ROULETTE CLIENT-SERVER IN JAVA

1-Scopo dell'applicazione:

L'applicazione Roulette Client-Server è un progetto che simula una roulette e permette a un giocatore (il client) di connettersi a un server e inserire le proprie scommesse e ricevere l'esito della giocata.

FUNZIONALITÀ PRINCIPALI:

Il client:

-Inserisce la puntata in € (saldo massimo disponibile)

-Scommette su: (Colore: "Rosso", "Nero" o "No", Pari/Dispari: "Pari", "Dispari" o "No", Numero: da 0 a 36, oppure -1 se non vuole scommettere sul numero)

-Deve selezionare almeno un'opzione

-Visualizza l'esito della giocata e il saldo aggiornato

-Decide se continuare con la partita corrente o iniziarne una nuova

Il server:

-Genera casualmente il numero della roulette (0-36)

-Determina il colore corrispondente: ("Verde" per lo 0, "Rosso" se dispari, "Nero" se pari)

-Confronta le scommesse del client con il numero estratto

-Aggiorna e invia il saldo al client

-Gestisce eventuali errori o disconnessioni

2-Protocollo e pacchetti usati:

L'applicazione usa un protocollo basato su pacchetti di testo inviati tramite socket TCP e ogni pacchetto contiene una singola informazione e il server/client li leggono in ordine.

Pacchetti inviati dal client:

Connessione iniziale: Il client stabilisce la connessione TCP sulla porta 12345

Puntata: Inviata come numero

Opzioni di gioco: Inviate in sequenza come stringhe e/o un intero per il numero

Decisione: Invia la richiesta continua o nuova partita per gestire il saldo del turno successivo

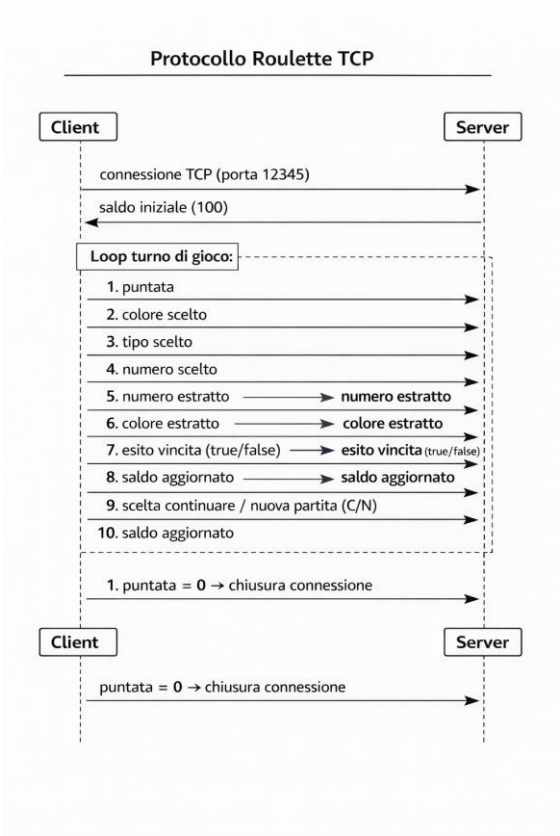
Chiusura: Inviando il valore 0 come puntata, il giocatore segnala l'intenzione di chiudere la connessione

Pacchetti inviati dal server:

Saldo iniziale: Inviato saldo (100€) non appena il client si connette

Risultato estrazione: Invia in ordine il numero estratto, il colore estratto, l'esito della vittoria e il saldo aggiornato

Conferma saldo: Invia il saldo aggiornato o resettato a 100€ in base alla scelta del client per la nuova partita



Realizzato con “Draw.io” (https://app.diagrams.net/)

Link GitHub: https://github.com/fedeeBoh4/Progetto-Roulette.git

3-Spiegazione codice nelle sue parti più importanti e logiche:

-Server:

```
while (true) {
    Socket socketClient = server.accept();

  //Avvio il thread per gestire la partita del singolo giocatore
    Partita partita = new Partita(socketClient);
    Thread thread = new Thread(partita);
    thread.start();
}
```

Questo blocco permette al server di gestire più giocatori contemporaneamente senza blocchi.

-Partita:

```java
int estratto = random.nextInt(37);
String colEstratto;

if (estratto == 0) {
    colEstratto = "Verde";
} else if (estratto % 2 == 0) {
    colEstratto = "Nero";
} else {
    colEstratto = "Rosso";
}

boolean vinto = true;

//Se non c'è nessuna scommessa attiva, il giocatore perde
if (colScelto.equalsIgnoreCase("No") && tipoScelto.equalsIgnoreCase("No") && numScelto == -1) {
    vinto = false;
} else {
    if (estratto == 0) {
        //Lo zero vince solo se indovinato come numero secco
        if (numScelto == 0) {
            vinto = true;
        } else {
            vinto = false;
        }
    } else {
        if (!colScelto.equalsIgnoreCase("No") && !colScelto.equalsIgnoreCase(colEstratto)) {
            vinto = false;
        }
        if (tipoScelto.equalsIgnoreCase("Pari") && estratto % 2 != 0) {
            vinto = false;
        }
        if (tipoScelto.equalsIgnoreCase("Dispari") && estratto % 2 == 0) {
            vinto = false;
        }
        if (numScelto != -1 && numScelto != estratto) {
            vinto = false;
        }
```

Questo è la logica del gioco del gioco, dove si calcola il risultato e si gestisce lo zero.

-Client:

```java
while (true) {
    System.out.print("Scegli Numero (0-36, -1 no): ");
    if (sc.hasNextInt()) {
        num = sc.nextInt();
        sc.nextLine();

        //Controllo se il numero è tra 0 e 36 oppure è esattamente -1
        if ((num >= 0 && num <= 36) || num == -1) {
            break;
        } else {
            System.out.println("Errore: Inserisci un numero tra 0 e 36 (o -1)");
        }
    } else {
        System.out.println("Errore: Inserisci un valore numerico");
        sc.next();
    }
}
```

Questo blocco garantisce che il server riceva solo dati corretti, evitando crash ed errori del sistema.

4-Controllo input:

Il controllo degli input avviene direttamente nel ciclo principale del client per bloccare errori prima dell'invio e il programma verifica che la puntata sia un numero valido e non superiore al saldo, in caso di errore avvisa l'utente e richiede il dato senza errori.
Per le scommesse testuali, viene usato “equalsIgnoreCase”, rendendo validi input come "Rosso", "rosso" o "ROSSO". Se l'utente non seleziona alcuna opzione (tutti "No" e numero -1), il server assegna automaticamente la perdita della puntata. Il client controlla anche che il numero scelto sia rigorosamente tra -1 e 36

LAVORO REALIZZATO DA: FEDERICO BALDINI & CRISTIAN TERZI