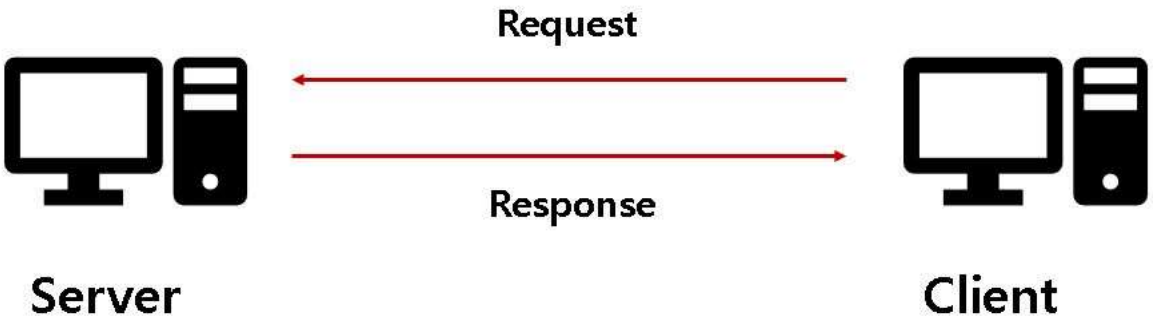
Chapter 1

자바 network을 사용한 채팅 프로그램

서버와 클라이언트



채팅프로그램을 만들기 전 알아 두어야 할 서버 - 클라이언트 의 관계

서버 – 사용자가 채팅 내용을 치면 결과를 다른 사용자 한테 전달해주는 전달자

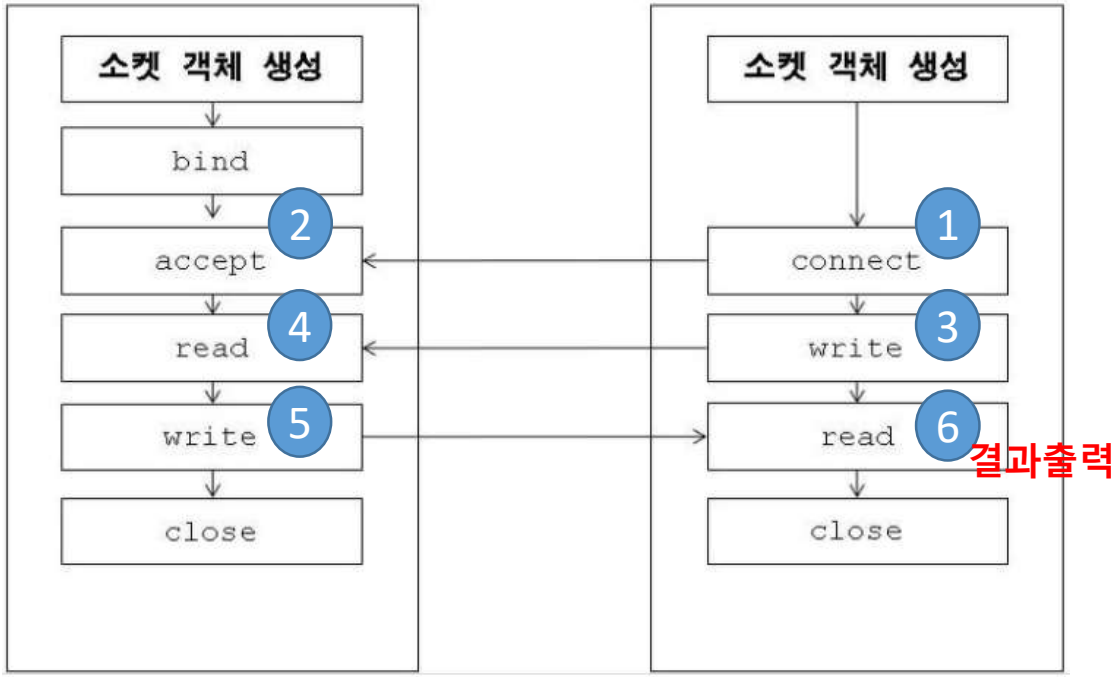
클라이언트 – 사용자가 사용하는 환경

TCP/IP(Transmission Control Protocol) - 다른 컴퓨터와 통신을 하기 위한 통신 규약
컴퓨터 프로그램간 데이터를 안정적으로 전달할 수 있게 해줍니다.

Socket - 컴퓨터 네트워크를 경유하는 프로세스 간 통신의 종착점입니다. 쉽게 생각하면 톨게이트로 생각하면 됩니다.

즉 채팅내용(데이터)는 클라이언트 -> 회선 -> 서버의 소켓에 도달

통신 절차



서버 클라이언트

1. 클라이언트에서 connect 연결 요청
2. 대기중인 서버가 accept로 연결 수락
3. 클라이언트에서 글을 쓰고 (outputStream 으로 전송)
4. 서버에서 InputStream을 통해 글을 받음
5. 서버에서 outputStream을 통해 글을 클라이언트로 전송
6. 각 클라이언트에서 InputStream을 통해 글을 받음

사용할 클래스

사용할 클래스

Socket	Client Socket을 구현하는 클래스. 두 기기 사이에 종단점에 위치하여 데이터를 교환한다.
Server Socket	Client에서 들어오는 요청을 기다리는 Server Socket을 구현하는 클래스.
InputStreamReader	byte 단위로 들어오는 InputStream에 대하여 char 단위로 읽고쓰는 Reader 인터페이스를 제공해 주는 보조스트림
BufferedReader	Buffer를 통해 char 단위로 읽고 쓸 수 있게 해주는 보조 스트림
OutputStreamWriter	byte 단위로 쓰는 OutputStream에 대해 특정 인코딩의 char 단위로 읽고 쓰게 해주는 보조 스트림
PrintWriter	text로 대표되는 객체들을 출력할 수 있게 해주는 보조 스트림

Server측 코드

```
public class MainServer {


	public static ArrayList<PrintWriter> list = new ArrayList<>();

	public static void main(String[] args) {
		try {
			ServerSocket serverSocket = new ServerSocket(8383);

			while(true) {
				System.out.println("-----연결대기-----");
				Socket socket = serverSocket.accept();
				System.out.println("------연결됨------");

				ClientManager client = new ClientManager(socket);

				list.add(new PrintWriter(socket.getOutputStream() ));

				client.start();

			}

		} catch (Exception e) {
			e.printStackTrace();
		}

	}
}
```

서버의 소켓을 열어둡니다. 8383포트번호

요청이 들어오면 클라이언트 관리 클래스를 생성하고,
연결된 socket클래스를 전달합니다.
서버에 접속하는 소켓의 output을 리스트에 저장한다.

클라이언트의 연결요청을 대기하는 메서드
즉, 클라이언트가 해당 포트번호로 연결요청이 들어오면 accept는 대기명령을 해제하고
클라이언트와 연결시켜주는 Socket클래스를 생성해서 반환합니다.

ClientManager클래스

```
public class ClientManager extends Thread{

	private Socket socket;
	private String id;

	public ClientManager(Socket socket) {
		this.socket = socket;
	}
	@Override
	public void run() {

	try {
		BufferedReader bf = new BufferedReader(new InputStreamReader(socket.getInputStream(), "UTF-8"));

		while(true) {
			String message = bf.readLine();

			if(message == null) {
				System.out.println(id + "님이 퇴장했습니다");
				for(int i = 0; i < MainServer.list.size(); i++) {
					PrintWriter pw = MainServer.list.get(i);
					pw.println(id + "님이 퇴장했습니다");
					pw.flush();
				}
			break;
			} //end if

			String[] split = message.split(":");
			if(split.length == 2 && split[0].equals("ID")) {
				this.id = split[1];
				System.out.println(this.id + "님이 입장하였습니다");
				for(int i = 0; i < MainServer.list.size(); i++) {
					PrintWriter pw = MainServer.list.get(i);
					pw.println(id + "님이 입장하였습니다");
					pw.flush();
				}
				continue;
			}
```

소켓으로 부터 전달된 입력 스트림을 받는다.

Message가 없다는 것은 연결이 끊겼다는 것을 의미하고
퇴장메시지를 list안에 있는 모든 클라이언트 한테 전달해 주는 부분

구분자를 통해 아이디 부분을 추출하고 모든 클라이언트에
입장 메시지를 전달하는 부분

ClientManager클래스

```
            for(int i = 0; i < MainServer.list.size(); i++) {
                  PrintWriter pw = MainServer.list.get(i);
                  pw.println(this.id + ">" + message);
                  pw.flush();
            }
      } //end while

      MainServer.list.remove(new PrintWriter(socket.getOutputStream() ) );

      } catch (Exception e) {
            System.out.println(this.id + "님 연결이 끊겼습니다");
      }
   }
}
```

하나의 클라이언트에서 넘어온 메시지를 접속되어 있는 모든 클라이언트에 전달하는 부분

While문을 탈출하면, 연결의 종료를 의미하므로 리스트에서 해당 클라이언트를 지운다

Client측 코드

```
public class MainClient {

	public static String userID;

	public static void main(String[] args) {

		try {
			Socket clientSocket = new Socket("연결아이피", 8383);

			Receive receive = new Receive(clientSocket);

			Sender sender = new Sender(clientSocket);

			receive.start();
			sender.start();

			} catch (Exception e) {
			System.out.println("클라이언트 main에러");
			e.printStackTrace();
		}

	}
}
```

서버로 연결할 소켓을 생성하고, 연결할 IP주소, 포트번호를 할당합니다.
클라이언트에서 메시지를 받는 클래스 Receive
클라이언트에서 메시지를 보내는 클래스 Sender

Receive클래스

```
public class Receive extends Thread {

	private Socket socket;

	public Receive(Socket socket) {
		this.socket = socket;
	}

	@Override
	public void run() {

		try {
			BufferedReader bf = new BufferedReader(new InputStreamReader(socket.getInputStream(), "UTF-8"));

			while(true) {
				String message = bf.readLine();

				String[] split = message.split(">");
				if(split.length == 2 && split[0].equals(MainClient.userID) ) {
					continue;
				}

				System.out.println(message);
			}

		} catch (Exception e) {
			System.out.println("클라이언트 receive에러");
			e.printStackTrace();
		}

	}
}
```

메시지를 받아 왔을 때, 자신이라면 콘솔창에
ID부분을 지워주는 부분
서버로 부터 넘어온 messag를 > 기반으로 자르고
Id부분이 본인의 id와 같다면 아이디 부분 출력은 넘겨준다.

Sender클래스

```
public class Sender extends Thread{

	private Socket socket;

	public Sender(Socket socket) {
		this.socket = socket;
	}
	@Override
	public void run() {
		try {
			BufferedReader bf = new BufferedReader(new InputStreamReader(System.in, "UTF-8"));
			PrintWriter out = new PrintWriter(socket.getOutputStream());

			System.out.print("닉네임:");
			while(true) {
				String id = bf.readLine();
				if(id == null || Pattern.compile(":").matcher(id).find() )
					System.out.println("사용할 수 없는 아이디 입니다");
					System.out.print("닉네임:");
					continue;
				} else {
					MainClient.userID = id;
					break;
				}
			}

			out.println("ID:" + MainClient.userID);
			out.flush();
```

사용자에게 최초 사용할 아이디를 입력받는 부분
정규표현식 :을 찾아서 :이 포함되 있다면 다시 입력 받는다.
아이디를 입력받고 멤버변수 ID를 저장해 준다.

연결되어 있는 서버측 소켓에 ID:입력한아이디 형식으로 데이터를 전달하는 부분

Sender클래스

```
                while(true) {
                        String message = bf.readLine();

                        if(message.equals("exit")) {
                                break;
                        }

                        out.println(message);
                        out.flush();
                }

                out.close();
                bf.close();
                socket.close();

        } catch (Exception e) {
                System.out.println("클라이언트 sender에러");
                e.printStackTrace();
        }
    }
}
```

사용자에게 메시지를 입력받는 부분
Exit 를 받으면 탈출하여 접속을 종료하게 되고
입력받은 메시지를 서버로 전달

Chapter 24

수고하셨습니다