

디자인 및 구현 문서

2020036371 조재혁

I. 디자인 설계

본 문서는 구현을 시작하기 전 설계하면서 생각했던 과정을 기술한다.

우선 P2P의 정의대로 각 Peer가 server와 client의 역할을 동시에 수행해야하기 때문에 class는 하나로만 구현해야 된다고 판단하였다. 그리고 그 내에 2개의 thread를 만들어서 각각 sending(client)의 역할과 receiving(server)의 역할을 수행하도록 디자인 하였다.

단순히 메시지를 주고 받기만 하면 살짝 아쉬운 면이 있어서 기본적인 기능들을 추가하고 싶었다. 본인의 실력으로 구현하는 데 크게 어렵지 않은 Peer의 입장 메시지, 퇴장 메시지, 그리고 본인의 메시지 숨김의 기능을 추가로 구현하기로 결정하였다. 하지만 특정 메시지를 출력하는 일은 server thread에서 해야 하는데, server package의 datagram packet 클래스는 메시지, 메시지 길이, IP 주소, port 번호 4개만 전송하기 때문에 메시지를 보낸 Peer을 구별하는 방법을 모색해야만 했다. 고민해본 결과, Packet을 보낼 때 가장 앞부분에 Peer의 이름과 메시지의 종류를 같이 보내는 방식을 쓰면 되겠다고 생각이 들었다. 우선 모든 Packet을 크게 3개의 파트로 분리하였다. Join, Exit등의 command을 담당하는 파트, Peer의 네임을 저장하는 파트, 그리고 커맨드가 아닌 메시지의 경우에 사용되는 메시지 파트이다. 이렇게 Packet을 포장해서 보내면 server쪽에서 parsing하여 command에 맞게 메시지를 출력하는 방식으로 진행하게 되었다.

II. 구현 과정

본 문서는 기본적인 구현 과정은 생략하고 어려움을 느꼈던 부분이나 명세 외로 개인적으로 추가한 기능들에 대한 내용을 다룬다.

가장 먼저 일반 메시지와 #으로 시작하는 command을 구분하는 작업을 하였다.

Join과 Exit외의 다른 기능은 따로 만들지 않았으므로 제일 먼저 입력 받는 string은 반드시 #JOIN + room name + peer name의 구조를 가지는 것으로 전제하였다. #JOIN 외의 커맨드나 일반 메시지가 들어오는 경우 오류 메시지를 출력하고 프로그램을 종료하는 정도로 넘어갔다. 다음은 SHA-256를 사용한 IP address 변환인데, JAVA에서는 unsigned int 등의 자료형을 지원하지 않는다는 사실을 알았다. 그래서 string을 일단 digest한 후에 끝 2bytes끼리 묶어서 0xFF와 & 연산을 수행하여 int로 표시되게 하되, 자료형은 long을 사용하여 overflow를 방지하였다. 여기 까지가 2개의 thread를 수행하기 전 기본적인 setting 과정이다.

Client 구현 방법부터 보면, 디자인 한대로 message를 3개의 part로 분리하는 것에 많은 신경을 썼다. JAVA의 split 함수를 사용하면 특정 char로 string을 분리시킬 수 있기 때문에 @를 각 파트를 나누는 키워드로 설정했다. (#은 command의 시작 키워드로 이미 설정되었기 때문에 다른 아무 특수 문자 중 임의로 선택하였다.) 사용자에게서 메시지가 들어오면 command인지 message인지 먼저 구분한 후, command인 경우 message part을 생략하고 peer name + command 구조로 packet을 생성하고, message인 경우 peer name + "MSG"(message 또한 MSG라는 커맨드를 할당했다.) + message의 구조로 packet을 생성했다. 이 과정에서 chunk (512byte)의 범위를 벗어나는 경우가 생길 수 있는데, 이는 반복문으로 string을 계속 자르는데 모든 잘린 string의 앞부분에 peer name과 command를 추가하면서 앞서 기술했던 3 parted packet의 구조를 유지했다. (packet을 새로 보낼 때마다 server 입장에서 2번째로 온 packet이 앞서 도착한 packet과 같은 peer에서 온 것인지 알 수가 없기 때문이다.)

마지막으로 server 구현이 있는데, client에서 대부분의 일을 처리하기 때문에 크게 어려운 점은 없었다. 수신한 packet을 @기준으로 split한 후에 각 command에 맞는 일을 처리하고, 모든 출력문에 있어서 packet에 담긴 peer name과 본인의 peer name이 같은 경우 무조건 출력문이 나오지 않도록 제한했다. 한가지 추가한 점은, EXIT command의 packet 주인이 본인인 경우에는 프로그램을 종료하도록 설정하였다.

III. 실행 방법

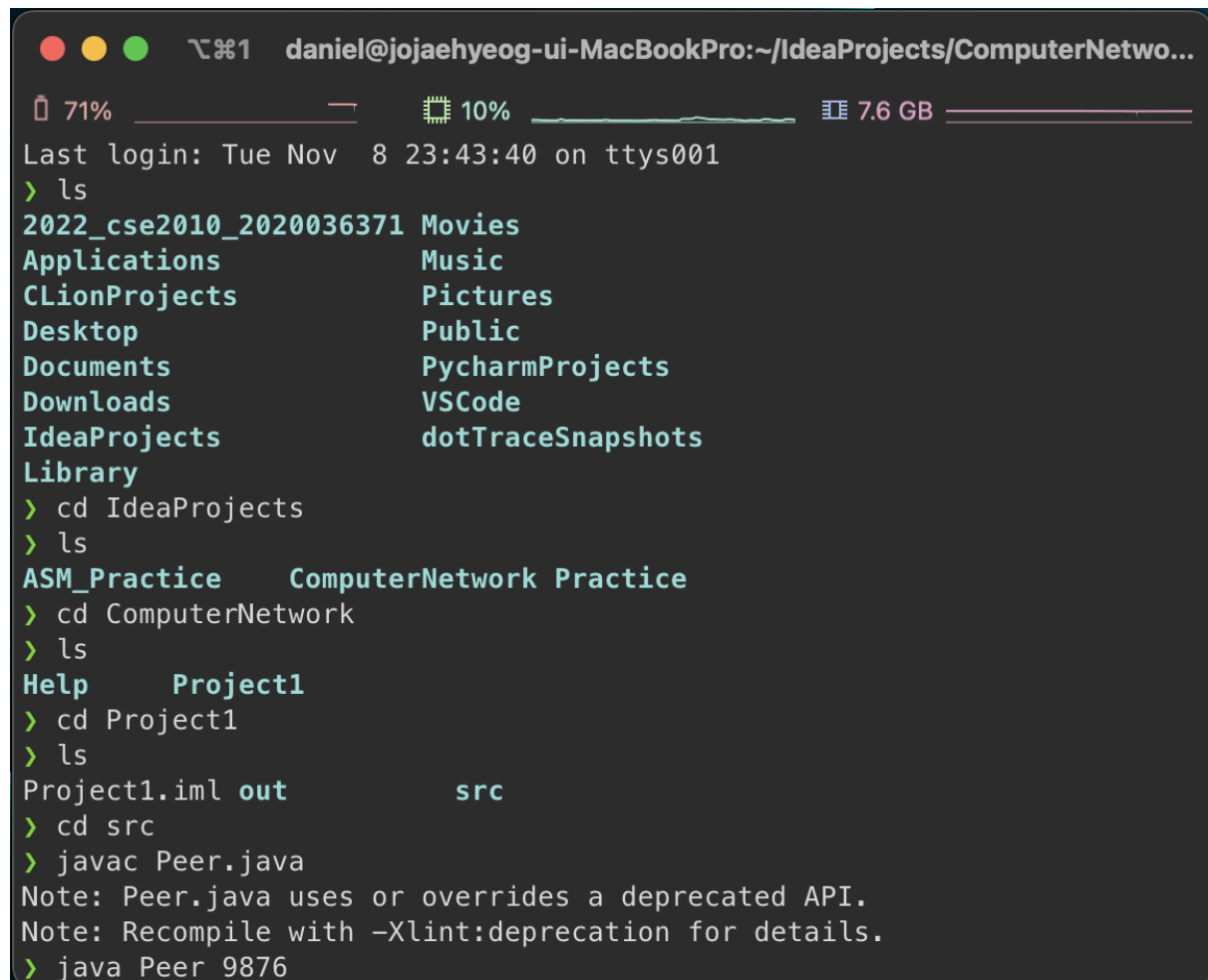
본 문서는 class 파일을 컴파일 및 실행하는 방법과 예시 화면을 제시한다. 본인은 mac 환경에서 작업하여 terminal로 컴파일 하는 과정을 거쳤으며, windows 환경에서도 cmd로 작동하는지 확인한 결과 문제 없었다.

- 다음 단계에 따라 실행 -

1. terminal (cmd) 실행 후 java 파일이 있는 경로로 이동
2. 컴파일: javac Peer.java
3. 실행: java Peer port (port: 임의의 port number)
4. 추가로 peer을 생성하고 싶은 경우 위의 단계 반복 (2단계 skip)

아래에 예시 컴파일 화면과 실행 화면을 첨부 하였다.

예시 컴파일 화면 (최초 Peer)



```
daniel@jojaehyeog-ui-MacBookPro:~/IdeaProjects/ComputerNetwo...
71% 10% 7.6 GB
Last login: Tue Nov  8 23:43:40 on ttys001
> ls
2022_cse2010_2020036371  Movies
Applications             Music
CLionProjects            Pictures
Desktop                  Public
Documents                 PycharmProjects
Downloads                 VSCode
IdeaProjects              dotTraceSnapshots
Library
> cd IdeaProjects
> ls
ASM_Practice  ComputerNetwork  Practice
> cd ComputerNetwork
> ls
Help  Project1
> cd Project1
> ls
Project1.iml  out  src
> cd src
> javac Peer.java
Note: Peer.java uses or overrides a deprecated API.
Note: Recompile with -Xlint:deprecation for details.
> java Peer 9876
```

예시 실행 화면 (Peer 4개)

The image displays four terminal windows from a macOS environment, each running a Java application. The windows are titled "daniel@jojaehyeog-ui-MacBookPro:~/IdeaProjects/C..." and show the output of a Java program. The program appears to be a simple chat or status-sharing application involving several peers (PeerA, PeerB, PeerC, PeerD) and a central server (Peer). The output shows messages like "PeerB has joined the room", "PeerC has joined the room", "PeerD has joined the room", and various status updates and greetings.

```

Terminal 1:
> java Peer 9876
#JOIN cnet PeerA
PeerB has joined the room
PeerC has joined the room
PeerD has joined the room
Hi
I love computer network class
PeerB: same here
PeerC: have a nice day :)
PeerD: you too
PeerD: i should go now
PeerD: bye
PeerD has left the room
PeerC: me too
PeerC has left the room
bye PeerB
#EXIT
~/I/C/P/src >

Terminal 2:
> java Peer 9876
#JOIN cnet PeerB
PeerC has joined the room
PeerD has joined the room
PeerA: Hi
PeerA: I love computer network class
same here
PeerC: have a nice day :)
PeerD: you too
PeerD: i should go now
PeerD: bye
PeerD has left the room
PeerC: me too
PeerC has left the room
PeerA: bye PeerB
PeerA has left the room
#EXIT
~/I/C/P/src >

Terminal 3:
> java Peer 9876
#JOIN cnet PeerC
PeerD has joined the room
PeerA: Hi
PeerA: I love computer network class
PeerB: same here
have a nice day :)
PeerD: you too
PeerD: i should go now
PeerD: bye
PeerD has left the room
me too
#EXIT
~/I/C/P/src >

Terminal 4:
> ls
Project1.iml out src
> cd src
> java Peer 9876
#JOIN cnet PeerD
PeerA: Hi
PeerA: I love computer network class
PeerB: same here
PeerC: have a nice day :)
you too
i should go now
bye
#EXIT
~/I/C/P/src >

```