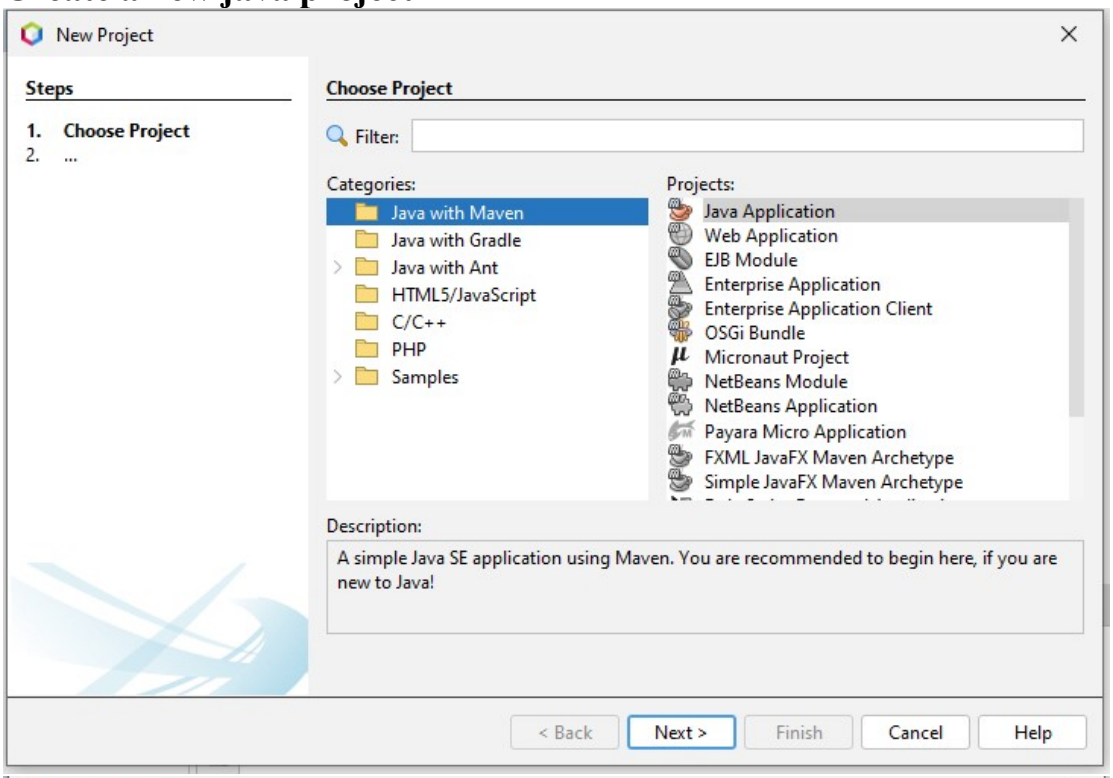
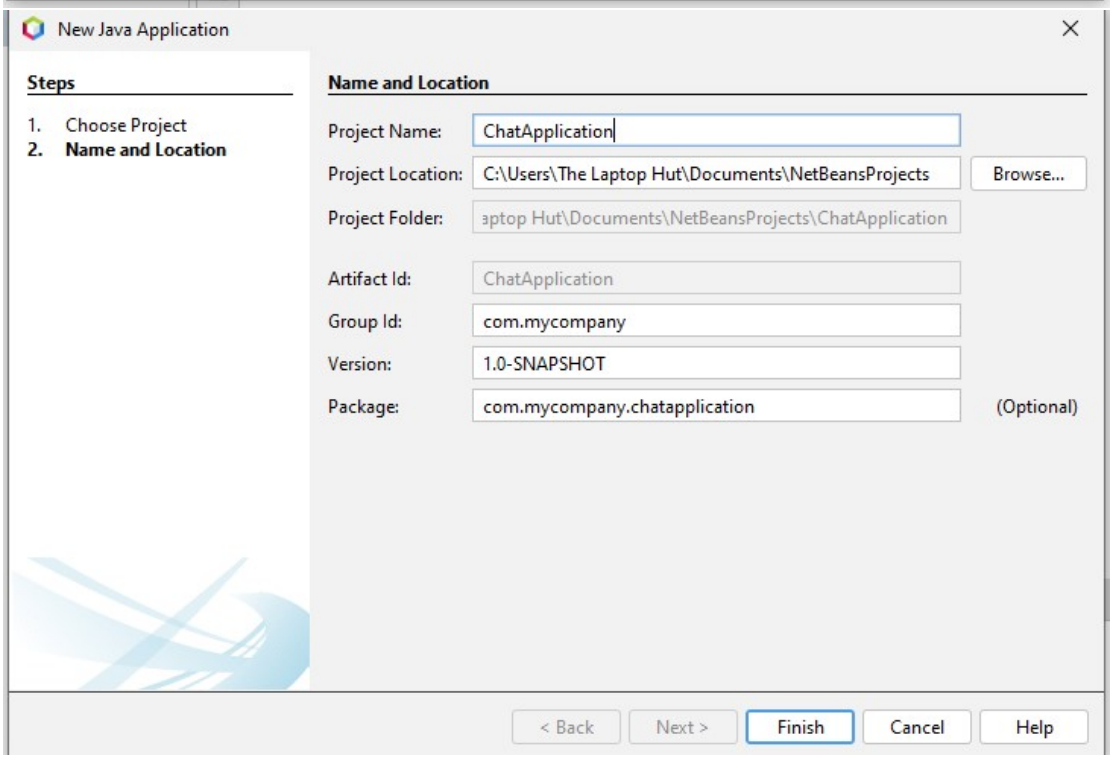
TASK 2

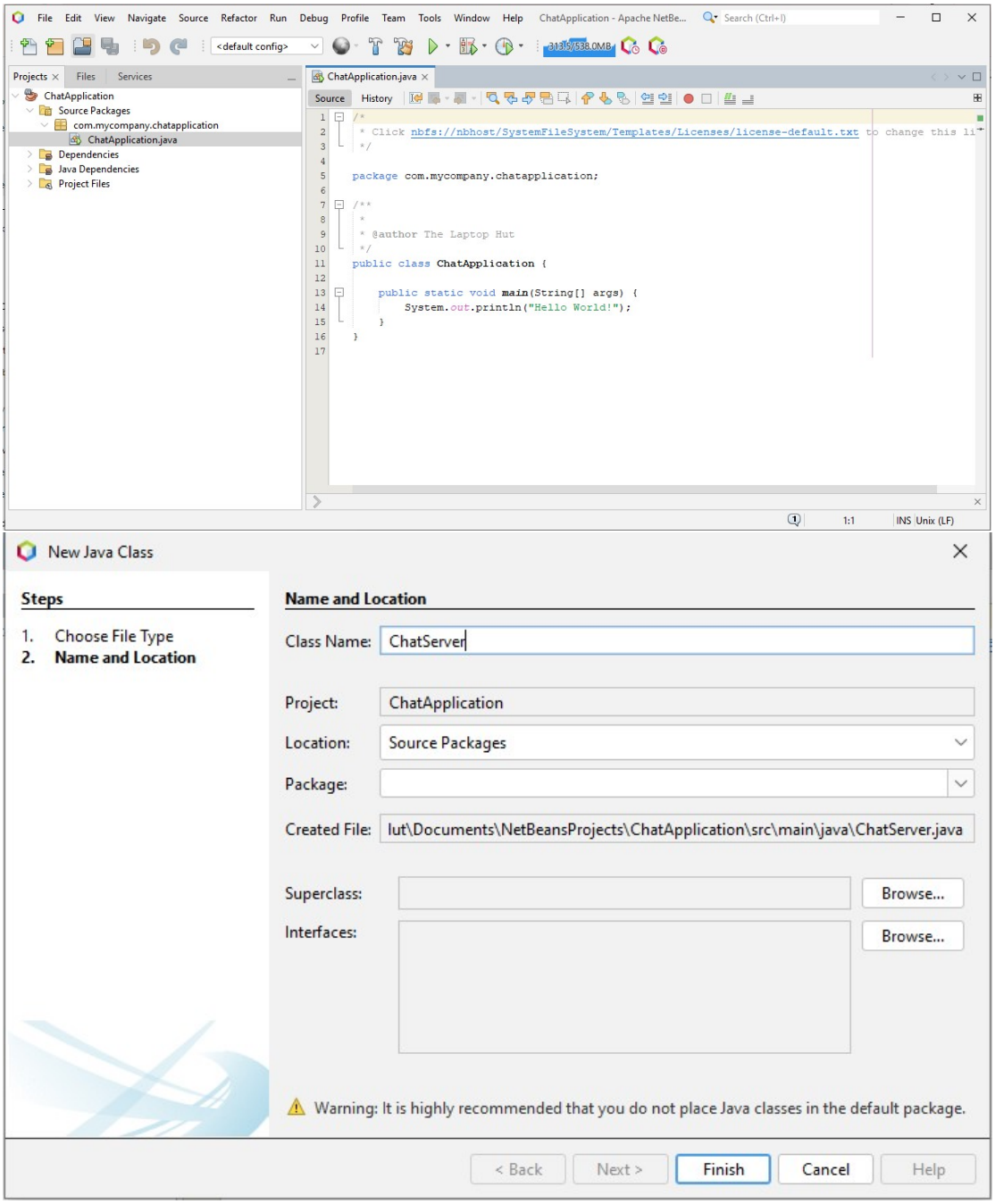
Develop a multithreaded chat application where multiple clients can connect to a server and exchange messages in real-time.

Steps of Solution

Create a new java project







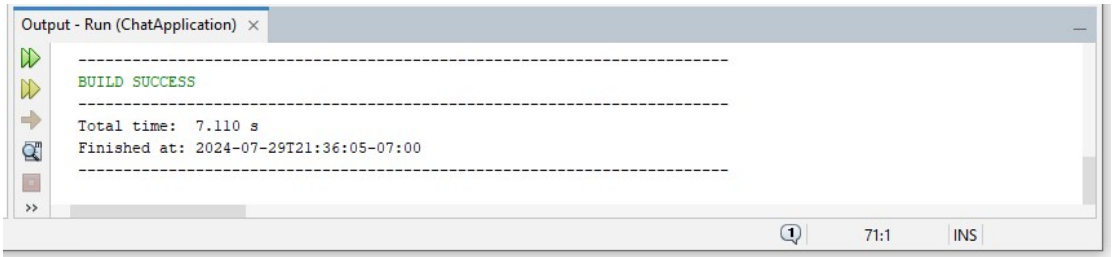
Server Code

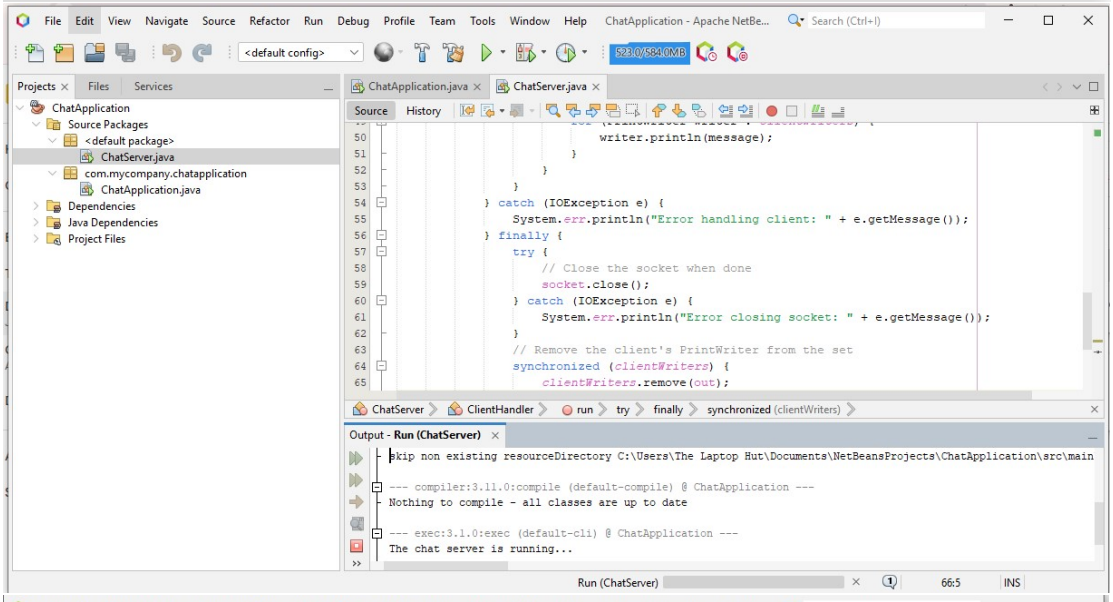
```
import java.io.*;
import java.net.*;
import java.util.*;

public class ChatServer {
    // Set to hold client PrintWriter objects
    private static final Set<PrintWriter> clientWriters = new HashSet<>();

    public static void main(String[] args) {
        System.out.println("The chat server is running...");
        int port = 12345; // Choose your port number
```

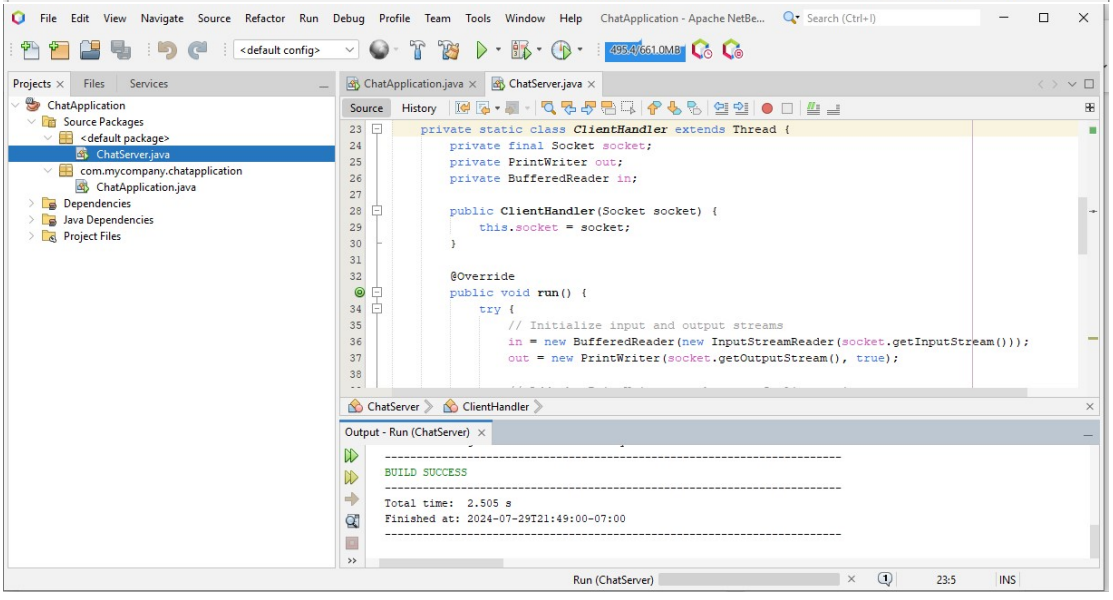
```
    try (ServerSocket serverSocket = new ServerSocket(port)) {
        while (true) {
            // Accept new client connections and handle them in a new thread
            new ClientHandler(serverSocket.accept()).start();
        }
    } catch (IOException e) {
        System.err.println("Error starting server: " + e.getMessage());
    }
}

private static class ClientHandler extends Thread {
    private final Socket socket;
    private PrintWriter out;
    private BufferedReader in;

    public ClientHandler(Socket socket) {
        this.socket = socket;
    }

    @Override
    public void run() {
        try {
            // Initialize input and output streams
            in = new BufferedReader(new InputStreamReader(socket.getInputStream()));
            out = new PrintWriter(socket.getOutputStream(), true);

            // Add the PrintWriter to the set of client writers
            synchronized (clientWriters) {
                clientWriters.add(out);
            }

            String message;
            while ((message = in.readLine()) != null) {
                System.out.println("Received: " + message);
                // Broadcast the message to all clients
                synchronized (clientWriters) {
                    for (PrintWriter writer : clientWriters) {
                        writer.println(message);
                    }
                }
            }
        } catch (IOException e) {
            System.err.println("Error handling client: " + e.getMessage());
        } finally {
            try {
                // Close the socket when done
                socket.close();
            } catch (IOException e) {
                System.err.println("Error closing socket: " + e.getMessage());
            }
            // Remove the client's PrintWriter from the set
            synchronized (clientWriters) {
                clientWriters.remove(out);
            }
        }
```

```
        }
    }
}
```

Run the Server code



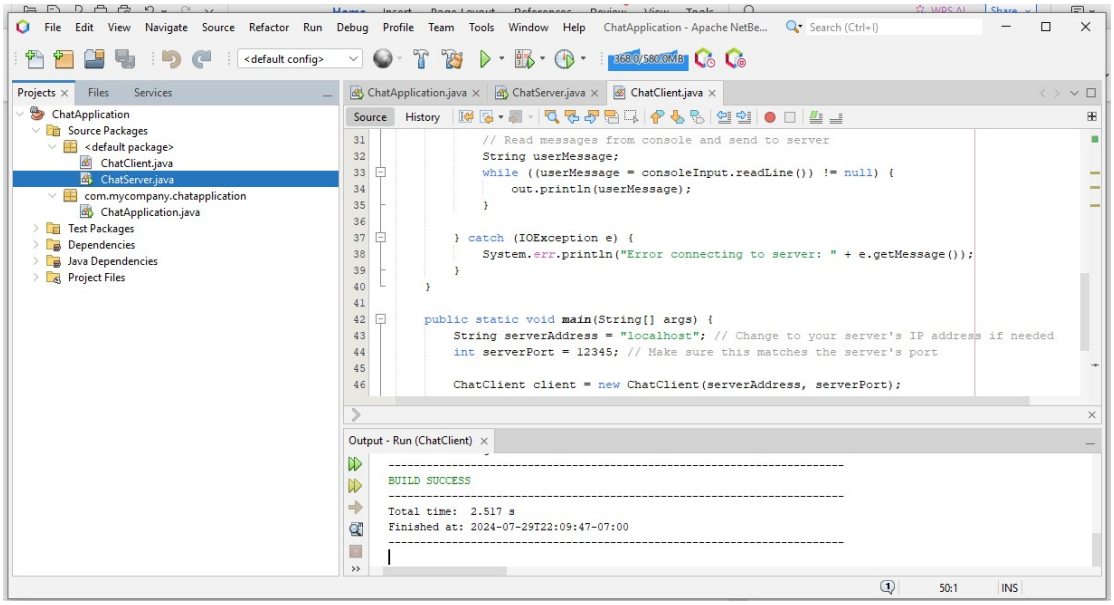




Chat Client 1

```
import java.io.*;
import java.net.*;

public class ChatClient {
    private final String serverAddress;
    private final int serverPort;

    public ChatClient(String address, int port) {
        this.serverAddress = address;
        this.serverPort = port;
    }

    public void start() {
        try (Socket socket = new Socket(serverAddress, serverPort);
             BufferedReader in = new BufferedReader(new InputStreamReader(socket.getInputStream()));
             PrintWriter out = new PrintWriter(socket.getOutputStream(), true);
             BufferedReader consoleInput = new BufferedReader(new InputStreamReader(System.in))) {

            // Thread to read messages from server
            new Thread(() -> {
                try {
                    String serverMessage;
                    while ((serverMessage = in.readLine()) != null) {
                        System.out.println("Server: " + serverMessage);
                    }
                } catch (IOException e) {
                    System.err.println("Error reading from server: " + e.getMessage());
                }
            }).start();

            // Read messages from console and send to server
            String userMessage;
            while ((userMessage = consoleInput.readLine()) != null) {
                out.println(userMessage);
            }

        } catch (IOException e) {
            System.err.println("Error connecting to server: " + e.getMessage());
        }
    }

    public static void main(String[] args) {
        String serverAddress = "localhost"; // Change to your server's IP address if needed
        int serverPort = 12345; // Make sure this matches the server's port

        ChatClient client = new ChatClient(serverAddress, serverPort);
        client.start();
    }
}
```

Run ChatClient

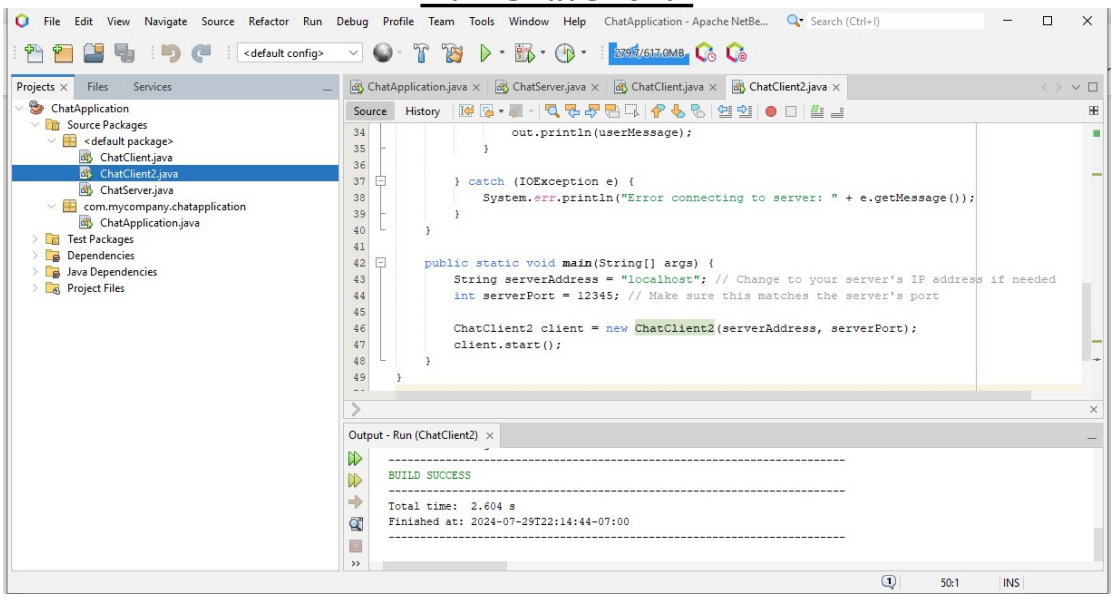


Chat Client 2

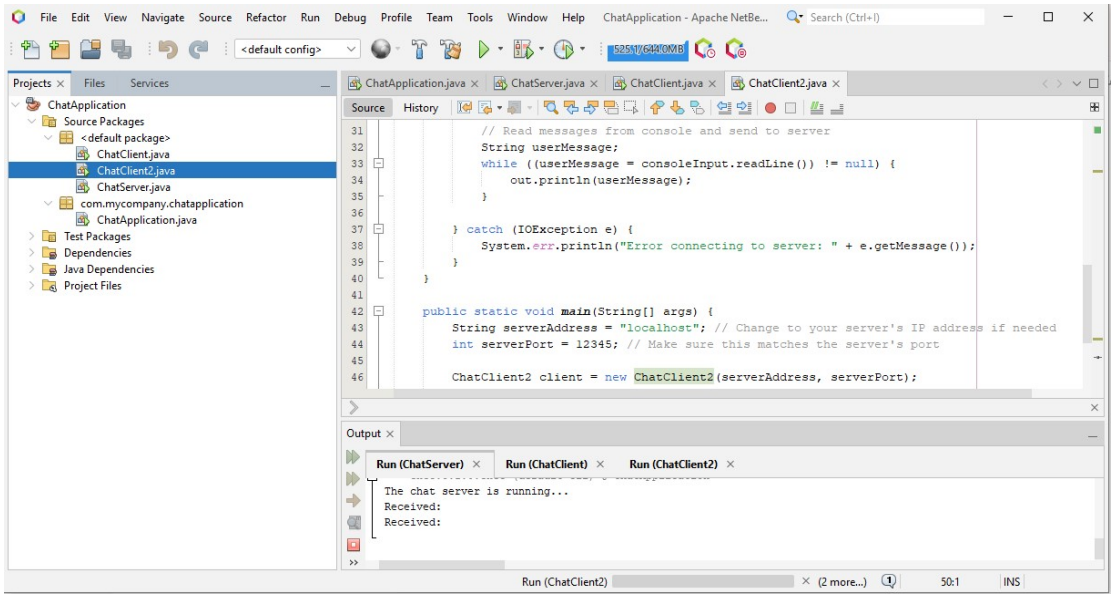
```
import java.io.*;
import java.net.*;

public class ChatClient2 {
    private final String serverAddress;
    private final int serverPort;

    public ChatClient2(String address, int port) {
        this.serverAddress = address;
        this.serverPort = port;
    }

    public void start() {
        try (Socket socket = new Socket(serverAddress, serverPort);
             BufferedReader in = new BufferedReader(new InputStreamReader(socket.getInputStream()));
             PrintWriter out = new PrintWriter(socket.getOutputStream(), true);
             BufferedReader consoleInput = new BufferedReader(new InputStreamReader(System.in))) {

            // Thread to read messages from server
            new Thread(() -> {
                try {
                    String serverMessage;
                    while ((serverMessage = in.readLine()) != null) {
                        System.out.println("Server: " + serverMessage);
                    }
                } catch (IOException e) {
                    System.err.println("Error reading from server: " + e.getMessage());
                }
            }).start();

            // Read messages from console and send to server
            String userMessage;
            while ((userMessage = consoleInput.readLine()) != null) {
                out.println(userMessage);
            }

        } catch (IOException e) {
            System.err.println("Error connecting to server: " + e.getMessage());
```

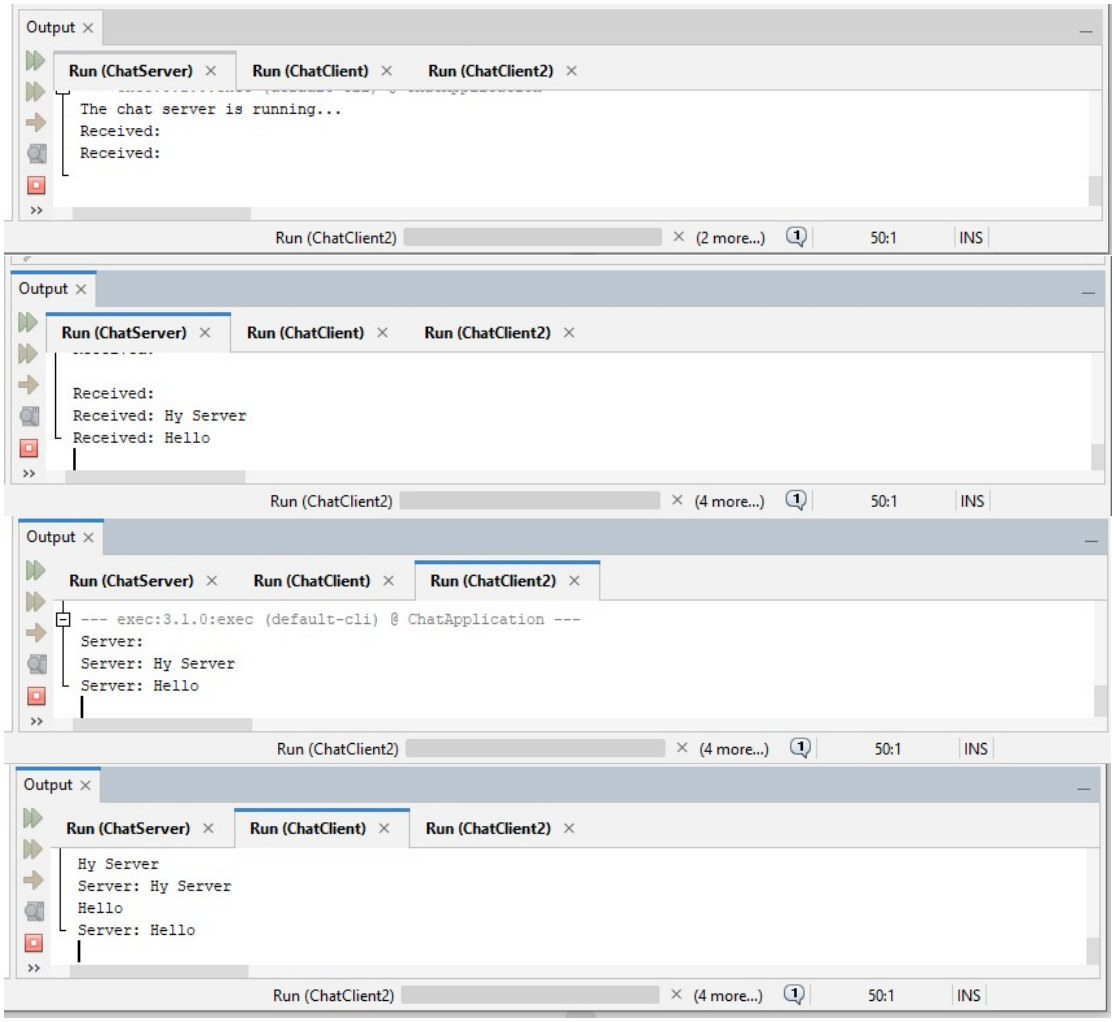
```
        }
    }

    public static void main(String[] args) {
        String serverAddress = "localhost"; // Change to your server's IP address if needed
        int serverPort = 12345; // Make sure this matches the server's port

        ChatClient2 client = new ChatClient2(serverAddress, serverPort);
        client.start();
    }
}
```

Run ChatClient2



Successful Run both Clients and Server




Output
Run (ChatServer)
Run (ChatClient)
Run (ChatClient2)
The chat server is running...
Received:
Received:
Run (ChatClient2)
(2 more...)
50:1
INS
Output
Run (ChatServer)
Run (ChatClient)
Run (ChatClient2)
Received:
Received: Hy Server
Received: Hello
Run (ChatClient2)
(4 more...)
50:1
INS
Output
Run (ChatServer)
Run (ChatClient)
Run (ChatClient2)
--- exec:3.1.0:exec (default-cli) @ ChatApplication ---
Server:
Server: Hy Server
Server: Hello
Run (ChatClient2)
(4 more...)
50:1
INS
Output
Run (ChatServer)
Run (ChatClient)
Run (ChatClient2)
Hy Server
Server: Hy Server
Hello
Server: Hello
Run (ChatClient2)
(4 more...)
50:1
INS

<><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><>