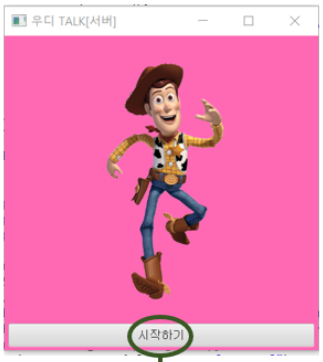
[소개]

Woody Talk

지난번, 교수님이 강의해주신
서버, 클라이언트 연결 강의에서
좀 더 발전해서
자바 FX를 이용하여 서버 클라이언트
채팅 프로그램을 만들어보았습니다.

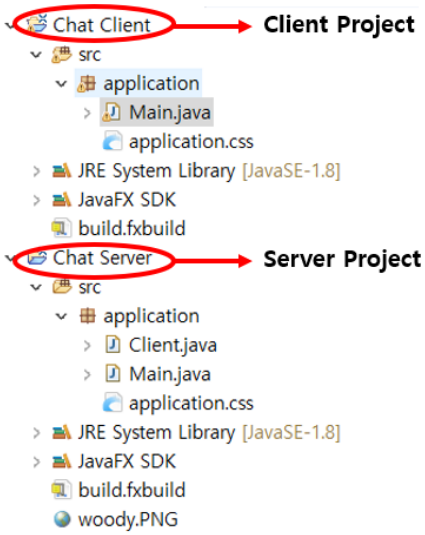


<서버 창>

버튼 누르면 서버 작동

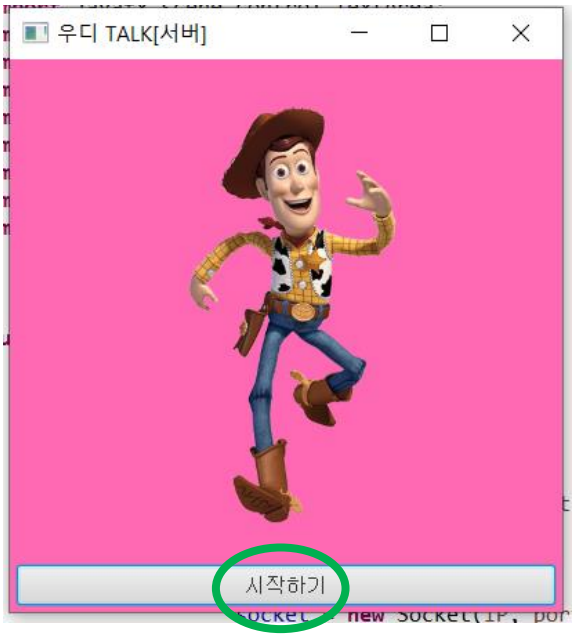
<채팅 창 [클라이언트]>

Woody Talk



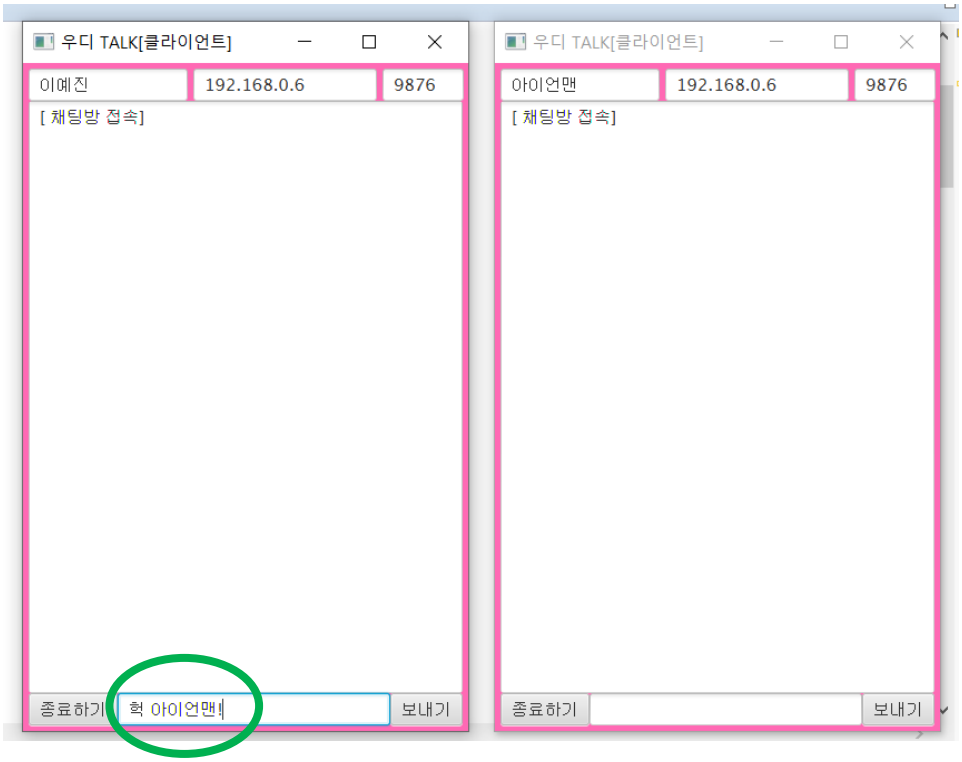
서버와 클라이언트가 따로 동작 해야 하기 때문에
서버와 클라이언트로 나눠서 프로젝트를 생성하였습니다.

[구현]



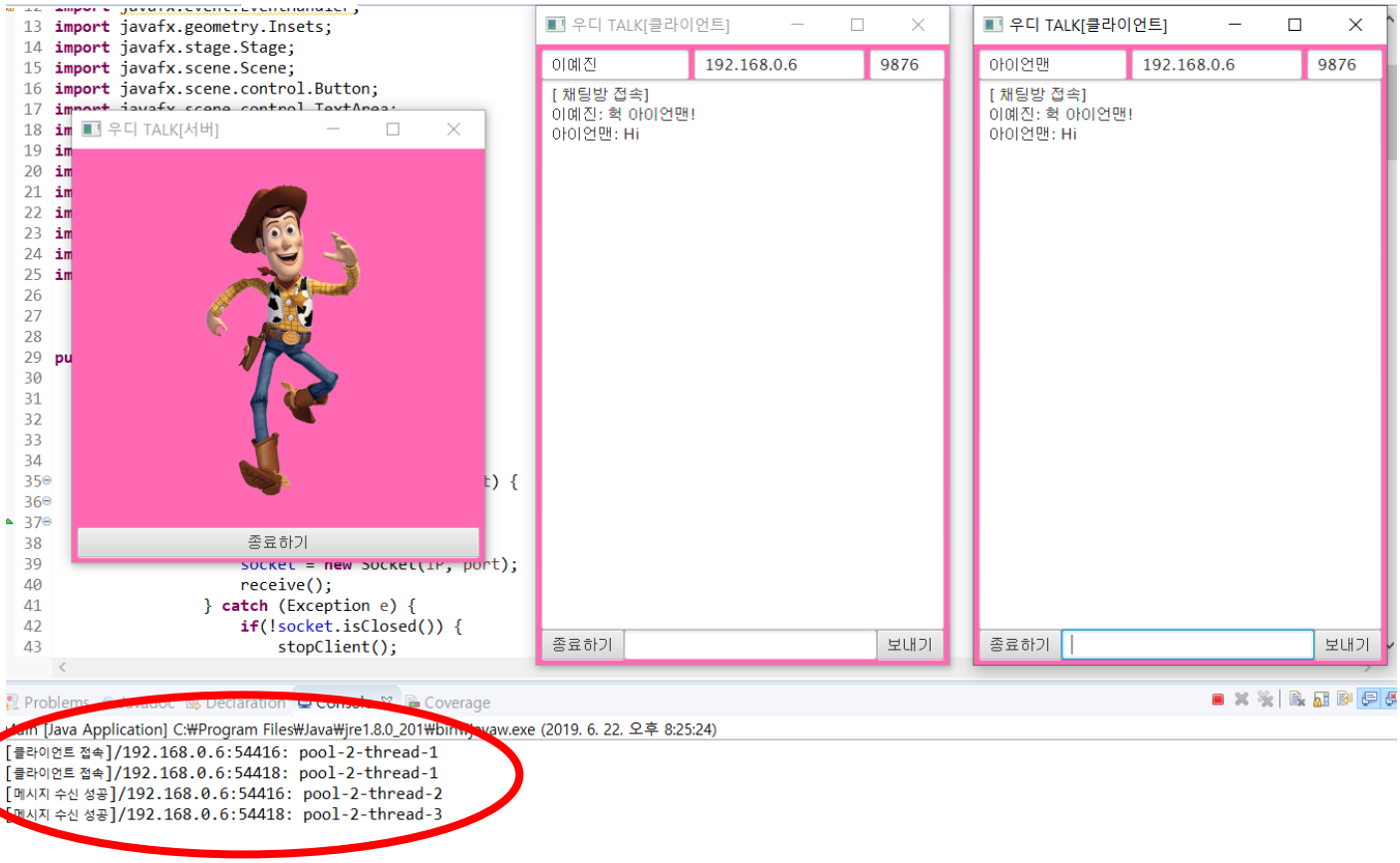
먼저 서버창에서 시작하기 버튼을 누릅니다.

두개의 클라이언트 창으로 채팅을 할 예정입니다.



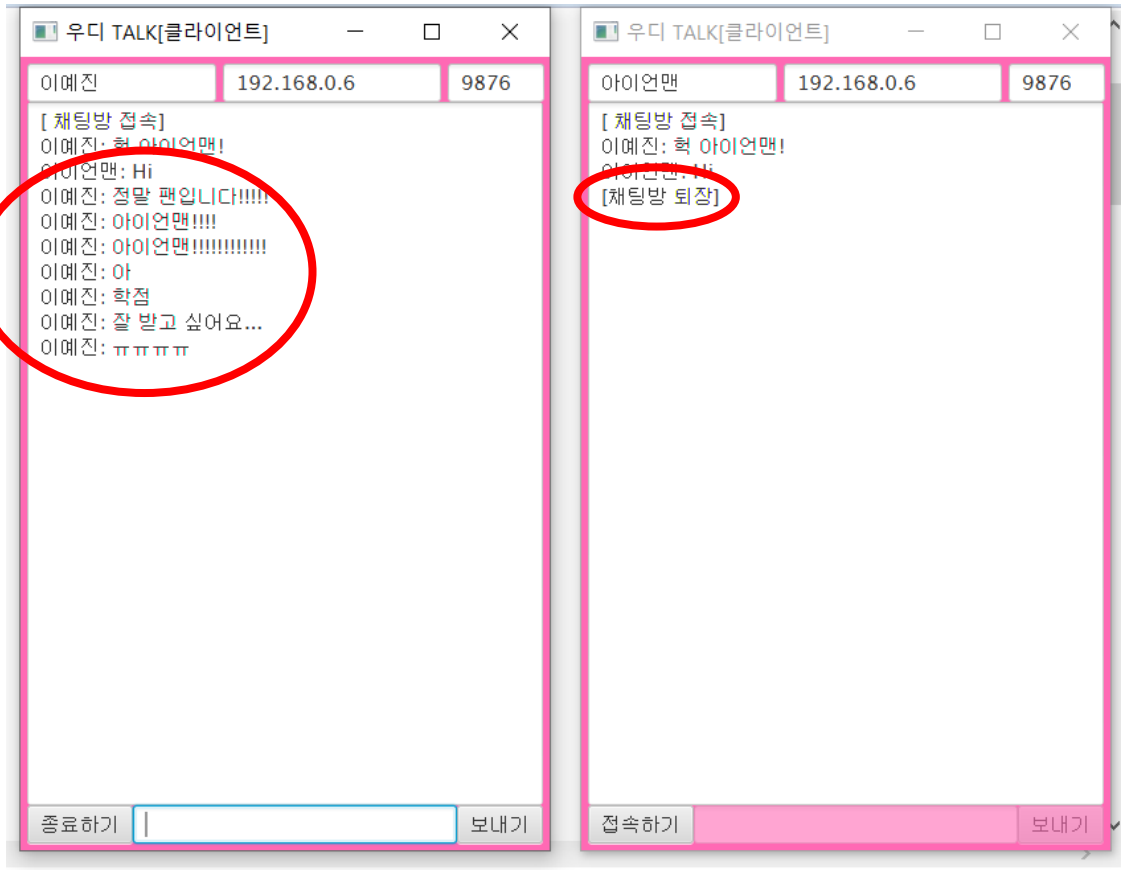
제가 아이언맨에게

대화를 보냅니다.



그 결과, 아이언맨과 **서로 대화를 나눌 수 있습니다.**

콘솔창을 보면 각 클라이언트에게 스레드가 각각 할당된 것을 확인해 볼 수 있습니다.

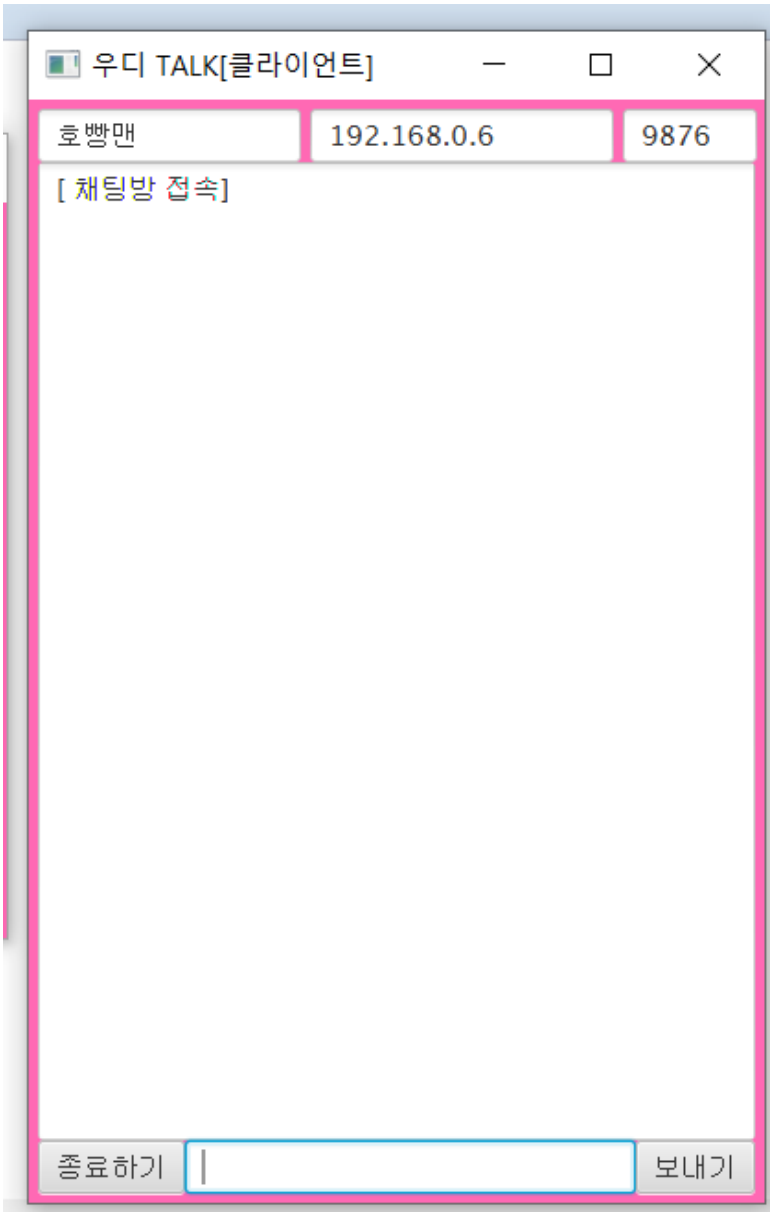


아이언맨이 종료하기 **버튼을 누르고 퇴장**을 했더니

제가 아무리 메시지를 보내도

아이언맨이 접속한 창에는

메시지가 뜨지 않습니다.



서버에 연결을 하지 않은 상태로

접속을 시도했더니...

```
Main (1) [Java Application] C:₩Program Files₩Java₩jre1.8.0_201₩bin₩javaw.exe (2019. 6. 22. 오후 8:34
Exception in thread "Thread-4" java.lang.NullPointerException
        at application.Main$1.run(Main.java:42)
```

이렇게 오류가 뜨는 것을 확인 할 수 있습니다.

[Chat Server안에 있는 Main.java]

```
package application;

import java.net.InetSocketAddress;
import java.net.ServerSocket;
import java.net.Socket;
import java.util.Iterator;
import java.util.Vector;
import java.util.concurrent.ExecutorService;
import java.util.concurrent.Executors;

import javafx.application.Application;
import javafx.application.Platform;
import javafx.geometry.Insets;
import javafx.stage.Stage;
import javafx.scene.Scene;
import javafx.scene.control.Button;
import javafx.scene.control.TextArea;
import javafx.scene.image.Image;
import javafx.scene.image.ImageView;
import javafx.scene.layout.Background;
import javafx.scene.layout.BackgroundFill;
import javafx.scene.layout.BorderPane;
import javafx.scene.layout.CornerRadii;
import javafx.scene.paint.Color;
import javafx.scene.text.Font;



public class Main extends Application {

	public static ExecutorService threadPool;
	//threadPool로 thread 수 제한해서 서버 성능 저하 방지해준다.
	public static Vector<Client> clients = new Vector<Client>();
	//접속한 클라이언트들을 관리할 수 있게 해줌

	ServerSocket serverSocket;

	//서버를 구동해서 클라이언트 연결을 기다린다.
	public void startServer(String IP, int port) {
	//어떤 IP와 port로 연결하는지
		try {
			serverSocket = new ServerSocket();
			serverSocket.bind(new InetSocketAddress(IP, port));
		} catch (Exception e) {
			e.printStackTrace();
			if(!serverSocket.isClosed()) {
				stopServer();
			}
			return;
		}
```

```
        //클라이언트가 접속할 때까지 계속 기다리는 쓰레드
        Runnable thread = new Runnable() {
                @Override
                public void run() {
                        while(true) {
                                try {
                                        Socket socket = serverSocket.accept();
                                        clients.add(new Client(socket));
                                        System.out.println("[클라이언트 접속]"
                                                        +socket.getRemoteSocketAddress()
+ ": " + Thread.currentThread().getName());
                                } catch (Exception e) {
                                        if(!serverSocket.isClosed()) {
                                                stopServer();
                                        }
                                        break;
                                }
                        }

                }

        };
        threadPool = Executors.newCachedThreadPool();
        threadPool.submit(thread);
}

//서버의 작동을 중지시킨다.
public void stopServer() {
        try {
                Iterator<Client> iterator = clients.iterator();
        //작동중인 모든 소켓을 닫아줍니다.
                while(iterator.hasNext()) {
                        Client client = iterator.next();
                        client.socket.close();
                        iterator.remove();
        }
        //서버 소켓 객체 닫기
        if(serverSocket != null && !serverSocket.isClosed()) {
                serverSocket.close();
        }
        //쓰레드 풀 종료하기
        if(threadPool != null && !threadPool.isShutdown()) {
                threadPool.shutdown();
        }
        } catch (Exception e) {
                e.printStackTrace();
        }
}



//UI만들고, 프로그램 동작
@Override
public void start(Stage primaryStage) {
        BorderPane root = new BorderPane();
        root.setPadding(new Insets(5));
        TextArea textArea = new TextArea();
        textArea.setEditable(false);
        textArea.setFont(new Font("나눔고딕", 15));
```

```
                Button toggleButton = new Button("시작하기");
                toggleButton.setMaxWidth(Double.MAX_VALUE);
                BorderPane.setMargin(toggleButton, new Insets(1, 0, 0, 0));
                root.setBottom(toggleButton);

                String IP = "192.168.0.6";
                int port = 9876;

                toggleButton.setOnAction(event -> {
                        if(toggleButton.getText().equals("시작하기")) {
                                startServer(IP, port);
                                Platform.runLater(() -> {
                                        String message = String.format("[서버 시작]\n", IP, port);
                                        textArea.appendText(message);
                                        toggleButton.setText("종료하기");
                                });
                        } else {
                                stopServer();
                                Platform.runLater(() -> {
                                        String message = String.format("[서버 종료]\n", IP, port);
                                        textArea.appendText(message);
                                        toggleButton.setText("시작하기");
                        });
                }

        });


                Scene scene = new Scene(root, 400, 400);
                BackgroundFill background_fill = new BackgroundFill(Color.HOTPINK,
CornerRadii.EMPTY, Insets.EMPTY);
                Background background = new Background(background_fill);
                root.setBackground(background);
                //이미지 삽입
                Image image = new Image("file:woody.png");
                ImageView iv = new ImageView();
                iv.setImage(image);

                iv.setPreserveRatio(true);
                iv.setFitHeight(320);
                iv.setLayoutY(30);    //이미지 위치 조절했어요
                iv.setLayoutX(120);
                root.getChildren().add(iv);

                primaryStage.setTitle("우디 TALK[서버]");
                primaryStage.setOnCloseRequest(event -> stopServer());
                primaryStage.setScene(scene);
                primaryStage.show();
        }

        // 프로그램 진입
        public static void main(String[] args) {
                launch(args);
        }
}
```

[Chat Server안에 있는 Client.java]

```
package application;

import java.io.IOException;
import java.io.InputStream;
import java.io.OutputStream;
import java.net.Socket;

//한명의 클라이언트랑 통신할 수 있게 하는 클래스 생성
public class Client {

	Socket socket; //소켓생성

	public Client(Socket socket) {
		this.socket = socket; //초기화
		receive();
	}

	//클라이언트에게 메세지를 전달 받는 메소드
	public void receive() {
	   Runnable thread = new Runnable() {       //하나의 thread 생성.
		@Override
		public void run() {
		//run안에서 thread가 어떻게 동작하는지 정의해줍니다.
		   try {
			while(true) {
			  InputStream in = socket.getInputStream();
			//내용전달받게 inputstream객체사용
			  byte[] buffer = new byte[512];     //512바이트 전달받음
			  int length = in.read(buffer);
			  while(length == -1) throw new IOException();
			  System.out.println("[메시지 수신 성공]"
				+ socket.getRemoteSocketAddress()
				+ ": "+ Thread.currentThread().getName());
			  String message = new String(buffer, 0,length, "UTF-8");
			  for(Client client : Main.clients) {
				client.send(message);
			  }
			}
		   } catch(Exception e) {
			try {
				System.out.println("[메시지 수신 오류]"
					+ socket.getRemoteSocketAddress()
					+ ": " + Thread.currentThread().getName());
			} catch (Exception e2) {
				e2.printStackTrace();
			}
		   }
		  }
	    };
		Main.threadPool.submit(thread);
	}
```

```
	//클라이언트에게 메세지를 전송하는 메소드
	public void send(String message) {
		Runnable thread = new Runnable() {
			@Override
			public void run() {
				try {
					OutputStream out = socket.getOutputStream();
				//보낼때는 outputstream !!!
					byte[] buffer = message.getBytes("UTF-8");
					out.write(buffer);
					out.flush();
				} catch (Exception e) {
					try {
						System.out.println("[메시지 송신 오류]"
								+ socket.getRemoteSocketAddress()
								+ ": " +
Thread.currentThread().getName());
						Main.clients.remove(Client.this);
						socket.close();
					} catch (Exception e2) {
						e2.printStackTrace();
					}
				}

			}

		};
		Main.threadPool.submit(thread);

	}
}
```

[Chat Client 안에 있는 Client.java]

```
package application;

import java.io.IOException;
import java.io.InputStream;
import java.io.OutputStream;
import java.net.Socket;



import javafx.application.Application;
import javafx.application.Platform;
import javafx.event.EventHandler;
import javafx.geometry.Insets;
import javafx.stage.Stage;
import javafx.scene.Scene;
import javafx.scene.control.Button;
import javafx.scene.control.TextArea;
import javafx.scene.control.TextField;
import javafx.scene.layout.Background;
import javafx.scene.layout.BackgroundFill;
import javafx.scene.layout.BorderPane;
import javafx.scene.layout.CornerRadii;
import javafx.scene.layout.HBox;
import javafx.scene.layout.Priority;
import javafx.scene.paint.Color;



public class Main extends Application {

	Socket socket;
	TextArea textArea;

	//클라이언트 프로그램 동작 메소드
	public void startClient(String IP, int port) {
		Thread thread = new Thread() {
			public void run() {
				try {
					socket = new Socket(IP, port);
					receive();
				} catch (Exception e) {
					if(!socket.isClosed()) {
						stopClient();
						System.out.println("[서버 접속 실패]");
						Platform.exit();
					}
				}
			}
		};
		thread.start();
	}

	// 클라이언트 프로그램 종료 메소드
	public void stopClient() {
		try {
			if(socket != null && !socket.isClosed()) {
				socket.close();
			}
		} catch (Exception e) {
			e.printStackTrace();
		}
```

```
}

//서버로부터 메시지를 전달받는 메소드 (무한루프)
public void receive() {
	while(true) {
		try {
			InputStream in = socket.getInputStream();
			//메세지를 받을때는 inputstream 사용합니다.
			byte[] buffer = new byte[512];
			int length = in.read(buffer);
			if(length == -1) throw new IOException();
			String message =  new String(buffer, 0, length, "UTF-8");
			Platform.runLater(() -> {
				textArea.appendText(message);
			});
		} catch (Exception e) {
			stopClient();
			break;
		}
	}
}

//서버로 메시지를 보내는 메소드
public void send(String message) {
	Thread thread = new Thread() {
		public void run() {
			try {
				OutputStream out = socket.getOutputStream();
//메세지를 보낼때는 outputstream사용합니다.
				byte[] buffer = message.getBytes("UTF-8");
				out.write(buffer);
				out.flush();
			} catch (Exception e) {
				stopClient();
			}
		}
	};
	thread.start();
}

//실제로 프로그램을 동작시키는 메소드
@Override
public void start(Stage primaryStage) {
	BorderPane root = new BorderPane();
	root.setPadding(new Insets(5));
	HBox hbox = new HBox();
	hbox.setSpacing(5);

	TextField userName = new TextField();
	userName.setPrefWidth(150);
	userName.setPromptText("이름을 입력하세요.");
	HBox.setHgrow(userName, Priority.ALWAYS);

	TextField IPText =  new TextField("192.168.0.6");
	//제 PC의 IP주소를 넣었습니다.
	TextField portText = new TextField("9876");
	portText.setPrefWidth(80);

	hbox.getChildren().addAll(userName, IPText, portText);
	root.setTop(hbox);
```

```
textArea = new TextArea();
textArea.setEditable(false); //화면에 출력된 내용을 봐야한다.
root.setCenter(textArea);

TextField input = new TextField();
input.setPrefWidth(Double.MAX_VALUE);
input.setDisable(true);

input.setOnAction(event-> {
	send(userName.getText() + ": " + input.getText() + "\n");
	input.setText("");
	input.requestFocus();
});

Button sendButton = new Button("보내기");
sendButton.setDisable(true);

sendButton.setOnAction(event-> {
	send(userName.getText() +": " + input.getText() + "\n");
	input.setText("");
	input.requestFocus();
});

Button connectionButton = new Button("접속하기");
connectionButton.setOnAction(event-> {
	if(connectionButton.getText().equals("접속하기")) {
		int port =9876;
		try {
			port = Integer.parseInt(portText.getText());
		} catch (Exception e) {
			e.printStackTrace();
		}
		startClient(IPText.getText(), port);
		Platform.runLater(() -> {
			textArea.appendText("[ 채팅방 접속]\n");
		});
		connectionButton.setText("종료하기");
		input.setDisable(false);
		sendButton.setDisable(false);
		input.requestFocus();
	} else {
		stopClient();
		Platform.runLater(()-> {
			textArea.appendText("[채팅방 퇴장]\n");
		});
		connectionButton.setText("접속하기");
		input.setDisable(true);
		sendButton.setDisable(true);
	}
});
```

```
                BorderPane pane = new BorderPane();


                pane.setLeft(connectionButton);
                pane.setCenter(input);
                pane.setRight(sendButton);

                root.setBottom(pane);
                Scene scene = new Scene(root, 400, 600);
                //배경색
                BackgroundFill background_fill = new BackgroundFill(Color.HOTPINK,
CornerRadii.EMPTY, Insets.EMPTY);
                Background background = new Background(background_fill);
                root.setBackground(background);
                primaryStage.setTitle("우디 TALK[클라이언트]");
                primaryStage.setScene(scene);

                primaryStage.setOnCloseRequest(event -> stopClient());
                primaryStage.show();

                connectionButton.requestFocus();

        }

        //프로그램 진입
        public static void main(String[] args) {
                launch(args);
        }
}
```

Woody Talk - 소감

Woody Talk을 직접 만들면서 ThredPool에 대해서 익히고
Server와 Client에 대해 공부하고
연습해 볼 수 있어서 정말 좋았습니다.