**📡 Java RMI-Based Chat Application**

**🔖 Project Title:**

**Simple Chat Application using Java RMI**

**📘 Abstract**

This project implements a basic text-based chat system using **Java Remote Method Invocation (RMI)**. The system allows multiple clients to connect to a central server and exchange real-time messages. It demonstrates the use of distributed programming principles and Java RMI interfaces for remote communication between Java applications.

**🛠️ Technologies Used**

* **Java SE 8+**
* **Java RMI API**
* **Command Line Interface (CLI)**

**📁 File Structure**

| **File Name** | **Description** |
| --- | --- |
| ChatServer.java | RMI interface defining server-side methods for chat handling. |
| ChatServerImpl.java | Implementation of ChatServer, manages client list and messages. |
| ChatClient.java | RMI interface for client-side callbacks (to receive messages). |
| Server.java | Launches the RMI registry and binds ChatServerImpl. |
| Client.java | Connects to the server, sends messages, and receives callbacks. |

**System Architecture**

**➤ Architecture Type:**

**Client–Server model using RMI**

**➤ Flow:**

1. Server.java binds the ChatServer interface (ChatServerImpl.java) to RMI Registry.
2. Client.java connects to the server using RMI.
3. Clients register themselves for receiving messages.
4. When one client sends a message, the server distributes it to all registered clients using callback methods in ChatClient.

**🔌 Functional Overview**

**✅ Features Implemented:**

* Connect multiple clients to a single server.
* Broadcast messages from one client to all others.
* Basic command-line input and output.
* Live, real-time message delivery via RMI callbacks.

**🧪 How to Run**

**1. Compile the files**

javac ChatServer.java ChatClient.java Server.java Client.java

**2. Start the RMI Registry**

start rmiregistry

*Make sure rmiregistry is run in the same directory as your .class files.*

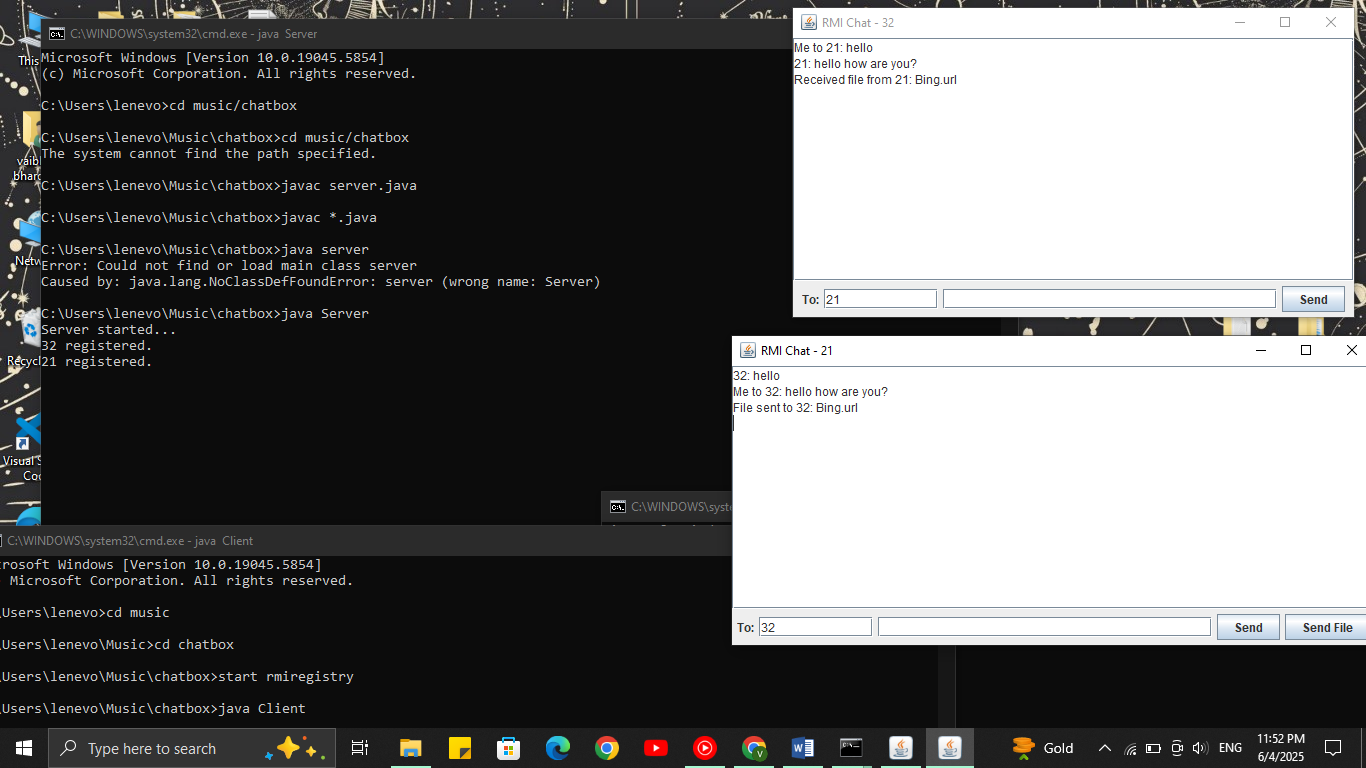
**3. Run the Server**

java Server

**4. Run one or more Clients**

java Client

**📸 Sample Output**



**📋 Limitations**

* No encryption or authentication.
* Not suitable for internet deployment (LAN only, unless secured).
* No GUI (uses CLI only).
* No message history or storage.

**🌟 Future Enhancements (Optional for Internship Growth)**

* GUI using JavaFX or Swing.
* SSL encryption.
* Private messaging (user-to-user).
* File transfer support.
* Chat history with SQLite or file logging.

**👨‍💻 Author**

**Vaibhav Bhardwaj**  
B.Tech CSE Student  
Focus Area: Java-based Systems and Networking

**📂 Sample GitHub Repository Structure**

chatbox/

├── ChatClient.java

├── ChatServer.java

├── ChatServerImpl.java

├── Client.java

├── Server.java

├── chatbox.txt

└── screenshot