

네트워크프로그래밍 기초 3

한성대학교 컴퓨터공학부

신 성

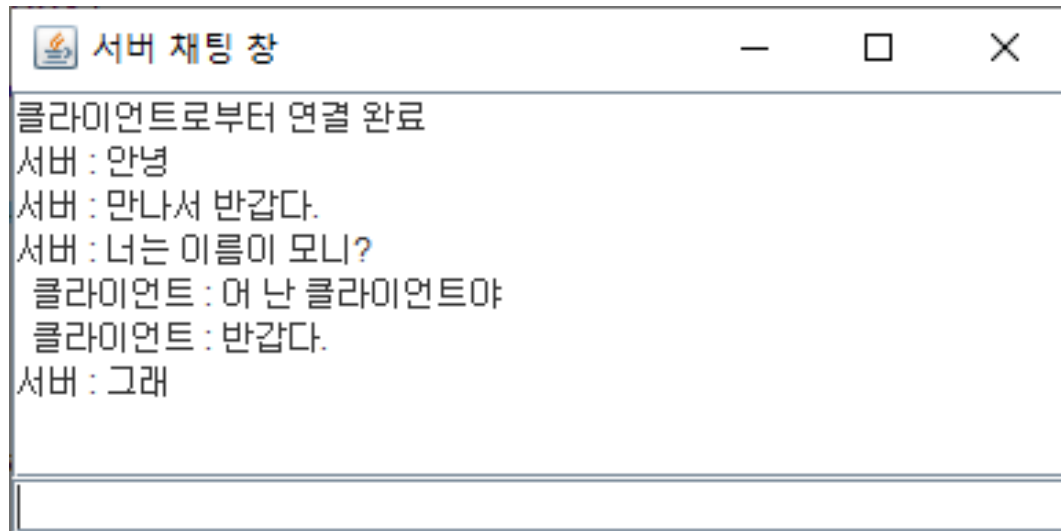


1. 서버-클라이언트 채팅(양방향 통신)

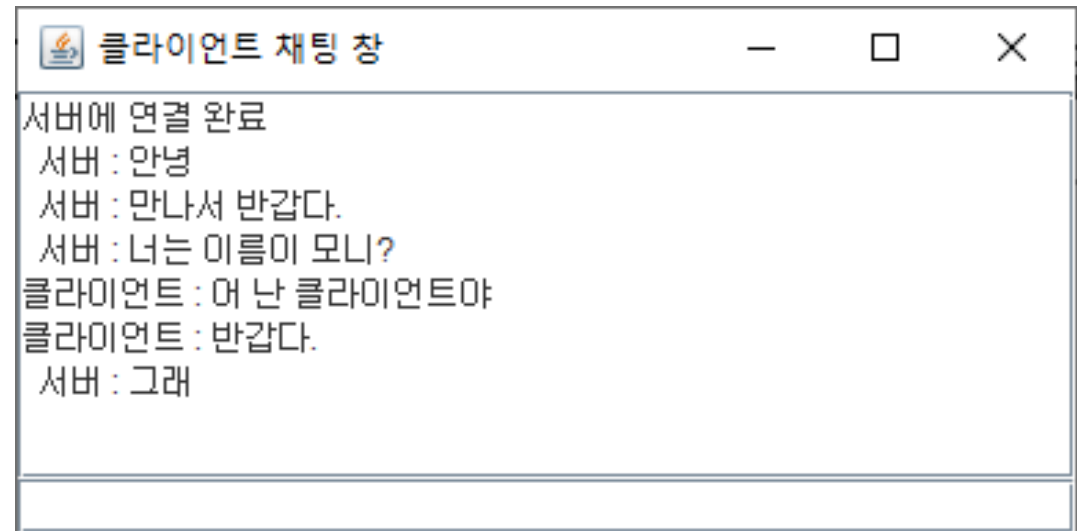
- 예전의 에코서버는 서버와 클라이언트가 한 번씩 순서대로 메시지를 주고 받음
- 이제 순서에 상관없이 자유롭게 여러 번 메시지를 전송하고 받을 수 있도록 작성
- 멀티 스레드 필요

☞ 새로운 서버 생성 후 다음 실행

ChatServer.java



ChatClient.java



```
import java.awt.BorderLayout;
import java.awt.Container;
import java.awt.event.*;
import java.io.*;
import java.net.*;
import javax.swing.*;
```


ActionListener 라는 인터페이스를 구현하는 ChatServer

```
public class ChatServer extends JFrame implements ActionListener {
    private BufferedReader in = null;
    private BufferedWriter out = null;
    private ServerSocket listener = null;
    private Socket socket = null;
    private Receiver receiver; // JTextArea를 상속받고 Runnable 인터페이스를 구현한 클래스로서 받은 정보를 담는 객체
    private JTextField sender; // JTextField 객체로서 보내는 정보를 담는 객체
```

이하 모든 함수에서 공유해서 써야 할 변수 생성

생성자

```
public ChatServer() {  
    setTitle("서버 채팅 창"); // 프레임 타이틀  
    setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE); //프레임 종료 버튼(X)을 클릭하면 프로그램 종료  
    Container c = getContentPane();  
  
    c.setLayout(new BorderLayout()); //BorderLayout 배치관리자의 사용  
    receiver = new Receiver(); // 클라이언트에서 받은 메시지를 출력할 컴퍼넌트  
    receiver.setEditable(false); // 편집 불가  
  
    sender = new JTextField();  
    sender.addActionListener(this);  
  
    add(new JScrollPane(receiver),BorderLayout.CENTER); // 스크롤바를 위해 JScrollPane 이용  
    add(sender,BorderLayout.SOUTH);  
  
    setSize(400, 200); // 폭 400 픽셀, 높이 200 픽셀의 크기로 프레임 크기 설정  
    setVisible(true); // 프레임이 화면에 나타나도록 설정  
  
    try {  
        setupConnection();  
    } catch (IOException e) {  
        handleError(e.getMessage());  
    }  
  
    Thread th = new Thread(receiver); // 상대방부터 메시지 수신을 위한 스레드 생성  
    th.start();  
}
```



```
private void setupConnection() throws IOException {  
    listener = new ServerSocket(9999); // 서버 소켓 생성  
    socket = listener.accept(); // 클라이언트로부터 연결 요청 대기  
    //System.out.println("연결됨");  
    receiver.append("클라이언트로부터 연결 완료");  
    int pos = receiver.getText().length();  
    receiver.setCaretPosition(pos); // caret 포지션을 가장 마지막으로 이동  
  
    in = new BufferedReader(new InputStreamReader(socket.getInputStream())); // 클라이언트로부터의 입력 스트림  
    out = new BufferedWriter(new OutputStreamWriter(socket.getOutputStream())); // 클라이언트의 출력 스트림  
}  
  
private static void handleError(String string) {  
    System.out.println(string);  
    System.exit(1);  
}
```

Runnable 이라는 인터페이스를 구현하는 Receiver

```
private class Receiver extends JTextArea implements Runnable {
    @Override
    public void run() {
        String msg = null;
        while (true) {
            try {
                msg = in.readLine(); // 상대방부터 한 행의 문자열 받기
            } catch (IOException e) {
                handleError(e.getMessage());
            }
            this.append("\n 클라이언트 : " + msg); // 받은 문자열을 JTextArea에 출력
            int pos = this.getText().length();
            this.setCaretPosition(pos); // caret 포지션을 가장 마지막으로 이동
        }
    }
}
```

이런 방식으로 하면 하나의 클래스로도 됨

```
@Override
public void actionPerformed(ActionEvent e) { // JTextField에 <Enter> 키 처리
    if (e.getSource() == sender) {
        String msg = sender.getText(); // 텍스트 필드에서 문자열 얻어옴
        try {
            out.write(msg+"\n"); // 문자열 전송
            out.flush();

            receiver.append("\n서버 : " + msg); // JTextArea에 출력
            int pos = receiver.getText().length();
            receiver.setCaretPosition(pos); // caret 포지션을 가장 마지막으로 이동
            sender.setText(null); // 입력창의 문자열 지움
        } catch (IOException e1) {
            handleError(e1.getMessage());
        }
    }
}

public static void main(String[] args) {
    new ChatServer();
}
```

```
import java.awt.*;
import java.awt.event.*;
import java.io.*;
import java.net.*;
import javax.swing.*;

public class ChatClient extends JFrame implements ActionListener {
    private BufferedReader in = null;
    private BufferedWriter out = null;
    private Socket socket = null;
    private Receiver receiver = null; // JTextArea를 상속받고 Runnable 인터페이스를 구현한 클래스로서 받은 정보를 담는 객체
    private JTextField sender = null; // JTextField 객체로서 보내는 정보를 담는 객체
```



```
public ChatClient() {
    setTitle("클라이언트 채팅 창"); // 프레임 타이틀
    setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE); //프레임 종료 버튼(X)을 클릭하면 프로그램 종료
    Container c = getContentPane();

    c.setLayout(new BorderLayout()); //BorderLayout 배치관리자의 사용
    receiver = new Receiver(); // 서버에서 받은 메시지를 출력할 컴퍼넌트
    receiver.setEditable(false); // 편집 불가

    sender = new JTextField();
    sender.addActionListener(this);

    c.add(new JScrollPane(receiver),BorderLayout.CENTER); // 스크롤바를 위해 JScrollPane 이용
    c.add(sender,BorderLayout.SOUTH);

    setSize(400, 200); // 폭 400 픽셀, 높이 200 픽셀의 크기로 프레임 크기 설정
    setVisible(true); // 프레임이 화면에 나타나도록 설정

    try {
        setupConnection();
    } catch (IOException e) {
        handleError(e.getMessage());
    }

    Thread th = new Thread(receiver); // 상대방부터 메시지 수신을 위한 스레드 생성
    th.start();
}
```

```
private void setupConnection() throws IOException {
    socket = new Socket("localhost", 9999); // 클라이언트 소켓 생성
    // System.out.println("연결됨");
    receiver.append("서버에 연결 완료");
    int pos = receiver.getText().length();
    receiver.setCaretPosition(pos); // caret 포지션을 가장 마지막으로 이동

    in = new BufferedReader(new InputStreamReader(socket.getInputStream())); // 클라이언트로부터의 입력 스트림
    out = new BufferedWriter(new OutputStreamWriter(socket.getOutputStream())); // 클라이언트의 출력 스트림
}

private static void handleError(String string) {
    System.out.println(string);
    System.exit(1);
}
```

```
private class Receiver extends JTextArea implements Runnable {
    @Override
    public void run() {
        String msg = null;
        while (true) {
            try {
                msg = in.readLine(); // 상대방부터 한 행의 문자열 받기
            } catch (IOException e) {
                handleError(e.getMessage());
            }
            this.append("\n 서버 : " + msg); // 받은 문자열을 JTextArea에 출력
            int pos = this.getText().length();
            this.setCaretPosition(pos); // caret(커서)을 가장 마지막으로 이동
        }
    }
}
```

@Override

```
public void actionPerformed(ActionEvent e) { // JTextField에 <Enter> 키 처리
    if (e.getSource() == sender) {
        String msg = sender.getText(); // 텍스트 필드에 사용자가 입력한 문자열
        try {
            out.write(msg+"\n"); // 문자열 전송
            out.flush();

            receiver.append("\n클라이언트 : " + msg); // JTextArea에 출력
            int pos = receiver.getText().length();
            receiver.setCaretPosition(pos); // caret 포지션을 가장 마지막으로 이동
            sender.setText(null); // 입력창의 문자열 지움
        } catch (IOException e1) {
            handleError(e1.getMessage());
        }
    }
}

public static void main(String[] args) {
    new ChatClient();
}
```

Runnable 인터페이스로 스레드 만들기

■ 스레드 클래스 작성

- Runnable 인터페이스 구현하는 새 클래스 작성

```
class TimerRunnable implements Runnable {  
    .....  
    @Override  
    public void run() { // run() 메소드 구현  
        .....  
    }  
}
```

■ 스레드 코드 작성

- run() 메소드 구현
 - run() 메소드를 스레드 코드라고 부름
 - run() 메소드에서 스레드 실행 시작

■ 스레드 객체 생성

```
Thread th = new Thread(new TimerRunnable());
```

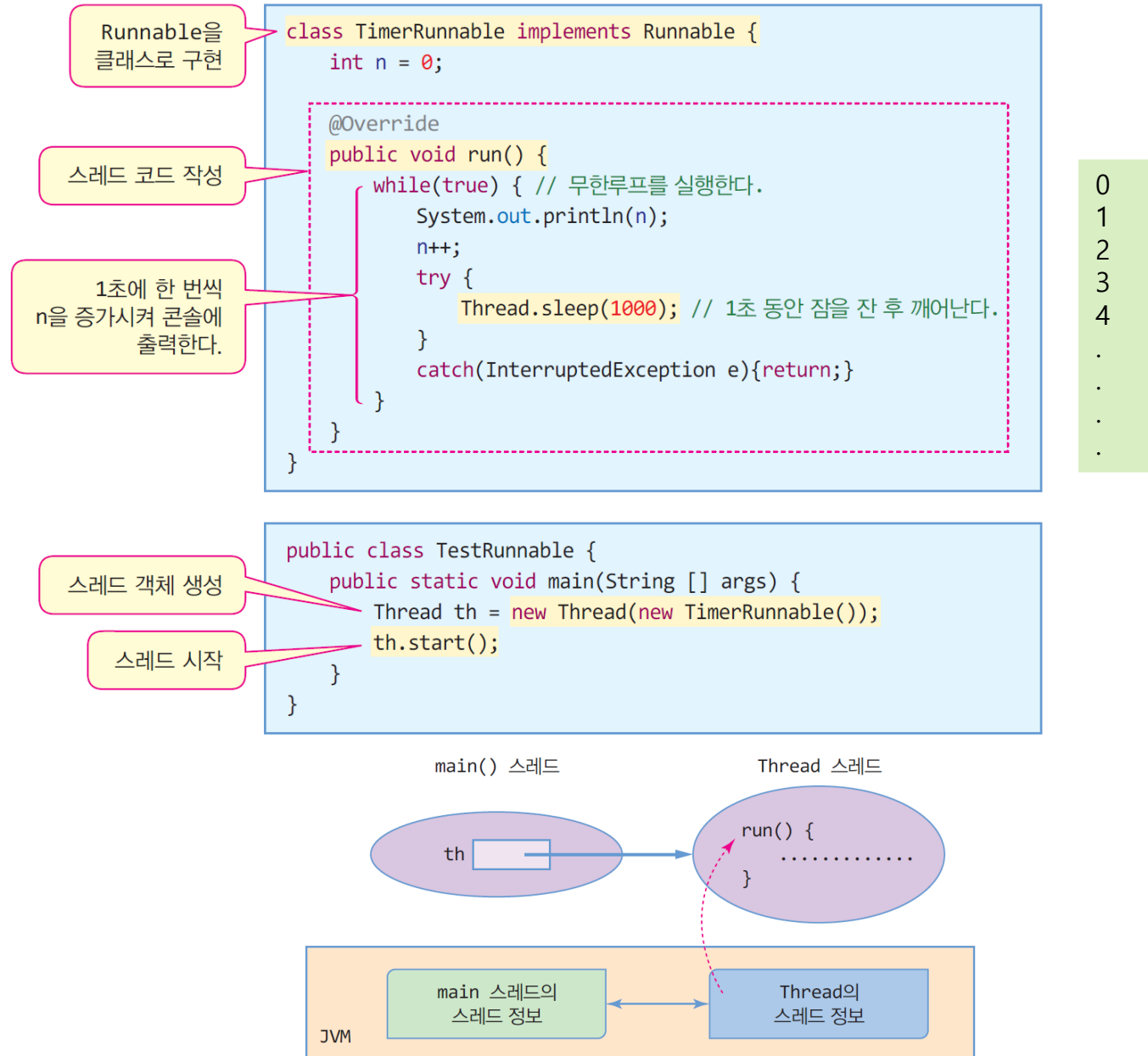
DP 를 이용한 이런 방식도 된다

■ 스레드 시작

- start() 메소드 호출

```
th.start();
```

*Runnable 인터페이스를 상속받아 1초 단위로 초 시간을 출력하는 스레드 작성



예제 : Runnable 인터페이스를 이용하여 1초 단위로 출력하는 타이머 스레드 만들기

```
import java.awt.*;
import javax.swing.*;
```

```
class TimerRunnable implements Runnable {
    private JLabel timerLabel;

    public TimerRunnable(JLabel timerLabel) {
        this.timerLabel = timerLabel;
    }
    @Override
    public void run() {
        int n=0;
        while(true) {
            timerLabel.setText(Integer.toString(n));
            n++;
            try {
                Thread.sleep(1000);
            }
            catch(InterruptedException e) {
                return;
            }
        }
    }
}
```

```
public class RunnableTimerEx extends JFrame {
    public RunnableTimerEx() {
        setTitle("Runnable을 구현한 타이머 스레드 예제");
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        Container c = getContentPane();
        c.setLayout(new FlowLayout());
        JLabel timerLabel = new JLabel();
        timerLabel.setFont(new Font("Gothic", Font.ITALIC, 80));
        c.add(timerLabel);

        TimerRunnable runnable = new TimerRunnable(timerLabel);
        Thread th = new Thread(runnable);

        setSize(250,150);
        setVisible(true);

        th.start();
    }
    public static void main(String[] args) {
        new RunnableTimerEx();
    }
}
```

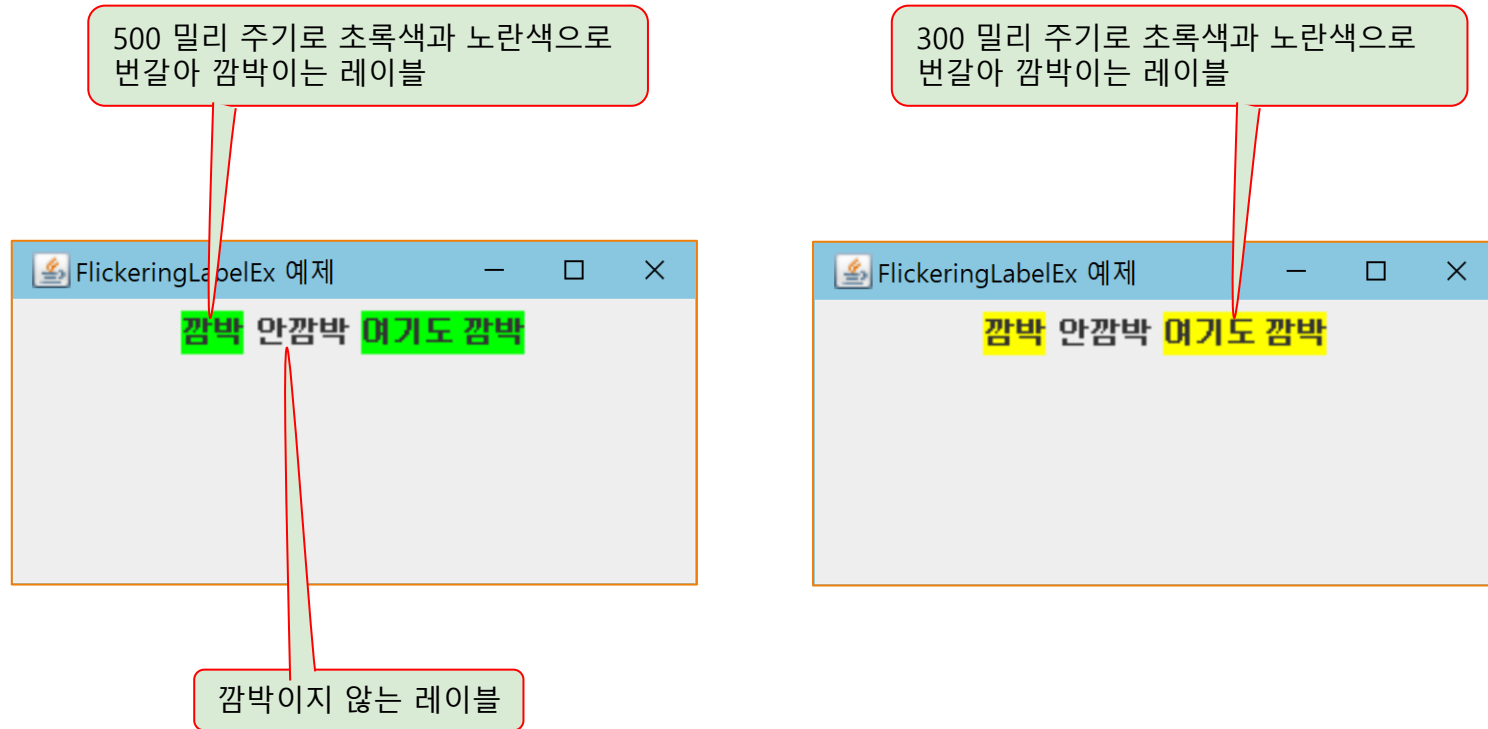
2

7

23

예제 : 깜박이는 문자열을 가진 레이블 만들기

JLabel을 상속받아 문자열을 깜박이는 FlickeringLabel 컴포넌트를 작성하라.



예제 : 정답 코드

```
import java.awt.*;
import javax.swing.*;

class FlickeringLabel extends JLabel implements Runnable {
    private long delay;
    public FlickeringLabel(String text, long delay) {
        super(text);
        this.delay = delay;
        setOpaque(true);
        Thread th = new Thread(this);
        th.start();
    }
    @Override
    public void run() {
        int n=0;
        while(true) {
            if(n == 0)
                setBackground(Color.YELLOW);
            else
                setBackground(Color.GREEN);
            if(n == 0) n = 1;
            else n = 0;
            try {
                Thread.sleep(delay);
            }
            catch (InterruptedException e) {
                return;
            }
        }
    }
}
```

```
public class FlickeringLabelEx extends JFrame {
    public FlickeringLabelEx() {
        setTitle("FlickeringLabelEx 예제");
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        Container c = getContentPane();
        c.setLayout(new FlowLayout());

        // 깜박이는 레이블 생성
        FlickeringLabel fLabel = new FlickeringLabel("깜박",500);

        // 깜박이지 않는 레이블 생성
        JLabel label = new JLabel("안깜박");

        // 깜박이는 레이블 생성
        FlickeringLabel fLabel2 = new FlickeringLabel("여기도 깜박",300);

        c.add(fLabel);
        c.add(label);
        c.add(fLabel2);

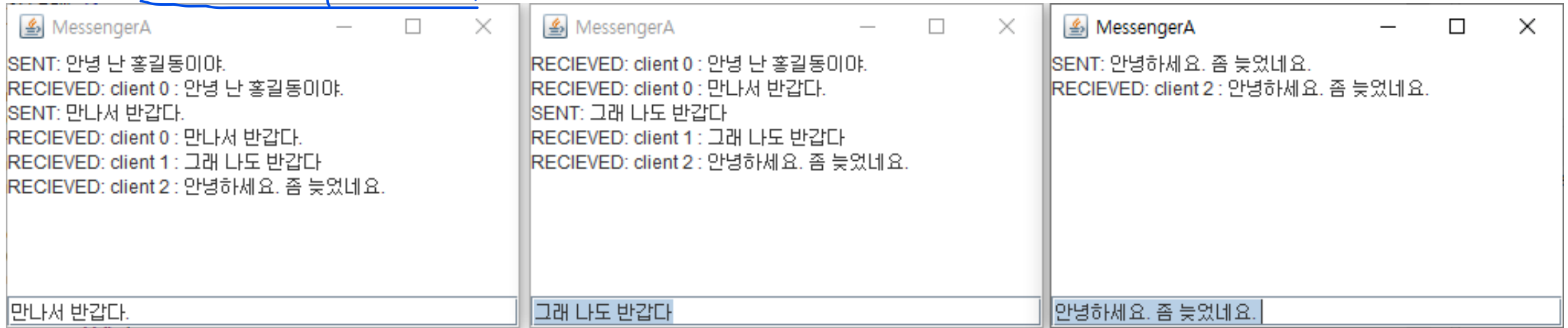
        setSize(300,150);
        setVisible(true);
    }

    public static void main(String[] args) {
        new FlickeringLabelEx();
    }
}
```

2. 다자 회의(채팅) 시스템

- 여러 명의 클라이언트가 같이 채팅
- 서버에서는 ArrayList를 이용하여 클라이언트 정보를 저장한다. 그리고 클라이언트가 접속할 때마다 서비스 스레드를 생성하여서 각 클라이언트를 서비스
- 만약 어떤 클라이언트가 메시지를 보내면 이 메시지는 ArrayList에 저장된 모든 클라이언트에게 보내면 된다.

ServerM.java (GUI 없음) 실행 후 MessengerMulti.java 여러 개 실행 가능(올려드린 파일 참고)



클라1

클라2

클라3

Q&A

담당교수 : 신성

E-mail : sihns@hansung.ac.kr

연구실 : 우촌관 702호

휴대폰 번호 : 010-8873-8353