATION_UPDATE",
        "totalIsles": "int",
        "students": "ArrayList<HashMap<RealmColors,Integer>>",
        "towerColors": "ArrayList<TowerColors>",
        "whereMNId": "int",
        "denyCards": "ArrayList<Boolean>",
        "numberOfIsles": "ArrayList<Integer>",
        "numberOfTowers": "ArrayList<Integer>"
},
```

EffectActivation_UpdateMsg (MAGICAL_LETTER_CARRIER):

```
{
        "MessageType": "EFFECTACTIVATION_UPDATE",
        "playerID": "int",
        "turnOrder": "int",
        "mnMovement": "int"
},
```

EffectActivation_UpdateMsg (GRANDMA_HERBS):

```
{
        "MessageType": "EFFECTACTIVATION_UPDATE",
        "characterCardIndex": "int",
        "cardCost": "int",
        "isleID": "int",
        "denyCard": "int",
        "denyCardsOnPlace": "int",
        "studentsOnCard": "HashMap<RealmColors,Integer>"
},
```

EffectActivation_UpdateMsg (CENTAUR / KNIGHT / FUNGIST):

```
{
         "MessageType": "EFFECTACTIVATION_UPDATE"
},
```

EffectActivation_UpdateMsg (MINSTREL / THIEF):

```
{
         "MessageType": "EFFECTACTIVATION_UPDATE",
         "students": "ArrayList<HashMap<RealmColors,Integer>>",
         "studentsInDining": "ArrayList<HashMap<RealmColors,Integer>>"
},
```

EffectActivation_UpdateMsg (SPOILED_PRINCESS):

```
{
        "MessageType": "EFFECTACTIVATION_UPDATE",
        "characterCardIndex": "int",
        "cardCost": "int",
        "denyCardsOnPlace": "int",
        "studentsOnCard": "HashMap<RealmColors,Integer>",
        "studentsInDining": "ArrayList<HashMap<RealmColors,Integer>>"
},
```

GamePhase_UpdateMsg:

```
{
        "MessageType": "GAMEPHASE_UPDATE",
        "activePlayer": "int",
        "activePlayerNickname": "String",
        "gamePhases": "GamePhases",
        "actionPhases": "ActionPhases"
}
```