채팅 프로그램 보고서

## 인공지능소프트웨어과 - 2403110263\_김태환

## 1. 프로젝트 개요

이 프로젝트는 클라이언트-서버 모델을 기반으로 한 실시간 채팅 애플리케이션 구현을 목표로 진행되었습니다. Java 언어를 사용하여 서버와 클라이언트 간 텍스트 메시지와 이미지 파일의 전송을 지원하는 프로그램을 개발했습니다. 주요 기능으로 다중 사용자 지원, 개인 메시지 전송, 이미지 업로드 및 다운로드가 포함됩니다.

## 2. 구현된 주요 기능

### 2.1 다중 사용자 채팅

- 서버는 여러 클라이언트를 동시에 수용할 수 있습니다.  
- 각각의 사용자는 고유한 닉네임을 설정해야 하며, 닉네임은 중복될 수 없습니다.  
- 모든 사용자는 다른 사용자가 보낸 메시지를 실시간으로 확인할 수 있습니다.

### 2.2 개인 메시지 전송

- 사용자는 `/to [닉네임] [메시지]` 명령어를 통해 특정 사용자에게 개인 메시지를 전송할 수 있습니다.  
- 닉네임이 잘못되었거나 상대방이 오프라인일 경우 오류 메시지가 표시됩니다.

### 2.3 닉네임 변경

- `/rename [새 닉네임]` 명령어를 통해 닉네임을 변경할 수 있습니다.  
- 새로운 닉네임은 고유해야 하며, 중복될 경우 오류 메시지가 표시됩니다.

### 2.4 이미지 파일 업로드 및 다운로드

- 업로드: 클라이언트는 `images/` 디렉터리의 파일을 `/img [파일명]` 명령어로 서버에 업로드할 수 있습니다.  
- 다운로드: `/get [이미지ID]` 명령어를 사용하여 서버에 저장된 이미지를 다운로드할 수 있습니다.  
- 서버는 업로드된 이미지를 고유한 이미지 ID로 관리하며, 다운로드 시 이를 활용합니다. 다운로드된 이미지는 클라이언트의 `images/` 디렉터리에 저장됩니다.

### 2.5 비동기 메시지 및 파일 전송

- 채팅 중에도 다른 사용자와의 메시지 및 파일 전송이 병렬적으로 처리됩니다.  
- 파일 다운로드 중에도 프로그램은 멈추지 않고 다른 작업을 진행할 수 있습니다.

## 3. 코드 구조

### 3.1 서버 코드

- `ChatServer` 클래스: 클라이언트 연결을 관리하며, 이미지 파일 저장 및 검색 기능을 제공합니다.  
- `ClientService` 클래스: 클라이언트와의 통신, 메시지 처리, 이미지 업로드/다운로드 요청을 처리합니다.

### 3.2 클라이언트 코드

- `ChatClient` 클래스: 사용자 입력을 기반으로 서버와 통신하며, 텍스트 메시지 및 파일 전송 기능을 제공합니다.

## 4. 디렉터리 구조

### 4.1 서버 디렉터리 구조

/채팅 서버 디렉터리/  
└── images/  
 └── 업로드된 이미지 파일

### 4.2 클라이언트 디렉터리 구조

/클라이언트 디렉터리/  
└── images/  
 └── 다운로드된 이미지 파일

## 5. 사용 방법

### 5.1 서버 실행

- `ChatServer` 클래스를 실행하여 서버를 시작합니다.  
- 서버는 기본적으로 `localhost:50005` 포트에서 클라이언트 연결을 대기합니다.(localhost에 ip입력 가능)

### 5.2 클라이언트 실행

- `ChatClient` 클래스를 실행하고 닉네임을 입력합니다.  
- 사용 가능한 명령어:  
 - 일반 메시지: `[메시지 내용]`  
 - 개인 메시지: `/to [닉네임] [메시지 내용]`  
 - 닉네임 변경: `/rename [새 닉네임]`  
 - 이미지 업로드: `/img [파일명]`  
 - 이미지 다운로드: `/get [이미지ID]`  
 - 종료: `/quit`

## 6. 테스트 결과

- 다중 사용자가 참여한 환경에서 메시지가 정상적으로 전달됨을 확인했습니다.  
- `/to`, `/rename` 명령어를 통해 개인 메시지 및 닉네임 변경 기능이 정상 동작함을 확인했습니다.  
- `images/` 디렉터리에 업로드된 파일이 서버에 저장되고, 다운로드 요청 시 올바르게 전송 및 저장되는 것을 확인했습니다.  
- 비동기 파일 전송과 채팅 기능이 문제없이 작동하며, 프로그램이 멈추지 않고 연속적으로 사용 가능함을 확인했습니다.

## 7. 결론 및 개선점

본 프로젝트는 텍스트 및 이미지 데이터를 실시간으로 주고받을 수 있는 클라이언트-서버 기반 채팅 애플리케이션을 성공적으로 구현한 사례입니다. 다중 사용자 지원과 파일 전송 기능은 실제 네트워크 애플리케이션의 실용성을 보여줍니다. 향후 추가적으로 보안 및 사용자 인터페이스 개선을 통해 더욱 발전된 프로그램을 개발할 수 있을 것입니다.