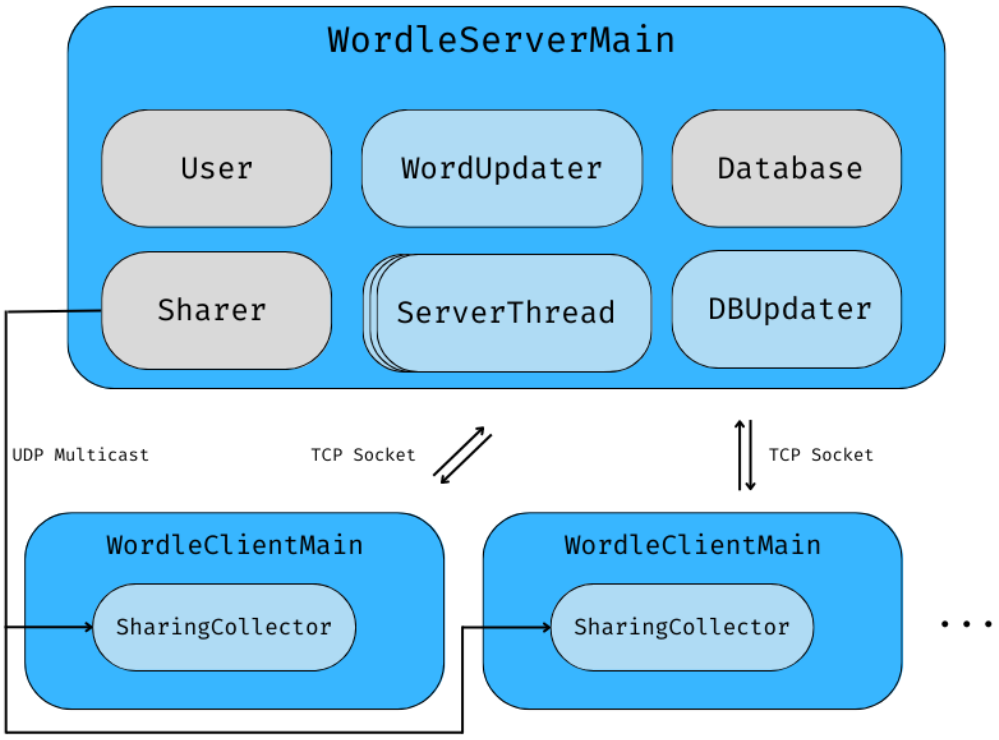
Laboratorio 3
Relazione sul progetto d'esame

Leonardo Crociani
Matricola: 615392

Sommario

Implementazione del sistema di gioco di **Wordle** in Java.

Parte I
Lo schema generale



Legenda

- Classe con metodo `main()`.
- Classe che implementa `Runnable` (thread).
- Classe non-thread.

Descrizione
La classe `WorldeServerMain`:

- Crea:
  - Un oggetto `Database` (che si serve della classe `User`)
  - Un oggetto `Sharer` per la condivisione dei risultati nel gruppo di multicast
- Lancia:
  - Un thread `DBUpdater` che si occupa a intervalli di tempo prefissati di scrivere il database aggiornato nel file `db.json`
  - Un thread `WordUpdater` che aggiorna periodicamente la parola segreta
- Costruisce una `CachedThreadPool` di `ServerThread` per gestire i client e si mette in ascolto su una socket TCP.

La classe `WordleClientMain`:

- Lancia un thread `SharingCollector` che si mette in attesa di notifiche (pacchetti UDP) dallo `Sharer`
- Instaura una connessione TCP con `WordleServerMain`, il quale, assegna al client un `ServerThread`

Parte II
Le scelte di progettazione e le strutture dati

Properties File
Le specifiche richiedono che server e client debbano inizializzare le variabili di configurazione autonomamente.
Per soddisfare la specifica, le classi principali `WordleServerMain` e `WordleClientMain` ottengono le opzioni di inizializzazione da due file speciali:
`server.properties` e `client.properties`.

CachedThreadPool
Dovendo servire un numero sconosciuto di utenti, ho optato per una `CachedThreadPool` per l'esecuzione dei `ServerThreads`. Questa è di più facile adattamento al numero di richieste rispetto ad una `FixedThreadPool`.

Database
Il database è implementato da due classi. La classe `Database` e la classe `DBUpdater`. La prima mantiene come attributo il database virtuale (una `HashMap<String, User>`) e offre metodi pubblici statici per la ricerca di utenti, per il loro aggiornamento e per il loro inserimento.
Periodicamente (la frequenza d'aggiornamento è specificata nel file `server.properties`) il database virtuale viene scritto sul file `db.json` dalla classe `DBUpdater`.

Il suggerimento
Quando un client sottomette una parola diversa dalla secretWord, il server invia al client un suggerimento. Quest'ultimo viene calcolato in base alla posizione dei caratteri della guessedWord in relazione a quella dei caratteri della secretWord. Nel calcolo si tiene conto anche del numero di occorrenze di ogni lettera. La stringa di suggerimento ha 10 caratteri (che codificano colori o simboli - specificato in `client.properties`). L' i-esimo carattere della stringa indica se l'i-esima lettera della guessedWord è:

- Presente nella secretWord e in posizione corretta (colore verde o '+')
- Presente nella secretWord ma in posizione errata (colore giallo o '?')
- Assente nella parola segreta (nessun colore o 'x')

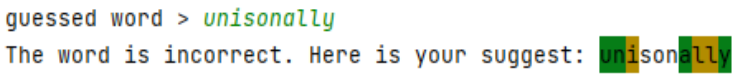


Figura 1: Un esempio di suggerimento con colori

Password Hashing
Per proteggere le informazioni nel database e sulla rete, viene calcolato l'hash della password con SHA-256 (è il client a farlo).

Notifiche (Sharing)
Anche le notifiche vengono inviate come una singola stringa di simboli. Se la funzione dei colori è attiva (si veda l'ultimo paragrafo) il client mostra sulla command line una versione colorata. In caso contrario, verranno mostrati solamente i simboli, la cui semantica è spiegata nel paragrafo "Il suggerimento".

Una notifica si compone dei tentativi fatti da un giocatore fino al raggiungimento della soluzione del gioco. Ogni tentativo, codificato come stringa, è raccolto dal server in una `ArrayList`. Quando il client lo richiede, le stringhe sono inviate in multicast sottoforma di notifica.

Figura 2: Un esempio di notifica.

Terminazione
Quando un client richiede la terminazione del gioco, digitando `exit()`, viene inviato al `ServerThread` una stringa contenente "EXIT". Questa indica al `ServerThread` di chiudere la socket di comunicazione e terminare al sua esecuzione.

Parte III
Sincronizzazione

Database
Data l'esistenza di più threads, ognuno dei quali opera sulla HashMap del DB, gli accessi vanno sincronizzati. Ho utillizzato il meccanismo dei `monitor` per garantire la mutua esclusione quando necessario.

Parte IV
Manuale d'uso

Tutte le operazioni vanno eseguite dalla cartella contenente sorgenti e files.

Compilazione

- Per il client digitare `javac WordleClientMain.java`
- Per il server (OS: Windows) :
  `javac -cp ";gson-2.10.jar" WordleServerMain.java`

Esecuzione dei file compilati

- Per il client scrivere `java WordleClientMain`
- Per il server (OS: Windows) digitare:
  `java -cp ";gson-2.10.jar" WordleServerMain`

Esecuzione dei file `.jar` (alternativa alle due fasi precedenti)

- Per il client scrivere `java -jar Client.jar`
- Per il server digitare `java -jar Server.jar`

Interazione con la Command Line del client
La command line si comporterà come una shell. I comandi verranno valutati ed eseguiti. Per l'elenco dei comandi, a programma avviato, digitare `help()` oppure proseguire con la lettura.
Lista comandi:

- `register(username,password)`
- `login(username,password)`
- `logout()` - non è necessario fornire l'username
- `playWordle()` - per l'inizializzazione del gioco
- `sendWord()` - eseguibile dopo aver chiesto l'inizializzazione con `playWordle()`. Sarà necessario immettere la parola e premere nuovamente invio.
- `sendMeStatistics()` - per ricevere le statistiche del giocatore autenticato
- `share()` - eseguibile una volta vinto il gioco.
- `showMeSharing()` - le notifiche verranno mostrate una volta che si è autenticati
- `exit()` - per terminare il processo e uscire dal gioco

Attivazione dei colori
I colori delle notifiche e quelli che codificano i suggerimenti sono disabilitati di default.
Sono però abilitabili settando a 1 la variabile `colors` in `client.properties`.
Si consiglia di settare i colori solo nei seguenti ambienti di test:

- Powershell (nuova versione) di Windows 11
- Terminale di IntelliJ IDEA
- Terminale di Eclipse

In tutti gli altri casi, la funzionalità dei colori non è garantita e ne sconsiglio l'attivazione.