

**An-Najah Nation University**

**Faculty of Engineering and**

**Information Technology**

# جامعة النجاح الوطنیة

**كلیة الھندسة وتكنولوجیا المعلومات**

**Computer Engineering Department**

**Networks1 (10636454)**

**Programming HW project**

### **. Introduction**

This project aims to develop a peer-to-peer (P2P) chat application using a combination of TCP and UDP protocols in Java, featuring an integrated graphical user interface (GUI). The application includes a login system, user management, file transfer, status tracking (Active/Busy/Away), and maintains core communication via UDP.

### **. System Design**

#### **Functional Requirements**

* **Login System:**
  + User authentication via TCP with a central server.
  + Credentials stored in a file (user\_credentials.txt).
* **User Management:**
  + Add/remove users via the server interface.
  + Display a list of active users with color-coded statuses.
* **Status System:**
  + Manual status change (Active/Busy/Away).
  + Automatic switch to "Away" after 30 seconds of inactivity.
* **File Transfer:**
  + Direct file transfer between peers via a dedicated port.
  + Display transfer statistics (size, speed, time).
* **Core Chat:**
  + Send messages via UDP with timestamps and distinct colors.

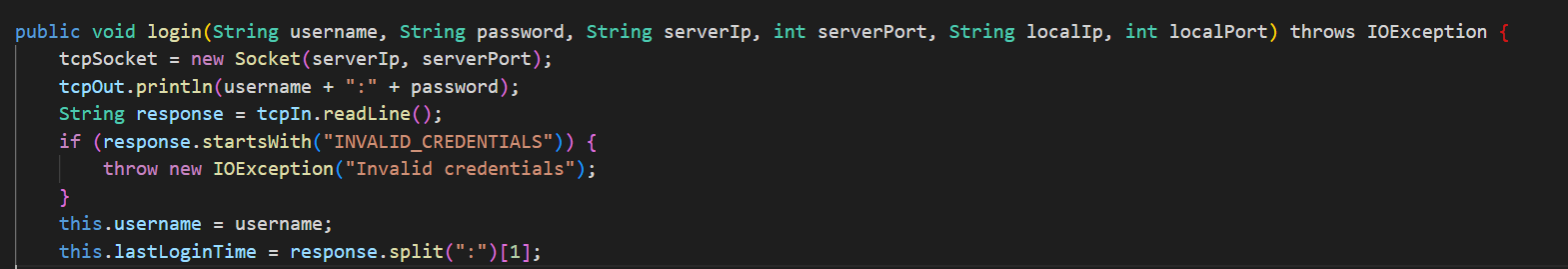
#### **Architecture Overview**

* **Server (Server.java):**
  + Manages TCP connections for authentication and user list updates.
  + Includes a GUI for user management and log viewing.
* **Client (Client.java and ClientGUI.java):**
  + Connects to the server via TCP to retrieve the user list.
  + Establishes UDP connections for direct peer-to-peer chat.
  + Supports file transfer via a separate port.

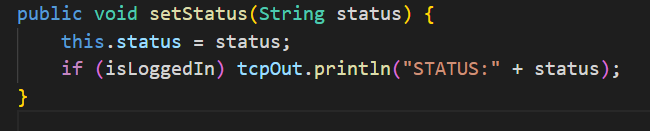
### **. Implementation**

#### **Key Code Components**

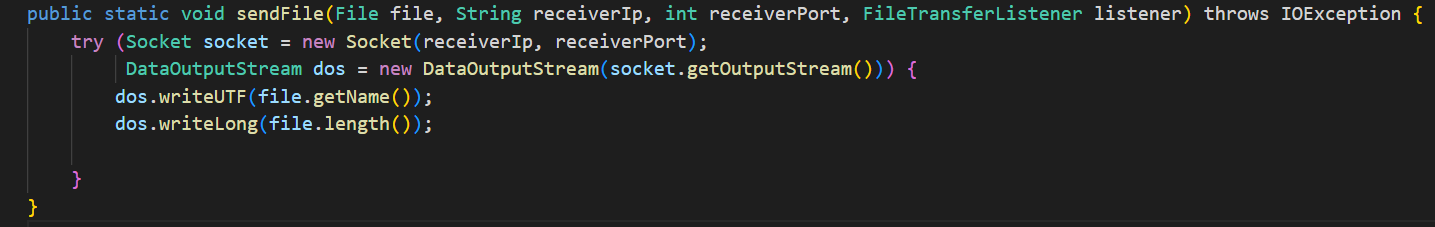
1. **User Authentication (Client.java):**



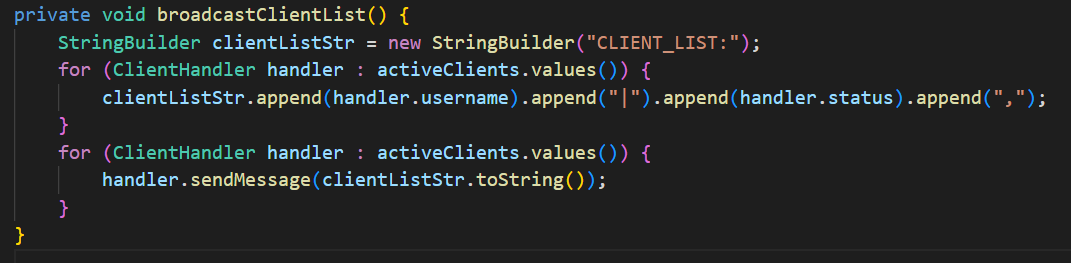
1. **Status Management (Client.java):**



1. **File Transfer (FileTransferManager.java):**



1. **User List Updates (Server.java):**



### 

### **. Testing**

* **Login Testing:**
  + Verified credential validation using user\_credentials.txt.
  + Displayed last successful login time in the client interface.
* **File Transfer Testing:**
  + Transferred files of varying sizes with statistics (e.g., 5MB file ≈ 1.2MB/s speed).
* **Auto-Status Testing:**
  + Status changed to "Away" after 30 seconds of inactivity, reverted to "Active" upon interaction.

### **. Challenges and Solutions**

* **Thread Synchronization:**
  + Used SwingUtilities.invokeLater() to update the GUI from sub-threads.
* **Port Management:**
  + Assigned port +1 for file transfers (e.g., 8888 for chat, 8889 for files).
* **Message Formatting:**
  + Colored messages in the GUI by type (sent/received) using DefaultListCellRenderer.

### **. Improvements**

* **Security Enhancements:**
  + Encrypt passwords in user\_credentials.txt.
* **Group Chat Support:**
  + Add group chat rooms via UDP multicast.
* **File Transfer Optimization:**
  + Support resuming interrupted file transfers.

### **. Conclusion**

A fully functional chat application was implemented, supporting direct communication (UDP) and centralized user management (TCP), with advanced features like file transfer and status tracking. The application is scalable for additional features such as encryption and voice chat.

**Attachments:**

1. Screenshots of Interfaces:
   * Server interface with user list.
   * Client interface with file transfer and chat.
2. [Demo Video Link] (attached with submission).

**Note:** The program was tested on multiple devices (not just localhost) as required by the assignment.

