WORDLE 3.0

PROGETTO LABORATORIO 3

Buccolieri Matteo (618045)

Il progetto è composto da due classi principali, **WordleClientMain** (n.b. da qui in avanti "client") e **WordleServerMain** (n.b. da qui in avanti server), che comunicano tra di loro grazie ad una connessione TCP persistente implementata con l'utilizzo di Java I/O.

Il server utilizza la funzione readConfig() per leggere e salvare i parametri di configurazione da un file di testo (server.properties), successivamente controlla l'esistenza di un file "users.json" nel quale devono essere presenti gli utenti registrati, se il file non esiste (n.b. nessun utente si è ancora registrato), viene creato. Successivamente scarica dal file "users.json" tutti gli utenti e le loro informazioni e le inserisce in una struttura dati.

Successivamente inserisce all'interno di una struttura dati predisposta tutte le parole che possono essere giocate, scaricandole dal file "words.txt" attraverso la funzione create_dictionary(File).

Dopo aver creato queste strutture dati il server avvia un TimerTask (ChoseWord) utilizzando la funzione scheduledAtFixedRate. La classe ChoseWord:

- Sceglie una parola dal dizionario in modo random utilizzando la funzione chose_word(dictionary).
- Permette a tutti gli utenti di rigiocare utilizzando la funzione now_can_play_again(users).
- Aggiorna il file users.json utilizzando la funzione backup_server_users.

Successivamente crea un threadPool che crea un WordleTask, thread che si occupa della gestione di una singola sessione di gioco.

Il client utilizza la funzione readConfig() per leggere e salvare i parametri di configurazione da un file di testo (client.properties), apre uno Scanner grazie al quale cattura gli input da tastiera dell'utente e successivamente si collega al server e interagisce con il thread WordleTask.

Le comunicazioni tra client e WordleTask avvengono tramite DataInputStream e DataOutputStream, la maggior parte delle volte attraverso writeInt e readInt per comodità, fanno eccezione il passaggio dell'username e della password che, essendo stringhe, avviene tramite writeUTF e readUTF.

Il client è formato da un ciclo while che permette la scelta della prima azione, se l'utente decide di registrarsi e la registrazione va a buon fine si rientrerà nello stesso ciclo while in modo che l'utente possa anche loggarsi nella stessa sessione. Qualora l'utente decidesse di loggarsi il metodo login() è composto da un while con all'interno vari if else per gestire le varie casistiche e da un secondo ciclo while nel momento in cui avviene il login. All'interno di questo secondo while troviamo uno switch che gestisce il secondo menu del gioco.

- playWordle(): Formato da un ciclo while che rimane vero finché la partita non termina, all'interno di questo ciclo troviamo vari else if per gestire le varie casistiche.
- seeStatistic(): riceve dal server le statistiche dell'utente e le restituisce all'utente.
- share: passa il numero di tentativi utilizzati per indovinare la parola corrente al server.
- showMeSharing: chiama la funzione print_mess sull'oggetto mess_receiver.
- logout: chiude la connessione.

La classe WordleTask è formata da un ciclo while nel quale troviamo una serie di else if che permettono di rispondere alle richieste che l'utente fa nel primo menù. Il primo if viene utilizzato per la registrazione ed al suo interno è posto un while che termina nel momento in cui la registrazione viene effettuata con successo. Il secondo if viene utilizzato per loggarsi, al suo interno è posto un while che permette di loggarsi effettivamente e successivamente un secondo while che terminerà al momento del logout. In questo while è presente uno switch che permette di gestire le richieste che effettua l'utente.

La parola da indovinare viene passata ad ogni classe attraverso la classe SecretWord che memorizzerà la parola ogni volta che cambia e usufruisce dei getter e setter per inviare e cambiare la parola.

Il crash lato server viene gestito impostando il valore is_logged di ogni utente a false nel momento in cui il server viene riavviato, poiché con il crash del server verranno chiusi tutti i

client.

Il crash lato client viene gestito in WordleTask con un try-catch, impostando is_logged dell'utente crashato a false.

Scelte implementative

Nel momento in cui un utente avvia l'applicazione verrà accolto da un menù iniziale ed avrà 3 scelte a disposizione:

- **REGISTER:** se l'utente non è ancora registrato può farlo attraverso il comando "register", il programma richiederà di inserire un nuovo username, questo username non può essere vuoto e non può essere uguale all'username di un altro utente, in questi casi il programma segnalerà all'utente il problema e lo inviterà ad inserire un nuovo username. Se l'username scelto soddisfa i criteri il programma invita l'utente ad inserire una password, anche quest'ultima non potrà essere vuota, nel caso in cui l'utente dovesse inserire una password vuota il programma segnalerà il problema ed inviterà l'utente ad immettere una nuova password.
- **LOGIN:** se l'utente è già registrato potrà loggarsi attraverso il comando "login", il programma richiederà all'utente utente di inserire un username, se l'username inserito non viene trovato dal server nella struttura dati predisposta per il salvataggio degli utenti il programma invierà un messaggio di errore ed inviterà l'utente ad inserire un altro username. Se l'username inserito è valido il programma richiederà una password di accesso, se la password risulterà errata verrà inviato un messaggio di errore e verrà richiesto l'inserimento della password nuovamente, in caso contrario l'utente accederà al gioco e potrà scegliere tra una serie di funzionalità che vedremo in seguito. Se un utente è già loggato su un dispositivo non li sarà possibile ri-loggarsi su un altro dispositivo, se non dopo aver eseguito il logout dal primo.
- **EXIT:** Se un utente desidera chiudere il programma senza loggarsi o registrarsi può facilmente farlo attraverso il comando "exit".

Dopo aver effettuato il login l'utente potrà scegliere tra una serie di comandi:

- **PLAY:** l'utente avvia il gioco, dovrà indovinare una parola di 10 caratteri in un massimo di 12 tentativi. La parola da indovinare cambia ogni tot di tempo, un utente non può

giocare la stessa parola più di una volta, se prova a giocare la stessa parola un messaggio di errore glielo impedirà.

- Se l'utente immette una parola con un numero di caratteri diversi il programma invierà un messaggio di errore per segnalarlo, ma non verrà considerato come un tentativo di risoluzione.
- Se l'utente immette una parola inesistente il programma invierà un messaggio di errore per segnalarlo, ma non verrà considerato come un tentativo di risoluzione.
- Se durante la partita la parola viene cambiata l'utente riceverà un messaggio di errore nel momento in cui immette una parola e la partita non verrà considerata giocata.
- Qualora la parola dovesse essere valida ma non corretta il programma restituirà una stringa che farà capire all'utente se una lettera è presente e nella posizione giusta, presente e nella posizione errata o assente (n.b "+" se presente e nella posizione giusta, "?" se presente ma nella posizione errata, "X" se assente).
- Se l'utente indovina la parola riceverà un messaggio che glielo comunica e visualizzerà nuovamente il menù.
- Se l'utente sta giocando una partita non può uscire da essa se non perdendo o vincendo, qualora dovesse presentarsi un crash la partita non verrà considerata giocata.

- **SEE STATISTIC:** L'utente potrà visualizzare tutte le statistiche aggiornate all'ultima partita giocata, per la precisione:
  - Username
  - Se la parola giocabile in quel momento è stata giocata
  - Partite giocate
  - Partite vinte
  - Percentuale vittorie
  - Serie di vittorie attuale
  - Serie di vittorie più lunga

    - Guess distribution

- **SHARE:** Solo dopo che l'utente avrà giocato la parola corrente potrà condividere i suoi risultati con gli altri utenti, se l'utente dovesse provare a condividere i risultati prima di aver giocato un messaggio di errore glielo impedirà.
- **SHOW ME SHARING:** Permette all'utente di vedere i risultati degli altri utenti che hanno deciso di condividere i loro risultati dopo aver giocato.
- **LOGOUT:** L'utente uscirà dall'applicazione.

STRUTTURE DATI

- User: oggetto che raccoglie tutte le informazioni di un utente. Per tener traccia della guess distribution al suo interno è posta una HashMap<Int,Int> dove il primo Int indica il numero di tentativi e il secondo il numero di parole indovinate con quel numero di tentativi.
- users: ArrayList<User> nella quale sono presenti tutti gli utenti.
- dictionary: ArrayList<String> contentente tutte le parole del file words.txt.
- repository: ArrayList<String> contenente tutti i messaggi inviati i udp dagli utenti.

METODI SINCRONIZZATI

I metodi sincronizzati sono quelli che vanno ad agire sulla users list:

- check(username, users): metodo che controlla dell'esistenza di un username nella lista
- search_user(username, users): metodo che recupera l'utente dalla lista.
- backup_server_users: aggiorna il file users.json.

COME COMPILARE ED ESEGUIRE

- Javac: Per compilare utilizzando Java utilizzare:

```
javac -cp "./../libs/gson-2.10.jar" ./*.java -d ./../bin
```

Verrà visualizzato il messaggio: “Note: ./Mess_Receiver.java uses or overrides a deprecated API.” a causa dell’utilizzo di un metodo deprecato.

Per eseguire il server utilizzare:

```
java -cp .:./../libs/gson-2.10.jar WordleServerMain
```

Per eseguire il client utilizzare:

```
java -cp .:./../libs/gson-2.10.jar WordleClientMain
```

- Jar: Per creare i file jar ho utilizzato:

```
jar cfm Server.jar Server-Manifest.txt *.class
```

```
jar cfm Client.jar Client-Manifest.txt *.class
```

Per eseguire il server jar:

```
java -jar Server.jar
```

Per eseguire il client jar:

```
java -jar Client.jar
```