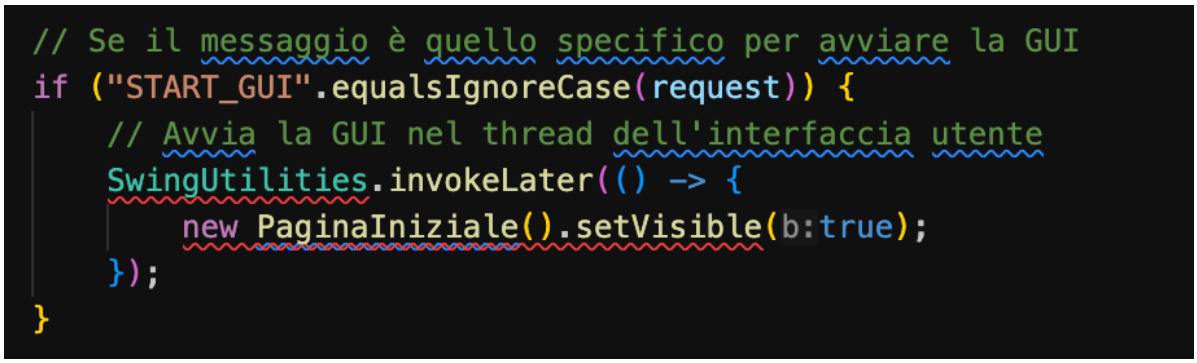
Server/Client

1. Observer Pattern

- **Dove si applica:** Interazione tra il client e il server per l'avvio della GUI
- **Motivazione:** per notificare l'avvio della GUI sul client in risposta a un evento specifico ("START_GUI") inviato dal client al server. Questo pattern è utile quando un oggetto (il server) deve notificare un altro oggetto (il client) riguardo a un cambiamento di stato
- **Codice rilevante**:



```
// Se il messaggio è quello specifico per avviare la GUI
if ("START_GUI".equalsIgnoreCase(request)) {
    // Avvia la GUI nel thread dell'interfaccia utente
    SwingUtilities.invokeLater(() -> {
        new PaginaIniziale().setVisible(b:true);
    });
}
```

2. Producer-Consumer Pattern

- **Dove si applica:** comunicazione tra client e server
- **Motivazione:** si applica qui in quanto il client funge da "produttore" di richieste e il server funge da "consumatore". Questo pattern gestisce il flusso di dati tra il client e il server, permettendo al server di accettare e gestire più richieste simultaneamente

```
while (true) {
    try {
        Socket clientSocket = serverSocket.accept();
        new Thread(new ClientHandler(clientSocket, databaseManager)).start();
    } catch (IOException e) {
        JOptionPane.showMessageDialog(parentComponent:null, "Errore nell'accettazione della connessione del client: " + e.getMessage(),
    }
}
```

3. Strategy Pattern

- **Dove si applica:** gestione delle risposte alle richieste dei client

- **Motivazione:** per gestire diversi tipi di richieste che il server può ricevere dal client. Ogni tipo di richiesta può essere trattato con una strategia diversa, incapsulata in una classe separata
- **Codice rilevante**:

```
// Leggi il messaggio dal client
String request = in.readLine();
JOptionPane.showMessageDialog(parentComponent:null, "Received message from client: " + request);

// Rispondi al client
out.println("Risposta a: " + request);
```

4. Singleton Pattern

- **Dove si applica:** gestione della connessione al database
- **Motivazione:** si applica al DatabaseManager per garantire che esista una sola istanza della connessione al database in tutto il sistema, prevenendo problemi di concorrenza e fornendo un punto di accesso globale alla connessione
- **Codice rilevante**:

  // Codice nel ServerCM
  databaseManager = new DatabaseManager(dbUrl, dbUsername, dbPassword);