RELAZIONE PROGETTO
RETI DI CALCOLATORI
A.A. 2021/2022

CECCHETTI CHIARA
583054

STRUTTURA:

All'interno della cartella "Progetto" sono presenti:

- Cartella "Client" contenente i file .java relativi al client
- Cartella "Server" contenente i file .java relativi al server
- Cartella "lib" contenente:
  - la libreria "gson-2.8.2.jar", libreria esterna utilizzata dalla classe WinsomeServer.java per le operazioni di serializzazione e deserializzazione.
  - file .jar eseguibili per client e server.
- Files di configurazione "configServer.txt" e "configClient.txt" (per maggiori informazioni leggerne il contenuto).

LATO SERVER:

Di seguito verranno elencate le varie classi presenti nella cartella server e per ognuna ne verrà descritto il funzionamento.

- **MainServer.java**: è il main del server, questa classe si occupa dell'esecuzione del server, dopo aver settato i vari valori riguardo le varie connessioni leggendo il relativo file di configurazione, attiverà i relativi Thread per la gestione della rete sociale i cui task (implementano Runnable) sono:

  - **HandlerClients.java** → si occupa delle connessioni dei vari client al server, istanziando un CachedThreadPool (scelto per la sua elasticità nel creare e rimuovere thread) a cui, ogni volta che una nuova connessione viene instaurata, verrà sottoposto un nuovo Task **Comunicazione.java** che si occuperà di gestire una singola connessione con un client e permettere quindi a quest'ultimo di interagire con la rete sociale; questo task inizialmente esporterà l'oggetto remoto del server e invierà al client le informazioni per reperire quest'ultimo, successivamente entrerà in un ciclo di attesa dove, all'arrivo di un comando eseguirà le relative richieste del client, andando a chiamare i metodi della classe **WinsomeServer.java** che restituirà i risultati delle operazioni e che verranno poi inoltrate al client.
    All'interno del task HandlerClients.java prima di ciclare sull'accettazione delle connessioni viene creato e istanziato il Register RMI per permettere ai vari client di usufruire dei metodi remoti specificati in **WinsomeServerInterface.java** e implementati in WinsomeServer.java per registrarsi alla rete e alle notifiche.

  n.b la classe **WinsomeServer.java** oltre a contenere i metodi per permettere al client di interagire con la rete sociale si occuperà del recupero dati dalla cartella **Files** (per la prima istanza di WinsomeServer.java quindi al riavvio del server)

  - **MemoryTask.java** → si occupa dell'aggiornamento periodico dei file contenenti le informazioni riguardo la rete sociale nella cartella **Files** chiamando il metodo statico updateData() all'interno della classe WinsomeServer.java.

  - **CalcoloRicompense.java** → si occupa del calcolo periodico delle ricompense, avverte i relativi client iscritti al gruppo Multicast e aggiorna i portafogli di coloro che hanno partecipato attivamente nella rete sociale.

Per terminare il server basta da linea di comando inserire il comando “off” e il main provvederà a interrompere i vari thread attivati e terminare l’esecuzione.

- **Post.java** classe rappresentante un post all’interno della rete sociale con autore del post, titolo, contenuto, se è rewin, lista commenti, lista voti, data/ora, numero like/dislike e id del post.
  Contiene i metodi relativi all’aggiunta di commenti e like/dislike
- **Commenti.java** classe rappresentante un commento di un post con contenuto commento, data/ora e autore commento
- **User.java** classe rappresentante un utente della rete sociale con username, password e lista di tags.
- **Voti.java** classe rappresentante un voto con valore data/ora e autore del voto
- **Wallet.java**: classe rappresentante un portafoglio associato a un utente, contiene metodi per aggiornare i saldi e la cronologia delle transazioni

LATO CLIENT

- **MainClient.java**: è il main del client, inizialmente istanzierà una classe di tipo **WinsomeClient.java** contenente i metodi specificati dall’interfaccia **WinsomeClientInterface.java** per aggiornare i followers tramite callback, successivamente verrà letto il file “configClient.txt” per settare i vari parametri di comunicazione, creato e istanziato il registro RMI e successivamente esportato l’oggetto remoto del client per permettere al server di aggiornare i followers.
  Dopo aver instaurato la connessione e ricevuto i valori per istanziare l’oggetto remoto del server (per poter chiamare i metodi remoti di registrazione) si procederà alla fase di registrazione/login.
  Se non viene “superata” la fase di registrazione/login l’applicazione terminerà perché non sarà possibile usufruire delle funzionalità della rete.

  Dopo un’eventuale successo nella registrazione/login (registrazione alle notifiche automatica) sarà possibile interagire con la rete sociale.
  Verrà attivato un Thread per la gestione del multicast (iscrizione e uscita dal gruppo) e la ricezione dei messaggi con il task **MulticastTask.java**

  Successivamente alla creazione del Thread si entrerà in un ciclo dove finché il comando non è quello di “logout” verrà controllata la sintassi delle richieste da inviare al server, solo nel momento di un comando corretto quest’ultimo verrà inoltrato, altrimenti verrà stampata la lista dei comandi disponibili e si ritornerà in attesa di una nuova richiesta.
  Al momento dell’inserimento del comando di “logout” avverrà la disattivazione delle notifiche, l’uscita dal gruppo Multicast e il client terminerà la sua esecuzione.

N.b: nello scrivere il comando Post <title> <Content> necessario apportare il carattere “;” per scindere tra titolo del post e contenuto.

COMPILAZIONE E ESECUZIONE

Posizionarsi all'interno della cartella "583054 CECCHETTI CHIARA":

Step 1: compilare

- Ubuntu/OS :
  - Server:
    - javac -cp .:Progetto/lib/gson-2.8.2.jar:. Progetto/Server/*.java
  - Client:
    - javac Progetto/Client/*.java
- Windows:
  - Server:
    - javac -cp .\Progetto\lib* .\Progetto\Server*.java
  - Client:
    - javac Progetto\Client*.java

Step 2: eseguire

- Ubuntu/OS:
  - Server:
    - java -cp .:Progetto/lib/gson-2.8.2.jar:. Progetto.Server.MainServer
    - java -jar Progetto/lib/serverJar.jar
  - Client:
    - java Progetto.Client.MainClient
    - java -jar Progetto/lib/clientJar.jar
- Windows:
  - Server:
    - java -cp .\Progetto\lib* Progetto.Server.MainServer
    - java -jar .\Progetto\serverJar.jar
  - Client:
    - java Progetto.Client.MainClient

- java -jar .\Progetto\lib\clientJar.jar

Se ci dovessero essere problemi nell'eseguire gli eseguibili .jar, dopo la compilazione dei vari file procedere con le seguenti istruzioni e successivamente eseguire come sopra.

- Windows:

jar -cfm .\Progetto\clientJar.jar /Progetto/Client/META-INF/MANIFEST.MF .\Progetto\Client*.class

jar -cfm .\Progetto\lib\serverJar.jar .\Progetto\Server\META-INF\MANIFEST.MF .\Progetto\Server*.class

- Ubuntu/OS:

jar -cfm Progetto/lib/clientJar.jar Progetto/Client/META-INF/MANIFEST.MF Progetto/Client/*.class

jar -cfm Progetto/lib/serverJar.jar Progetto/Server/META-INF/MANIFEST.MF Progetto/Server/*.class