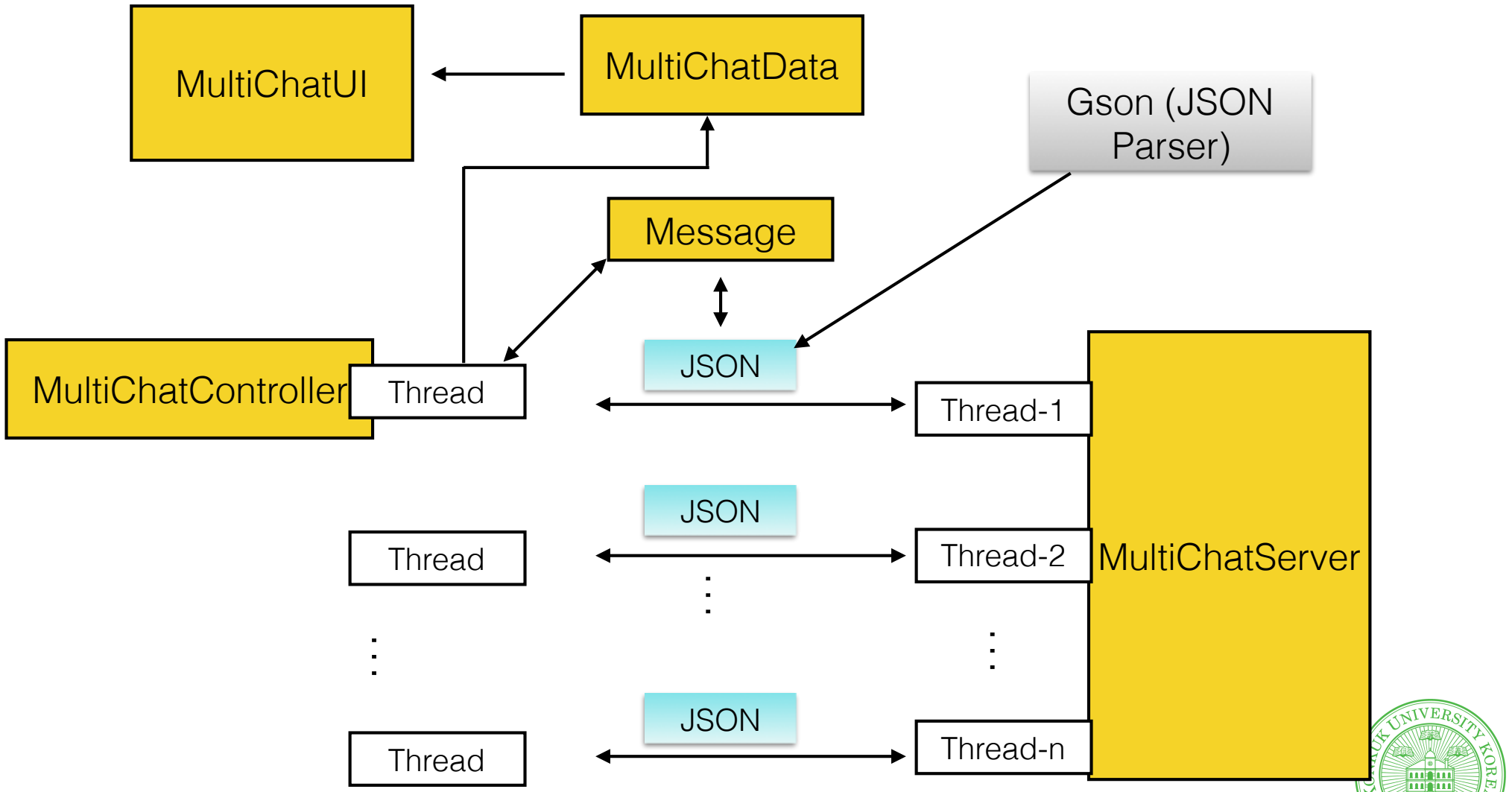
Project Description: Overview

Hyang-Won Lee
Department of Software, Konkuk University
leehw@konkuk.ac.kr
https://sites.google.com/site/leehwko/

Multi Chat Client/Server

: class in this project

- Project architecture



주요기술요소

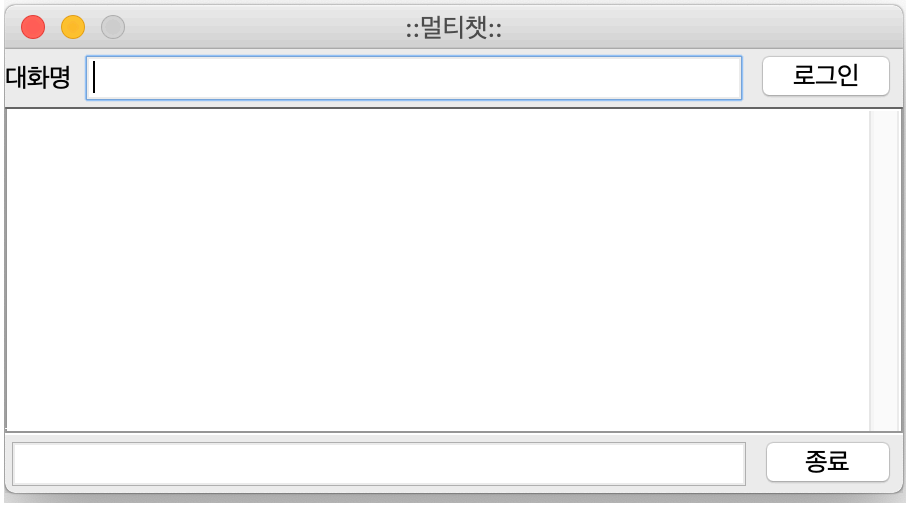
- 객체지향
  - 클라이언트와 서버로 메인 프로그램을 분리하고, 프로그램을 구성하는 클래스는 객체지향 프로그래밍 기법에 따라 세분화시키고 계층화
  - 인터페이스, 추상 클래스, 상속 메서드 오버로딩, 메서드 오버라이딩 직간접 적용
- UI, 멀티스레드, 네트워크 프로그래밍
  - Swing을 이용하여 클라이언트 UI구성
  - 동시에 여러 사용자가 접속할 수 있도록 멀티스레드 네트워크 서버 구현
  - 메시지 입력과 수신의 동시처리를 위한 스레드 기반 클라이언트 구현
  - TCP/IP 소켓을 이용하여 네트워크 프로그래밍

기능정의

- 클라이언트
  - 로그인 및 로그아웃
  - 대화명 입력 및 표시
  - 채팅 메시지 출력
  - 프로그램 종료
- 서버
  - 클라이언트 대기 및 연결
  - 다중 클라이언트 채팅 지원
  - 연결된 클라이언트 목록 관리
  - 채팅 메시지 수신 및 브로드 캐싱
  - 로그 출력

MultiChatUI Class

- 화면의 구성요소 정의, 레이아웃 이용하여 컴포넌트 배치
- 주요 메소드
  - MultiChatUI()
    - 화면을 구성하는 컴포넌트 초기화 및 레이아웃 배치 등
  - void addButtonActionListener
    - 이벤트 핸들러 등록 메서드로, 모든 버튼의 이벤트 핸들러를 한곳에서 등록
- 구현하게 될 UI 예시



MultiChatController Class

- 프로그램의 메인
  - UI와 서버 연결 및 채팅 메시지 전달
- 주요메소드
  - MultiChatController(MultiChatData chatData, MultiChatUI v)
  - void appMain()
    - 컨트롤러 클래스 메인 로직; UI에서 발생한 이벤트를 위임받아 처리
  - void connectServer()
    - 채팅 서버 접속을 위한 메소드
  - void run()
    - 서버 연결 후 메시지 수신을 UI동작과 상관없이 독립적으로 처리하는 스레드를 실행
  - main(String[] args)

MultiChatData Class

- 화면에 필요한 데이터를 제공하고 업데이트하는 기능 제공
- 채팅 프로그램에서 데이터 변경이 수시로 발생하는 부분
  - JTextArea에 출력하는 채팅 메시지
- 주요메소드
  - addObj(JComponent comp)
    - 데이터를 변경할 때 업데이트할 UI 컴포넌트를 등록
  - void refreshData(String msg)
    - 입력인자로 전달된 메시지 내용으로 UI 데이터 업데이트; 채팅 메시지 창의 텍스트를 추가하는 작업 수행

Message Class

- 클라이언트와 서버 간의 통신에 사용하는 JSON 규격의 메시지를 좀 더 쉽게 사용하려고 자바 객체로 변환하는 데 필요한 클래스
  - 참고: JSON(JavaScript Object Notation)은 원래 자바스크립트에서 객체를 표현하려고 만든 구문인데, 지금은 인터넷으로 시스템이나 프로그램 간에 데이터를 주고받는 메시지 규격으로 널리 사용됨
  - 이 프로젝트에서는 구글에서 만든 JSON파서인 Gson을 사용 (관련 라이브러리 추가 방법: ecampus 강의자료 참고)
- 주요 메소드
  - 간단한 getter/setter들

MultiChatServer Class

- 여러 클라이언트와 동시에 연결할 수 있는 서버 구현
- 연결된 클라이언트 또한 ArrayList를 이용하여 관리
- 주요 메소드
  - void start()
    - 서버의 메인 실행 메소드; ServerSocket을 생성하고 클라이언트 연결 및 스레드 생성/처리
  - msgSendAll(String msg)
    - 서버가 수신한 메시지를 연결된 모든 클라이언트에 전송하는 메소드
  - class ChatThread
    - 각 클라이언트와 연결 유지, 메시지 송수신 담당 스레드 클래스

메시지 설계

- 목적
  - 누가 보낸 메시지인지, 모든 사람에게 보내는 메시지인지, 특정인에게만 보내는 메시지 인지 구분 필요; 또한 현재 클라이언트가 로그인하려는 것인지, 아니면 로그아웃하려는 것인지도 알 수 있어야함
- 클라이언트 서버간 메시지 규격 (JSON)

번호	필드	설명
1	id	사용자 아이디
2	passwd	비밀번호
3	msg	전달 메시지
4	type	메시지 유형 (login, logout, msg)

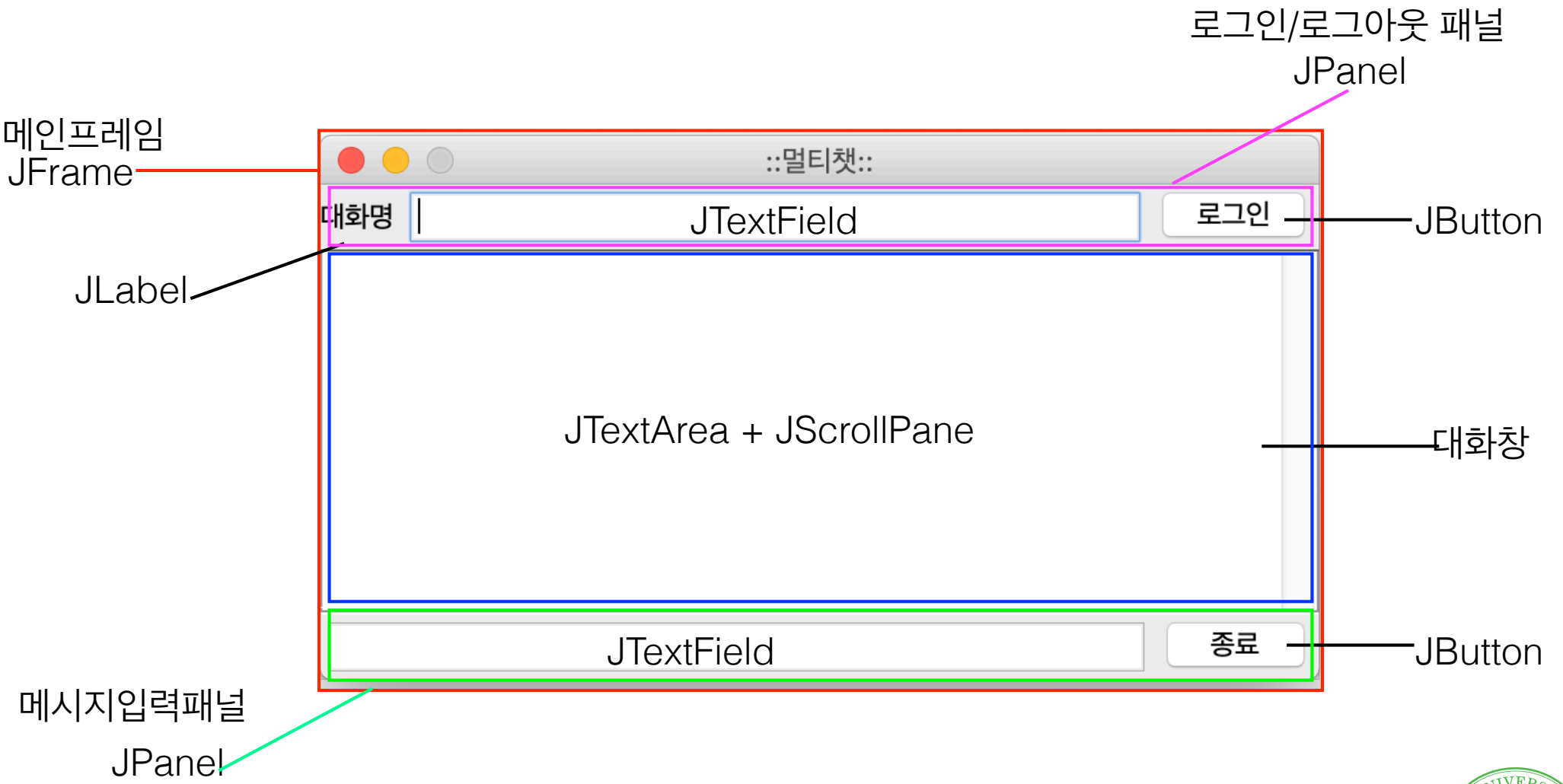
- 예
  - {"id" : "user1","passwd" : "1234","msg" : "hello","type" : "msg"}

Step 1. 클라이언트 UI 구현

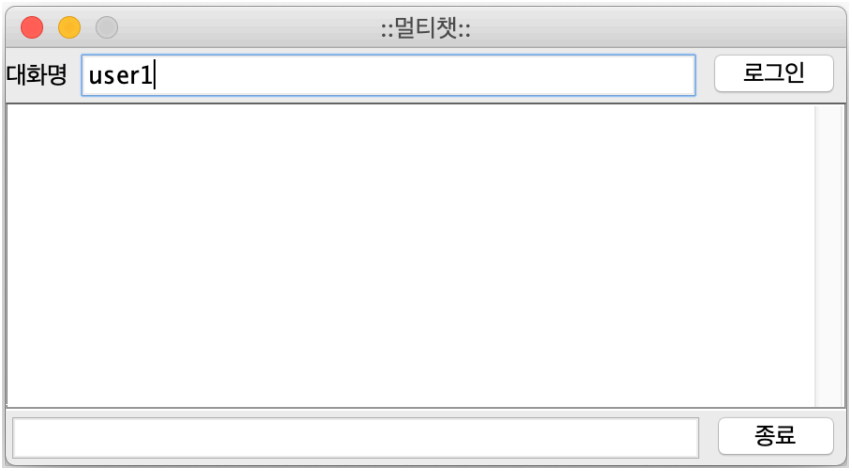
문제

- 다음 슬라이드에서 기술된 요구사항을 파악하고 MultiChatUI 클래스를 작성하시오
- ecampus에 업로드 된 파일 MultiChatUI.java에 작성하여 ecampus에 업로드
- URL 클래스 강의에서 DialogAuthenticator.java의 UI 부분을 참고하면 도움이 될 것임
- 생성자 MultiChatUI()와 addButtonActionListener()의 각 코멘트(//) 기술된 사항을 바로 아래에 추가하면 됨 (각 코멘트별 코드는 여러 줄일 수도 있고 한 줄인 것도 있음)

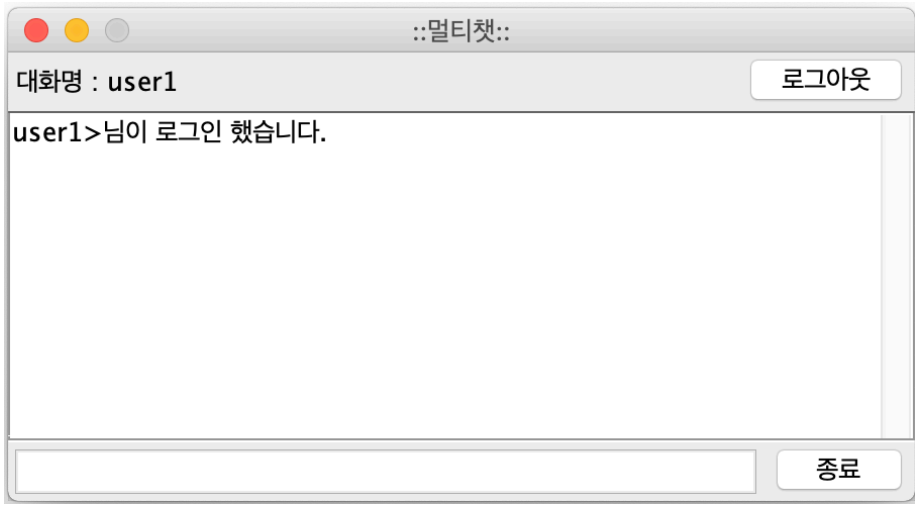
화면 레이아웃 스케치


로그인/로그아웃 패널
JPanel
메인프레임
JFrame
::멀티챗::
대화명
JTextField
로그인
JButton
JLabel
JTextArea + JScrollPane
대화창
JTextField
종료
JButton
메시지입력패널
JPanel

로그인 전/후 화면 비교



로그인 전



로그인 후

Step2. MultiChatController 클래스 구현

문제

- 다음 슬라이드에서 기술된 요구사항을 파악하고 MultiChatController 클래스를 작성하시오
- ecampus에 업로드 된 파일 MultiChatUI.java에 작성하여 ecampus에 업로드
- Step 1에서 완성한 MultiChatUI 클래스 필요
  - Message.java, MultiChatData.java도 필요하면 이 두 파일은 제공함 (여기에는 추가로 작업할 내용 없음)
- MultiChatController(), appMain(), connectServer(), run(), main()의 각 코멘트(//)에 기술된 사항을 바로 아래에 추가하면 됨 (각 코멘트별 코드는 여러 줄일 수도 있고 한 줄인 것도 있음)

appMain() 메소드 간략 설명

- 데이터 객체에 UI 객체를 추가하고, 버튼들의 이벤트 핸들러를 등록하는 addButtonActionListener() 메소드를 호출하면서 리스너 클래스를 익명의 내부 클래스로 만들어 전달
- MultiChatUI의 각 버튼에 대한 이벤트 코드도 appMain()에 작성
- 종료버튼
  - 프로그램 종료
- 로그인 버튼
  - 입력한 아이디를 가져와 outLabel에 출력하고 카드 레이아웃을 변경하여 로그인 상태(즉, 로그아웃이 보이게)로 전환
  - 서버에 연결
- 로그아웃
  - 출력 스트림을 이용하여 Message 객체를 생성한 후 JSON으로 변경하여 로그아웃 메시지를 전송
  - 그 다음 대화창을 비우고 로그인 패널로 전환
  - close() 메서드를 호출하여 입출력 스트림 닫기
- 엔터키 이벤트
  - JSON 규격 메시지를 생성하여 출력 스트림에 전달

다른 메소드 간략 설명

- connectServer()
  - 서버와 연결하고 입출력 스트림을 만든 후 메시지 수신에 필요한 스레드 생성
  - 서버와 연결==로그인 이므로 입출력 스트림을 생성한 후 바로 로그인 메시지 전달 필요
- run()
  - 서버에서 전송하는 메시지를 행 단위로 읽어와 JSON 메시지를 Message 객체로 변환
  - 데이터 클래스인 MultiChatData의 refresh() 메소드를 호출하여 변경할 메시지 전달
- main()
  - MultiChatController 객체 생성 후 appMain() 실행

Step3. MultiChatServer 클래스 구현

- 다음 슬라이드에서 기술된 요구사항을 파악하고 MultiChatServer 클래스를 작성하시오
- ecampus에 업로드 된 파일 MultiChatServer.java에 작성하여 ecampus에 업로드
- start(), msgSendAll(), run(), main()의 각 코멘트(//)에 기술된 사항을 바로 아래에 추가하면 됨 (각 코멘트별 코드는 여러 줄일 수도 있고 한 줄인 것도 있음)
- 각 이벤트(로그인, 로그아웃, 종료)시 채팅창에 출력되는 메시지는 203.252.148.148에 접속하여 확인 후 유사하게 대처

start() 메소드 간략 설명

- 서버소켓을 열고 무한 루프를 돌며 크라이언트의 연결을 기다림
- 새로운 클라이언트가 연결되면 스레드를 새로 생성하여 시작하고, 다시 다른 클라이언트의 연결을 기다림

ChatThread클래스 구현

- MultiChatServer의 내부에 정의된 클래스
- 기본적으로 Thread 클래스를 상속해서 구현
- run()메소드
  - 생성된 각 스레드에서 따로 동작
  - 클라이언트에서 전달하는 JSON 메세지를 읽어 와 Message 객체로 매핑한 후 Message 객체를 참조하여 메세지 유형에 따라 처리하는 구조
  - while() 블록에서 메세지를 행 단위로 읽어 문자열 변수로 받아옴
  - 문자열로 된 JSON 메세지를 프로그램에서 사용하기 쉽도록 Gson파서를 이용하여 Message클래스로 매핑
  - Message클래스의 getType() 메소드를 읽어 login, logout, msg에 따라 처리
- msgSendAll() 메소드
  - charlist ArrayList에서 ChatThread 클래스 인스턴스를 가져와 출력 스트림을 이용하여 메시지를 출력