Relazione wordle Gioele Modica (614923)

ServerMain

Il server dal file “Settings.txt” estrae il numero di porta su cui aprire la connessione, l’indirizzo multicast, la porta multicast e l’intervallo di tempo che deve intercorrere tra la generazione di una parola e l’altra. Dal file “Words.txt” il server estrae e salva in un’ arraylist, la lista di tutte le parole segrete, che utilizzerà sia per generare la nuova parola segreta, che per verificare se il tentativo dell’utente è valido. Il ServerMain successivamente avvierà diversi threads: Un thread per la generazione della parola segreta (ChangeWord), uno per simulare l’intervallo di tempo tra una parola e l’altra (TimeSimulator) e in fine utilizza una threadpool per generare dei ServerHandler che gestiranno i diversi client che parteciperanno al gioco (Il server quindi è in grado di gestire contemporaneamente più client). La terminazione del server non è stata gestita in quanto è pensato per essere sempre attivo.

ChangeWord

Il costruttore di questa classe prende come parametro la lista di tutte le secret words e l’intervallo di tempo che deve intercorrere tra la generazione di una parola e l’altra. In modo randomica ogni “x” MS viene estratta una secrete word, se la parola non è già stata estratta viene settata come nuova parola segreta, altrimenti viene effettuata un ulteriore estrazione. Per simulare l’intervallo che intercorre tra una generazione e l’altra viene avviato un thread (**TimeSimualtor**) che ha il solo scopo di effettuare una sleep di “x” ms per poi terminare, solo dopo la sua terminazione verrà generata una nuova parola e sarà riavviato il thread di simulazione del tempo. Inoltre questa classe ha un metodo che permette agli handler del server di ottenere la parola segreta corrente quando richiesta.

Multicast

Questa classe viene utilizzata per inviare messaggi UDP in multicast a tutti i client attualmente “loggati”, inviando una stringa con le statistiche di un client che ha deciso di pubblicare.

DatiUtente

Il costruttore di questa classe prende come parametro l’username e la password dell’utente. Questa classe contiene tutti i campi necessari per manipolare un utente registrato (e anche per registrarlo) così da poter scrivere/leggere utenti sul file Registrazione.json. Questa classe come metodi ha i getter e setter di ogni variabile.

```
    String username;
    String password;
    boolean haGiocato; //se un utente ha già giocato con la parola corrente
```

```
    int tentativi; //numero di tentativi con la parola corrente
    boolean login; //se l'utente è già loggato
    int paroleIndovinate; //numero di parole indovinate fino adesso
    String lastWord=""; //ultima parola con cui l'utente ha giocato
    //Boolean share; //
    int[] storico={0,0,0,0,0,0,0,0,0,0,0,0}; //ogni pos identifica il numero di
tentativi, il valore invece il numero di partite indoviante
```

ServerHandler

Questo thread ha lo scopo di gestire in modo sincronizzato le richieste di un client. Il costruttore della classe prende come paramentro:

```
Socket client,ChangeWord secret,ArrayList<String> lista_parole,Multicast
multi,String ipMulti,int portMulti)
```

Ovvero la socket su cui è connesso il client, l'oggetto che permette di ottenere la nuova secrete word, il dizionario di parole riconosciute dal sistema, l'oggetto multicast, l'ip multicast e la porta multicast. Tutti i metodi che andranno a manipolare il file json sono sincronizzati.

Il server tramite due buffer scambia messaggi con il client, presentando inizialmente un menù in cui il client può effettuare "registrazione, login, chiusura connessione". Quando il client richiede di effettuare la registrazione, invia username e password, l'handler tramite la funzione ***Registrazione*** carica in memoria tutti gli utenti già registrati nel file json, verifica che non vi siano utenti con username uguale a quello scelto dal client (invierà messaggio "not ok" in caso vi sia già presente) , aggiunge il nuovo utente e riscrive il file json aggiornato. In caso di successo verrà inviato il messagio "ok" al client e si passerà direttamente alla fase di login. Se la password scleta dall'utente è vuota viene inviato un messaggio di errore.

Dopo aver ricevuto username e password la funzione ***Login*** carica in memoria tutti gli utentei contenuti nel file json e verifica che vi sia corrispondenza. In caso di corrispondenza viene verificato anche che l'utente non abbia già effettuato il login. In caso di errore viene segnalato al client, altrimenti si passa direttamente alla fase di gioco.

Durante la fase di Gioco (dopo il login) vi è uno switch in cui su richiesta del client può: Tentare di fare indovinare la parola segreta, inviare le statistiche dell'utente che sta gestendo, condividire in multicast le statistiche del client che sta gestendo, permettere di fare il logout all'utente.

La funzione ***Gioca*** verifica se il client ha già giocato con questa parola, in caso invia un messaggio contenente "stop", Altrimenti aggiorna i parametri di gioco salvati nell'utente per permettergli di giocare (invio del messaggio "start"). Finchè i tentativi dell'utente sono inferiori a 12 ogni utente può inviare una parola: se la parola è inferiore o superiore a 10 caratteri o non è presente nnella lista viene inviato un messaggio contente "again" e non vengono sprecati tentativi, se la parola invece è presente nella lista viene incrementato di

1 i tentativi dell'utente e se la parola non è corretta viene confrontata carattere per carattere con la secrete word generando dei suggerimenti che verranno inviati al client. In caso di corrispondenza con la secrete word viene inviato un messaggio contenente"ok", e vengono aggionrate le statistiche del client. In caso il client dovesse termianre i tentativi viene inviato il messaggio "not ok". Il client gioca con la stessa secrete word finchè non esce dalla funzione di gioco tramite il comando "exit" o con i modi speicificati sopra.
La funzione ***modifica*** permette di manipolare le informazioni di gioco di un client all'interno del file json. Carica la lista di utenti in memorai, trova corrispondenza con l'utente attualmente gestito e in base al parametro "cambio" può eseguire:

- 0 -> Cambio stato di login di un client (true/false)
- 1 -> Incremento tentativi client
- 2 -> L'utente ha già giocato con la parola corrente viene settato a true
- 3 -> Azzeramento di tutte le varibili di gioco perchè vi è una nuova secrete word.
- 4 -> Incremento parole indovinate e azzeramento parametri di gioco.

In fine viene riscritto il file json con le informazioni aggiornate.

Il server effettua molte istruzioni di I/O per evitare di tenere permanentemente la memoria RAM carica, ma caricare i dati solo quando effettivamente necessario.

ClientMain

Il client dal file "Settings.txt" estrae l'indirizzo ip del server il numero di porta del server, l'indirizzo multicast e la porta multicast. Il client tramite due buffer scambia messaggi con il server e inizialmente gli sono permesse 3 azioni: Registrazione, Login e chiusura connessione. Tramite la funzione ***Registrazione*** il client invia il proprio username e password e in caso non dovesse ricevere messaggi di errore da parte del client passa alla fase di Login. La funzione ***Login*** invia al serve username e password e in caso non dovesse ricevere messaggi di errore passa alla fase di gioco. Durante la fase di gioco il client può: tentare di indovinare la parola segreta, richeidere le proprie statistiche, condividire in multicast le statistiche proprie, stampare i messaggi udp ricevuti fino a quel momento ed effettuare il logout. Tramite la funzione ***gioca*** se non riceve nessun messaggio di errore, il client può inviare le proprie guess word, o il comando exit per uscire prima dell'esaurimento dei tentativi. Il client inoltre sfrutta avvia un thread della classe **Statistiche** che resta in attesa di ricezione di messaggi udp da parte del server contenti le statistiche condivise dagli altri client in gioco.

Come utilizzare il client

Nel menù iniziale intuitivamente si può inserire “1” per effettuare la registrazione (successivamente verranno richiesti username e password), “2” per effettuare il login (successivamente verranno richiesti username e password), “3” per terminare la connessione.
Nel secondo menù è possibile inserire “1” per tentare di indovianre la parola, “2” per richiedere le statistiche personali, “3” per condividre le proprie statistiche con il server, “4” per stampare tutte le notifiche ricevute dal server, “5” per effettuare il logout. Se viene selezionato il comando “1” allora è possibile inserire la parola che si pensi corrisponda alla secrete word o il comando “exit” per uscire dalla funzione di gioco anticipatamente (sarà consentito giocare ancora con la stessa parola se non è stato effettuato il logout).

Librerie esterne

Gson-2.6.2.jar

Comandi di compilazione

Dopo essersi posizionati nella cartella "src":

Compilazione:

```
javac -cp ".;gson-2.6.2.jar" wordleServer/*.java
javac -cp ".;gson-2.6.2.jar" wordleClient/*.java
```

Esecuzione

```
java -cp ".;gson-2.6.2.jar" wordleServer/ServerMain
java -cp ".;gson-2.6.2.jar" wordleClient/ClientMain
```