

# Lecture Manager (Java Project)

## UI

- 계획
  - 1. 로그인 및 회원 가입 UI

Lecture Manager

학번 :

비밀 번호 :

회원가입

로그인

20171687 이상민  
로그인이 완료되었습니다.

로그인을 실패했습니다.

학번 :

이름 :





비밀 번호 :

확인

취소

- 2. 학생 UI

Student : 000

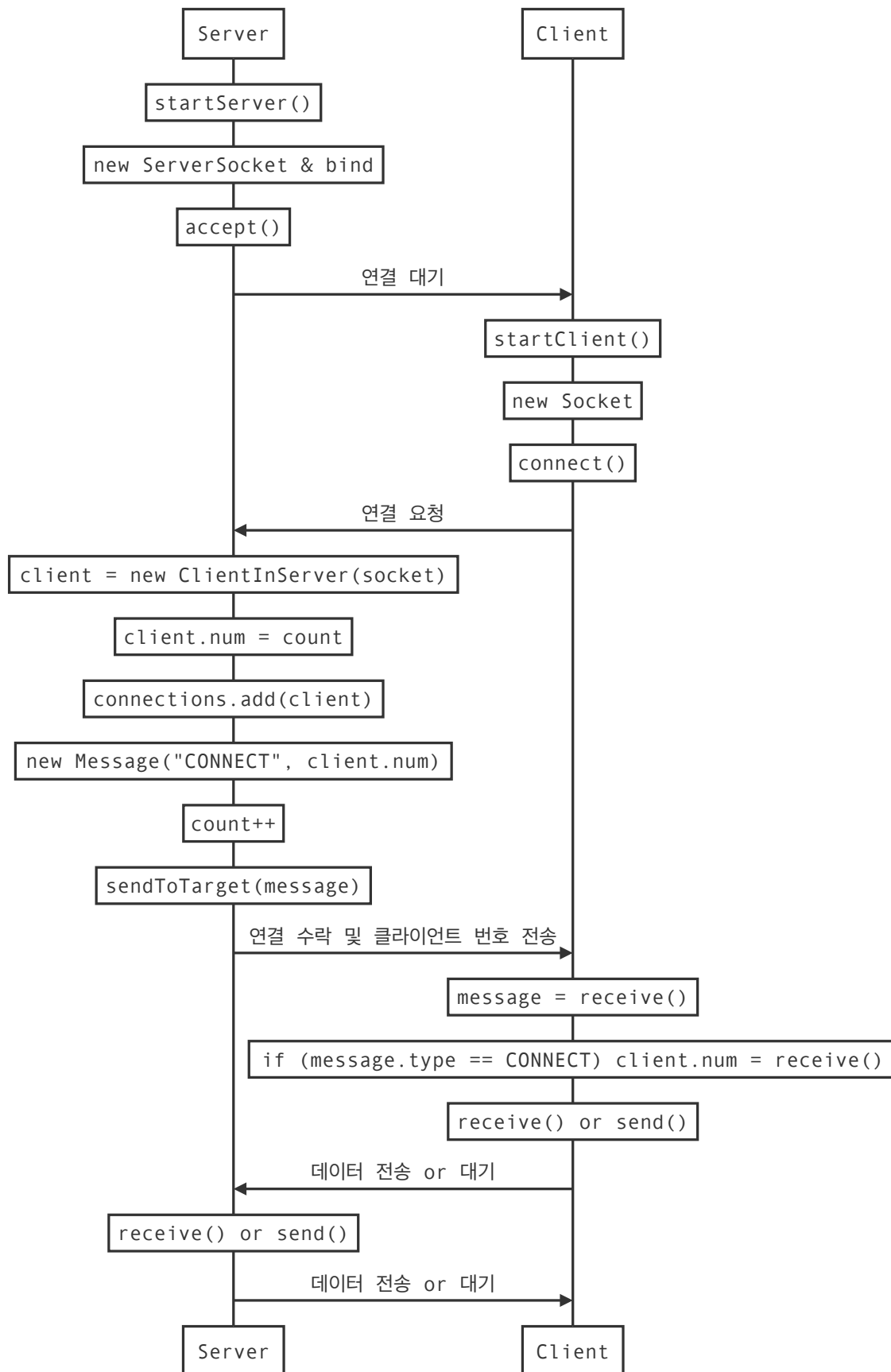
새로 고침	문제 내용	Code
실습 문제 1번 		
실습 문제 2번 		
실습 문제 3번 		
실습 문제 4번 		
<div>Run</div> <div>Run 제출</div>		
문제 개수 : 1 / 4		

### 3. 교수님 UI

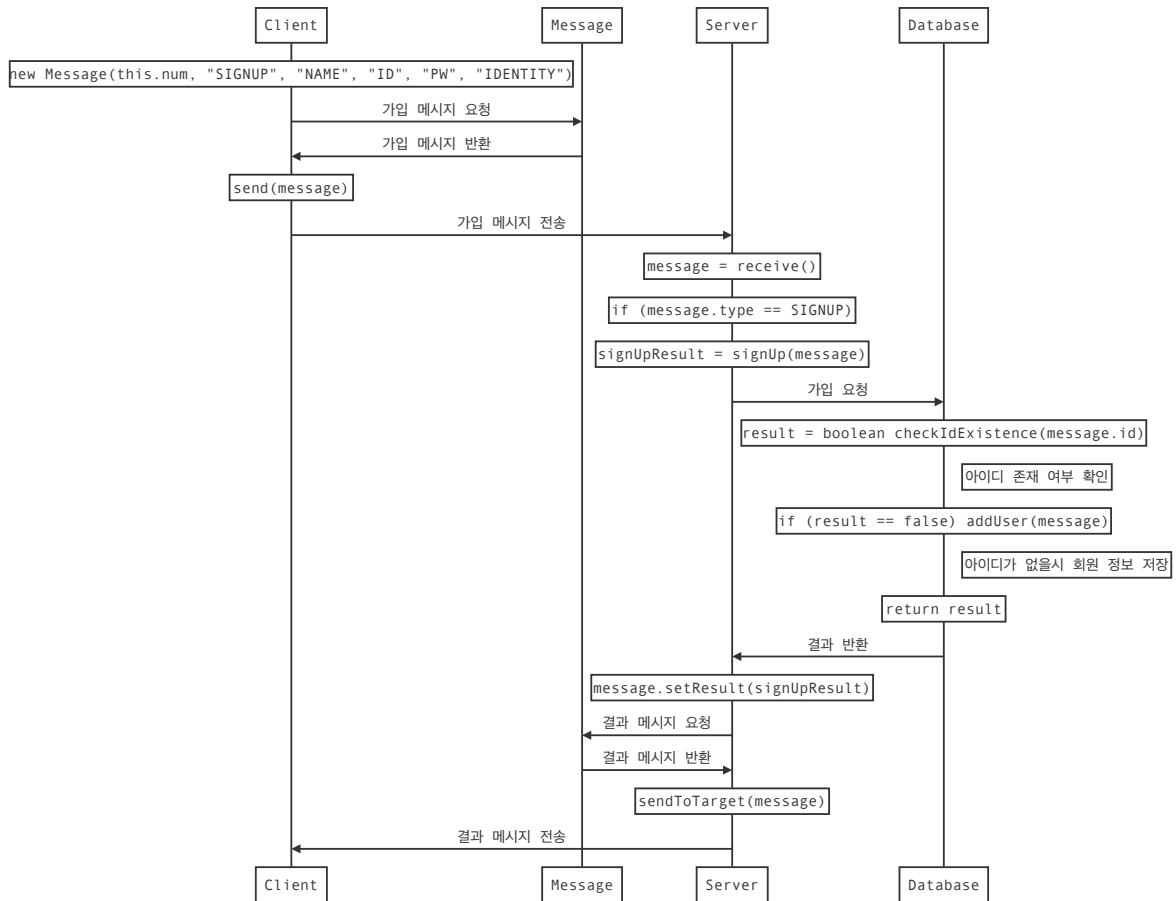
Professor : 000

제출 확인 ▼	실습1	실습2	실습3
20170000 구아람 20170000 이상민 ... ...	<pre> public class Example {     public static void main(String[] args) {         Scanner scanner = new Scanner(System.in);         System.out.print("n : ");         int number = scanner.nextInt();         function(number);     }     public static void function(int n) {         if(n==0) return;         else unction(n-1);         System.out.println(n);     } } </pre>		
<div>실행결과</div> <div>Run</div> <div>           n : 3            1            2            3         </div> <div>거부 승인</div>			

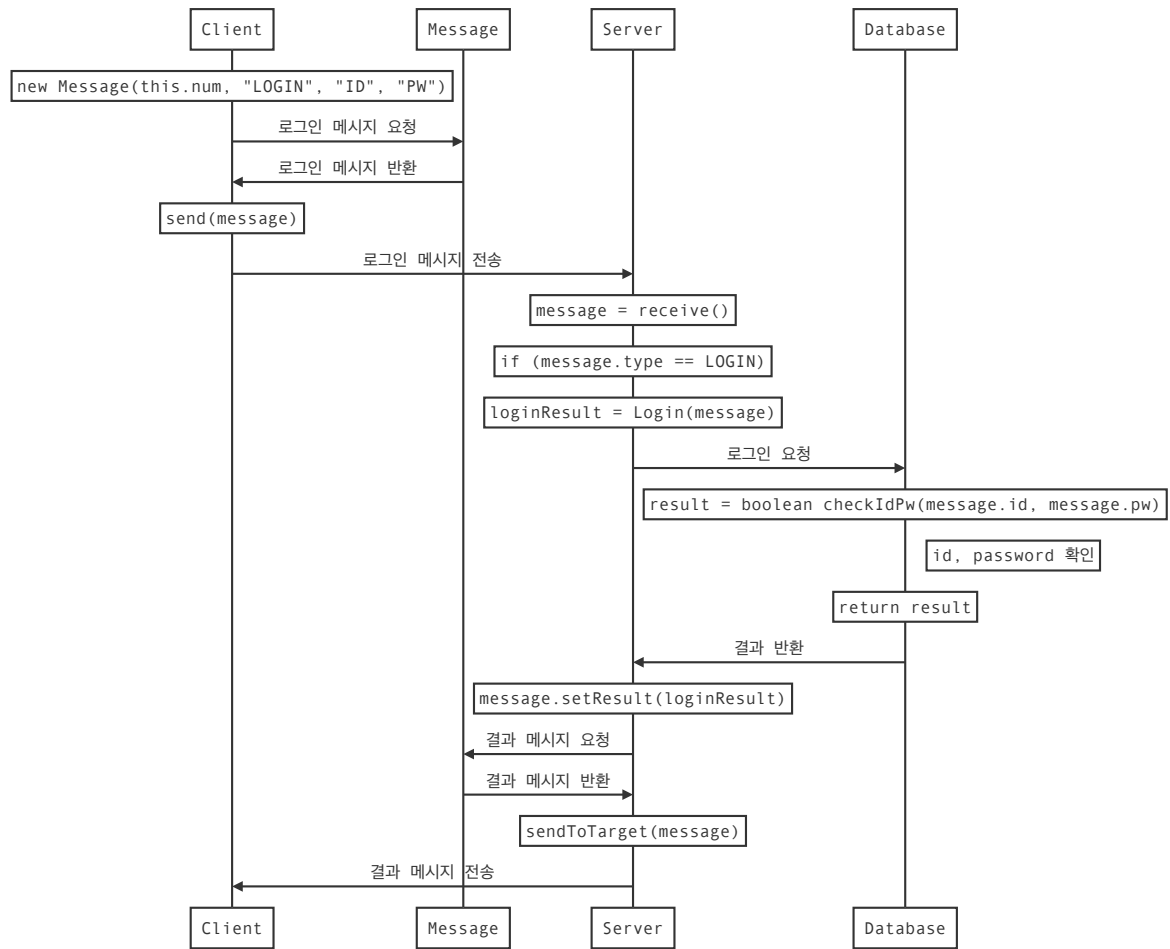




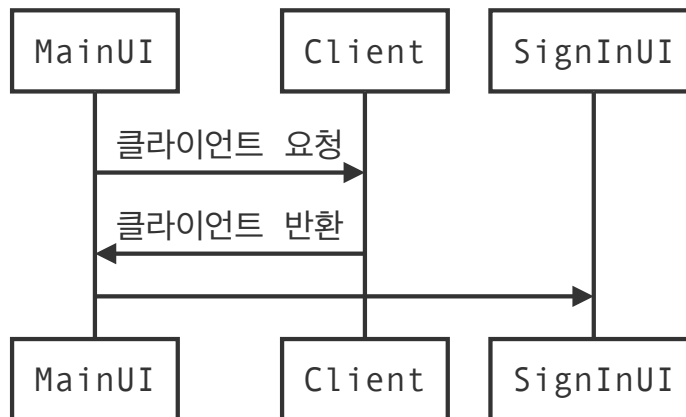
## 회원 가입



## 로그인



## UI



## 소켓 통신 예제

# Server(서버)

```
package chat_server_implement;

import com.sun.security.ntlm.Client;
import sun.lwawt.PlatformEventNotifier;

import javax.swing.*;
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.io.FileInputStream;
import java.io.IOException;
import java.io.InputStream;
import java.io.OutputStream;
import java.lang.reflect.InvocationTargetException;
import java.net.InetSocketAddress;
import java.net.ServerSocket;
import java.net.Socket;
import java.nio.charset.StandardCharsets;
import java.util.Iterator;
import java.util.List;
import java.util.Vector;
import java.util.concurrent.ExecutorService;
import java.util.concurrent.Executors;
import java.util.stream.Stream;

public class ServerExample {

    ExecutorService executorService;
    ServerSocket serverSocket;
    List<Client> connections = new Vector<>();

    JPanel mainPanel = new JPanel();
    static JTextArea jTextArea = new JTextArea();

    void startServer() {
        // ExecutorService 객체 호출
        executorService = Executors.newFixedThreadPool(
            // 사용가능한 CPU 코어 수 호출
            Runtime.getRuntime().availableProcessors()
        );

        try {
            // ServerSocket 객체 생성
            serverSocket = new ServerSocket();
            // ServerSocket 을 로컬로 IP를 잡고 5001 포트와 바인딩한다.
            serverSocket.bind(new InetSocketAddress("localhost", 5001));
        }
    }
}
```

```

    } catch (Exception e) {
        // 예외가 발생할 경우 서버를 닫고 메소드를 종료한다.
        if (!serverSocket.isClosed()) {
            stopServer();
            return;
        }
    }

    // 수락 작업 생성
    Runnable runnable = new Runnable() {
        @Override
        public void run() {
            System.out.println("[서버 시작]");
            jTextArea.append("[서버 시작]\n");

            while (true) {
                try {
                    // 연결 수락
                    Socket socket = serverSocket.accept();
                    String message = "[연결 수락: " +
                        socket.getRemoteSocketAddress() +
                        ": " + Thread.currentThread().getName() +
                        "]";
                    System.out.println(message);
                    jTextArea.append(message + "\n");

                    // Client 객체 저장
                    Client client = new Client(socket);
                    connections.add(client);

                    jTextArea.append("[연결 개수: " + connections.size() +
                        "]" + "\n");
                } catch (Exception e) {
                    if (!serverSocket.isClosed()) {
                        stopServer();
                        break;
                    }
                }
            }
        }
    };

    // 스레드풀에서 처리
    executorService.submit(runnable);
}

void stopServer() {
    try {
        // (원래 방법) 모든 Socket 닫기
        // Iterator<Client> iterator = connections.iterator();
    }
}

```



```

//         while (iterator.hasNext()) {
//             Client client = iterator.next();
//             client.socket.close();
//             iterator.remove();
//         }

// (스트림 이용 방법) 모든 Socket 닫기
connections.forEach(client -> {
    try {
        client.socket.close();
    } catch (IOException e) {
        e.printStackTrace();
    }
});

// ServerSocket 닫기
if (serverSocket != null && !serverSocket.isClosed()) {
    serverSocket.close();
}

// ExecutorService 종료
if (executorService != null && !executorService.isShutdown()) {
    executorService.isShutdown();
}

System.out.println("[서버 멈춤]");
jTextArea.append("[서버 멈춤]\n");

} catch (Exception e) {
}
}

// Client 를 내부 클래스로 선언
class Client {
    Socket socket;

    // 매개값으로 socket 을 받는 생성자
    Client(Socket socket) {
        this.socket = socket;
        receive();
    }

    void receive() {
        // 데이터 받기 작업 생성
        Runnable runnable = new Runnable() {
            @Override
            public void run() {
                try {
                    while (true) {

```

데이터 받기

```
byte[] byteArr = new byte[100];
InputStream inputStream = socket.getInputStream();

// 클라이언트가 비정상 종료를 했을 경우 IOException 발생
int readByteCount = inputStream.read(byteArr); //

// 클라이언트가 정상적으로 Socket 의 close() 를 호출했을 경우
if (readByteCount == -1) {
    throw new IOException();
}

String message = "[요청 처리: " +
socket.getRemoteSocketAddress() + ": " +
    Thread.currentThread().getName() + "];
System.out.println(message);
jTextArea.append(message + "\n");

// 문자열로 변환
String data = new String(byteArr, 0, readByteCount,
StandardCharsets.UTF_8);

// 모든 클라이언트에게 보냄 (선택적으로도 보낼 수 있다)
for (Client client : connections) {
    client.send(data);
}
} catch (Exception e) {
    try {
        connections.remove(Client.this);

        String message = "[클라이언트 통신 안됨: " +
            socket.getRemoteSocketAddress() +
            ": " + Thread.currentThread().getName() +
            "];

        System.out.println(message);
        jTextArea.append(message + "\n");

        socket.close();
    } catch (IOException e2) {
    }
}

}

};

// 스레드풀에서 처리
executorService.submit(runnable);
}
```

```

void send(String data) {
    // 데이터 보내기 작업 생성
    Runnable runnable = new Runnable() {
        @Override
        public void run() {
            // 클라이언트로 데이터 보내기
            try {
                byte[] byteArr = data.getBytes(StandardCharsets.UTF_8);
                OutputStream outputStream = socket.getOutputStream();
                outputStream.write(byteArr);
                outputStream.flush();
            } catch (Exception e) {
                try {
                    String message = "[클라이언트 통신 안됨: " +
                        socket.getRemoteSocketAddress() + ": " +
                        Thread.currentThread().getName() + "]\n";
                    System.out.println(message);
                    jTextArea.append(message + "\n");

                    connections.remove(Client.this);
                    socket.close();
                } catch (IOException e2) {
                }
            }
        }
    };
    // 스레드풀에서 처리
    executorService.submit(runnable);
}

void start() {
    ServerExample server = new ServerExample();

    JFrame jFrame = new JFrame("Server");

    mainPanel.setLayout(new BorderLayout());
    JButton jButton = new JButton("START");

    jTextArea.setEditable(false);

    jButton.addActionListener(new ActionListener() {
        @Override
        public void actionPerformed(ActionEvent e) {
            if (jButton.getLabel().equals("START")) {
                server.startServer();
                jButton.setLabel("STOP");
            } else {

```

```

        server.stopServer();
        jButton.setLabel("START");
    }
}

});

mainPanel.add(jTextArea, BorderLayout.CENTER);
mainPanel.add(jButton, BorderLayout.SOUTH);

jFrame.add(mainPanel);

jFrame.setSize(500, 300);
jFrame.setVisible(true);
}

public static void main(String[] args) {
    ServerExample serverExample = new ServerExample();
    serverExample.start();
}
}

```

## Client(클라이언트)

```

package chat_server_implement;

import javax.swing.*;
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.io.IOException;
import java.io.InputStream;
import java.io.OutputStream;
import java.net.InetSocketAddress;
import java.net.Socket;
import java.nio.charset.StandardCharsets;

public class ClientExample {

    Socket socket;
    JPanel mainPanel = new JPanel();
    static JTextArea jTextArea = new JTextArea();

    void startClient() {
        // 스레드 생성
        Thread thread = new Thread() {
            @Override

```

```

        public void run() {
            try {
                // 소켓 생성 및 연결 요청
                socket = new Socket();
                socket.connect(new InetSocketAddress("localhost", 5001));

                String message = "[연결 완료: " +
socket.getRemoteSocketAddress() + "]";
                System.out.println(message);

                JTextArea.append(message + "\n");
            } catch (Exception e) {
                String message = "[서버 통신 안됨]";
                System.out.println(message);

                JTextArea.append(message + "\n");
                if (!socket.isClosed()) {
                    stopClient();
                }
                return;
            }
            // 서버에서 보낸 데이터 받기
            receive();
        }
    };
    // 스레드 시작
    thread.start();
}

void stopClient() {
    try {
        String message = "[연결 끊음]";
        System.out.println(message);

        JTextArea.append(message + "\n");

        // 연결 끊기
        if (socket != null && !socket.isClosed()) {
            socket.close();
        }

    } catch (IOException e) {

    }
}

void receive() {
    while (true) {
        try {

```

터 받기

```
byte[] byteArr = new byte[100];
InputStream inputStream = socket.getInputStream();

// 서버가 비정상적으로 종료했을 경우 IOException 발생
int readByteCount = inputStream.read(byteArr); // 데이터 받기

// 서버가 정상적으로 Socket 의 close() 를 호출했을 경우
if (readByteCount == -1) {
    throw new IOException();
}

// 문자열로 변환
String data = new String (byteArr, 0, readByteCount,
StandardCharsets.UTF_8);

String message = "[받기 완료] " + data;

System.out.println(message);
jTextArea.append(message + "\n");

} catch (Exception e) {
    String message = "[서버 통신 안됨]";
    System.out.println(message);
    jTextArea.append(message + "\n");

    stopClient();
    break;
}
}

void send(String data) {
    // 스레드 생성
    Thread thread = new Thread() {
        @Override
        public void run() {
            try {
                byte[] byteArr = data.getBytes(StandardCharsets.UTF_8);

                // 서버로 데이터 보내기
                OutputStream outputStream = socket.getOutputStream();
                outputStream.write(byteArr);
                outputStream.flush();

                String message = "[보내기 완료]";
                System.out.println(message);

                jTextArea.append(message + "\n");
            }
        }
    };
    thread.start();
}
```

```

        } catch (Exception e) {
            String message = "[서버 통신 안됨]";
            System.out.println(message);
            jTextArea.append(message + "\n");
            stopClient();
        }
    }
};
// 스레드 시작
thread.start();
}

void start() {
    Client3Example client = new Client3Example();

    JFrame jFrame = new JFrame("Client");

    JPanel subPanel = new JPanel();

    mainPanel.setLayout(new BorderLayout());
    subPanel.setLayout(new BorderLayout());

    jTextArea.setLineWrap(true);
    jTextArea.setWrapStyleWord(true);
    JScrollPane scrollPane = new JScrollPane(jTextArea);

    JButton startBtn = new JButton("Start");
    JTextField textField = new JTextField();
    JButton sendBtn = new JButton("Send");

    startBtn.addActionListener(new ActionListener() {
        @Override
        public void actionPerformed(ActionEvent e) {
            if (startBtn.getLabel().equals("Start")) {
                jTextArea.setText("Start" + "\n");
                client.startClient();
                startBtn.setLabel("Stop");
            } else {
                client.stopClient();
                startBtn.setLabel("Start");
            }
        }
    });

    sendBtn.addActionListener(new ActionListener() {
        @Override
        public void actionPerformed(ActionEvent e) {
            client.send(textField.getText());
        }
    });
}

```

```
        textField.setText("");
    }
});

subPanel.add(startBtn, BorderLayout.WEST);
subPanel.add(textField, BorderLayout.CENTER);
subPanel.add(sendBtn, BorderLayout.EAST);

mainPanel.add(jTextArea, BorderLayout.CENTER);
mainPanel.add(subPanel, BorderLayout.SOUTH);

jFrame.add(mainPanel);
jFrame.setSize(500, 300);
jFrame.setVisible(true);
}

public static void main(String[] args) {
    Client3Example client = new Client3Example();
    client.start();
}
}
```