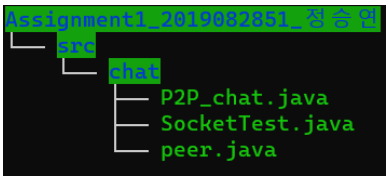
Computer network assignment1

2019082851
정승연

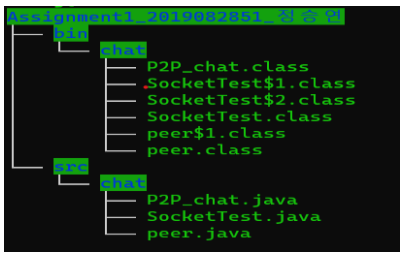
A. How to compile assignment1

1. At the root directory, use `javac --source-path src -d bin src/chat/*.java` command to compile source file in bin directory.
2. At the root directory, use `java -cp bin chat.P2P_chat <port value>` command to execute program.

In addition, this is tree for initial root directory.



After execute A.1 command, this is tree structure for root directory.



This is example about compile and run program.

```
chung@DESKTOP-DCBVVD5:/mnt/c/Users/wjdac/Desktop/Assignment1_2019082851_정승연$ javac --source-path src -d bin src/chat/*.java
chung@DESKTOP-DCBVVD5:/mnt/c/Users/wjdac/Desktop/Assignment1_2019082851_정승연$ java -cp bin chat.P2P_chat 3000
1
To join chat room input like this: #JOIN chat_room_name user_name
#JOIN chat_room A
225.70.0.25
# chat to others
#EXIT
exit chat room
```

Yellow commands are what I type in.

important <port value> should be higher than 1024.

In the case of Linux, ports less than 1024 are called well-known ports and can only be run with root privileges.

important to communicate with others, users should share “same port value” and “same chat room name”

B. Example of running programs

```
chung@DESKTOP-DCBVVD5:/mnt/c/Users/wjdac/Desktop/Assignment1_2019082851_정승연$ java -cp bin chat.P2P_chat 1025
1
To join chat room input like this: #JOIN chat_room_name user_name
#JOIN chat_room peerA
225.70.0.25
# chat to others
IS ANYONE HERE??
peerA : IS ANYONE HERE??
```

```
chung@DESKTOP-DCBVVD5:/mnt/c/Users/wjdac/Desktop/Assignment1_2019082851_정승연$ java -cp bin chat.P2P_chat 1025
1
To join chat room input like this: #JOIN chat_room_name user_name
^[[A^C^[[Achung@DESKTOP-DCBVVD5:/mnt/c/Users/wjdac/Desktop/Assignment1_2019082851_정승연$ java -cp bin chat.P2P_chat 1025
1
To join chat room input like this: #JOIN chat_room_name user_name
```

Before right peer joining, message from left peer(peerA) cannot reach to right peer.

```
chung@DESKTOP-DCBVVD5:/mnt/c/Users/wjdac/Desktop/Assignment1_2019082851_정승연$ java -cp bin chat.P2P_chat 1025
1
To join chat room input like this: #JOIN chat_room_name user_name
#JOIN chat_room peerA
225.70.0.25
# chat to others
IS ANYONE HERE??
peerA : IS ANYONE HERE??

peerB : Is anyone here?
```

```
chung@DESKTOP-DCBVVD5:/mnt/c/Users/wjdac/Desktop/Assignment1_2019082851_정승연$ java -cp bin chat.P2P_chat 1025
1
To join chat room input like this: #JOIN chat_room_name user_name
^[[A^C^[[Achung@DESKTOP-DCBVVD5:/mnt/c/Users/wjdac/Desktop/Assignment1_2019082851_정승연$ java -cp bin chat.P2P_chat 1025
1
To join chat room input like this: #JOIN chat_room_name user_name
#JOIN chat_room peerB
225.70.0.25
# chat to others
Is anyone here?
peerB : Is anyone here?
```

After two peers are joined in chat room, message from peerB reach to peerA.

```
chung@DESKTOP-DCBVVD5:/mnt/c/Users/wjdac/Desktop/Assignment1_2019082851_정승연$ java -cp bin chat.P2P_chat 1025
1
To join chat room input like this: #JOIN chat_room_name user_name
#JOIN chat_room peerA
225.70.0.25
# chat to others
IS ANYONE HERE??
peerA : IS ANYONE HERE??

peerB : Is anyone here?

wow! hi. My name is peerA
peerA : wow! hi. My name is peerA

peerB : Hello my name is peerB.

It time to say goodbye. Bye Bye.
peerA : It time to say goodbye. Bye Bye.

#EXIT
exit chat room
```

```
To join chat room input like this: #JOIN chat_room_name user_name
^[[A^C^[[Achung@DESKTOP-DCBVVD5:/mnt/c/Users/wjdac/Desktop/Assignment1_2019082851_정승연$ java -cp bin chat.P2P_chat 1025
1
To join chat room input like this: #JOIN chat_room_name user_name
#JOIN chat_room peerB
225.70.0.25
# chat to others
Is anyone here?
peerB : Is anyone here?

peerA : wow! hi. My name is peerA

Hello my name is peerB.
peerB : Hello my name is peerB.

peerA : It time to say goodbye. Bye Bye.

Good Bye.
peerB : Good Bye.

#EXIT
exit chat room
```

This is simple scenario for chatting.

Because B is not yet joined, message"IS ANYONE HERE" from A is not reached to B.

Because A is quit, message"Good Bye" from B is not reached to A

C. Design of assignment1

I design assignment 1 with SocketTest.java.
Each peer can send message to other peers and should be receive message from others.
So, I divide role of each peer to Sender role and Server role.
Sender role is executed by startSender() function.
Server role is executed by startServer() function.

Sender role:
To send message, we need InetAddress for multicast UDP.
If user inputs message string, we translate it to byte array type.
Suppose we know the port number that is shared between users.
By using byte array and length of array, InetAddress and port number,
we make packet and send it by using socket.

Server role:
To receive message, we need MulticastSocket!
First need InetAddress for multicast UDP. By using InetAddress, we can make InetSocketAddress.
Also, to safe communication, we can create NetworkInterface.
Then we create MulticastSocket object with portnumber.
Last, make sockect object to join with InetSocketAddress and NetworkInterface.
This is initial process for making multicast socket available.
And then, create DatagramPacket which is used for buffer.
With while loop, we can get message with socket.

As summary, these are my simple design.
First, divide role: sender role and receiver role.
(Two role will be executed independently by using thread.)
Second, setup socket and packet with given information.
Third, send or receive message with while loop.

D. Implementation of assignment1

To help P2P_chat.java (which is main java source file), we need peer.java file which define sender and receiver role function.

THESE ARE FUNCTIONS IN peer.java.

1. public void startSender(InetAddress target_address, int portNumber, String user_name)

If user want to join chat, we execute this function. This function is for send message. As you see, if "#: used as the first character of message, we will send error message that you cannot use like that: Except "#EXIT", with "#EXIT" we will shut down whole program.

If user type message, this function automatically prepends "username" to the message. And as we design, function sends message to other users.

In addition, after sending message, something wrong about socket buffer. For example, I send "hello" and "hi" sequentially, another user receive message as "hello" and "hillo". To handle this problem, I additionally send packet with blank with same length we send, right after we send message. As a temporary workaround, I was able to solve the problem.

2. public void startServer(InetAddress target_address, int portNumber, String user_name)

Before executing startSender function, we execute this function.

This function is for receive message. To make this function independent with main execution flow, we create new thread. And then, this function will receive message as we design. In addition, I add filter that, if message sender is me, startServer function will not display that message.

In P2P_chat.java, main function exits.
As argument, main function will get port number.
If, argument is not correct format, function sends error message and exits.
If function get proper port number, function will tell user proper join instruction format.
If user type “#EXIT”, program exits.
If user type proper join instruction, we get InetAddress from chat room name by using SHA-512 hashing.
By using InetAddress, port number and username, we call peer class startServer function and startSender function to get message and to receive message.