

18장- 네트워크 프로그래밍



Socket 데이터 통신



네트워크 기초

네트워크

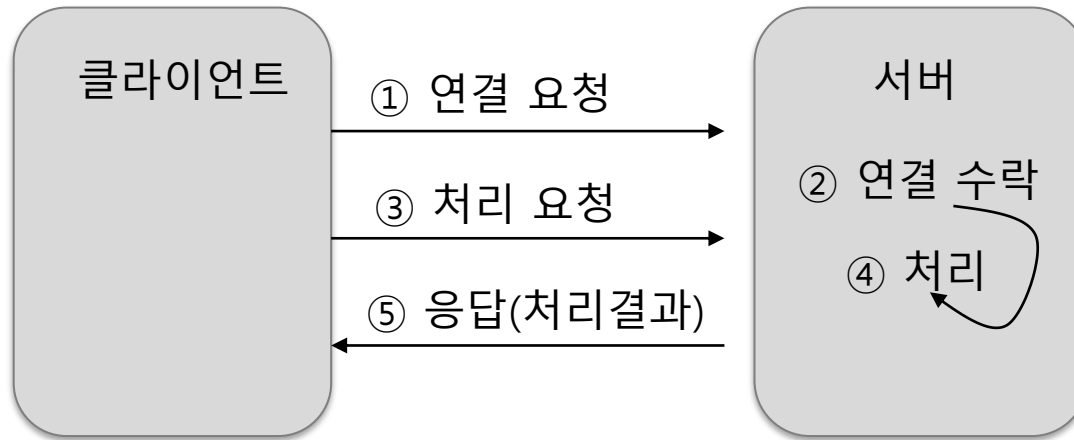
- 홈 네트워크 : 컴퓨터가 방마다 있고, 이들 컴퓨터를 유.무선 등의 통신회선으로 연결(LAN)
- 지역 네트워크 : 회사, 건물, 특정 영역에 존재하는 컴퓨터를 통신회선으로 연결한 것(MAN)
- 인터넷 : 지역네트워크를 통신 회선으로 연결한 것(WAN)

서버와 클라이언트

- **서버(Server)** : 서비스를 제공하는 프로그램
 - 웹서버, FTP서버, DBMS 서버, 메신저 서버
 - 클라이언트의 연결을 수락하고, 요청 내용을 처리한 후 응답을 보내는 역할
- **클라이언트(Client)** : 서비스를 받는 프로그램
 - 웹브라우저, FTP클라이언트, 메신저
 - 네트워크 데이터를 필요로 하는 모든 애플리케이션이 해당(모바일 App포함)



서버 / 클라이언트



- ▶ 클라이언트/서버(C/S: client/server) : 한 개의 서버와 다수의 클라이언트로 구성
- ▶ P2P(Peer to Peer) : 두 개의 프로그램이 서버인 동시에 클라이언트 역할을 함
먼저 접속을 시동한 컴퓨터가 클라이언트가 된다. (1:1 채팅과, 파일 공유 프로그램)



IP 주소와 포트(Port)

IP 주소

- IP(Internet Protocol) 주소 : 컴퓨터의 고유한 주소 - IPv4
- xxx.xxx.xxx.xxx(xxx는 0~255 사이의 정수)
- 네트워크 어댑터(Lan 카드) 마다 할당 - 유선/무선 랜카드

명령 프롬프트(cmd)

C:W>ipconfig 입력

```
C:\Users\김기웅>ipconfig
```

```
Windows IP 구성
```

```
이더넷 어댑터 이더넷:
```

```
미디어 상태 . . . . . : 미디어 연결 끊김
```

```
연결별 DNS 접미사. . . . . :
```

```
무선 LAN 어댑터 Wi-Fi:
```

```
연결별 DNS 접미사. . . . . :
```

```
링크-로컬 IPv6 주소 . . . . . : fe80::fdb4:956c:171d:19c7%14
```

```
IPv4 주소 . . . . . : 192.168.0.6
```

```
서브넷 마스크 . . . . . : 255.255.255.0
```

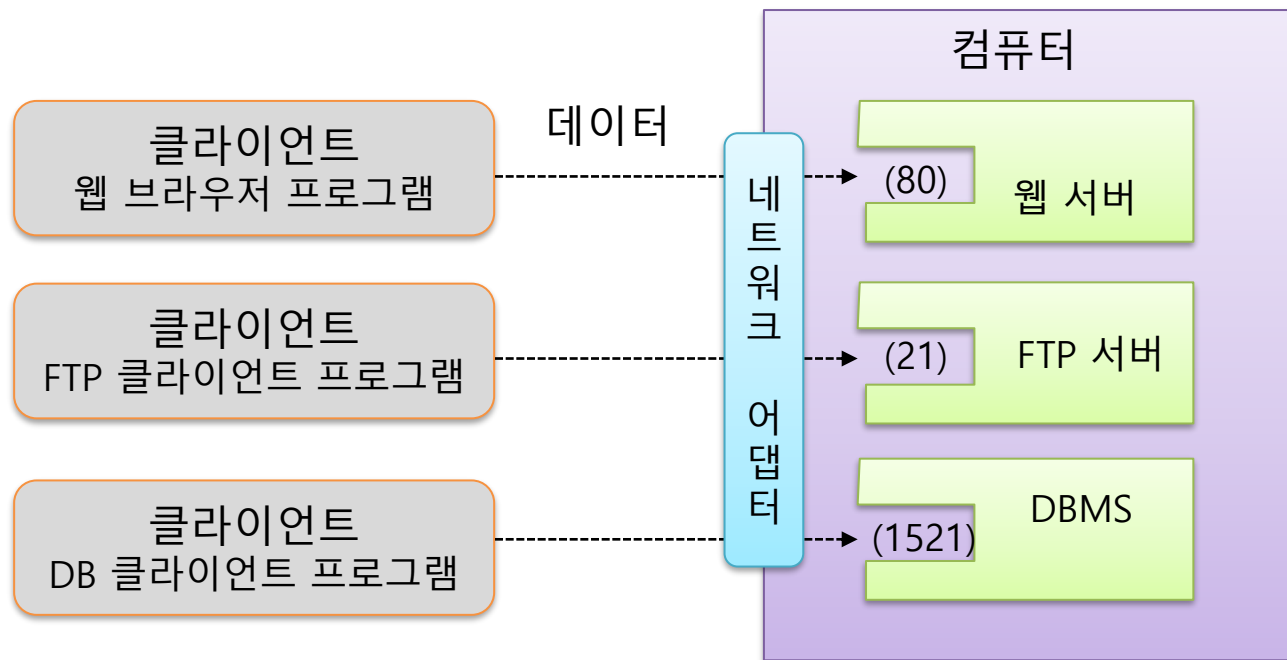
```
기본 게이트웨이 . . . . . : 192.168.0.1
```



IP 주소와 포트(Port)

포트(Port)

- 같은 컴퓨터 내에서 프로그램을 식별하는 번호
- 클라이언트는 서버 연결 요청시 IP 주소와 Port를 같이 제공
- 포트번호의 전체 범위 : 0 ~ 65535 범위의 값을 가짐



IP 주소와 포트(Port)

- Port는 운영체제가 관리하는 서버 프로그램의 연결 번호이다. 서버는 시작할 때 Port 번호에 바인딩한다. 예를 들어 서버는 80번, DBMS(Oracle)는 1521번으로 바인딩 할 수 있음.
- 클라이언트도 서버에서 보낸 정보를 받기 위해서는 Port 번호가 필요한데, 서버와 같이 고정적인 Port 번호에 바인딩하는 것이 아니라 운영체제가 자동으로 부여하는 번호를 사용한다.

클래스	범위	용 도
잘 알려진 포트번호 (Well Know Port Numbers)	0~1023	국제인터넷 주소관리기구(CANN)가 특정 애플리케이션용으로 미리 예약한 Port
예약된 포트번호 (Registered Port Numbers)	1024~49151	회사에서 등록해서 사용할 수 있는 Port
동적인 또는 개인 포트번호 (Dynamic Or Private PortNumbers)	49152~65535	운영체제가 부여하는 동적 Port 또는 개인 목적으로 사용할 수 있는 Port



IP 주소와 포트(Port)

IP 주소 얻어 오기

- **Java.net.InetAddress**
- IP 주소를 표현한 클래스
- 로컬 컴퓨터의 IP 주소 뿐만 아니라 도메인 이름을 DNS에서 검색한 후 IP 주소를 가져오는 기능 제공 (예. www.naver.com -> IP 주소)

1. 로컬 컴퓨터에서 얻기

```
InetAddress ia = InetAddress.getLocalHost();
```

2. 도메인 이름으로 얻기

```
InetAddress ia = InetAddress.getByName(String host)
```

```
InetAddress[ ] iaArr = InetAddress.getAllByName(String host)
```



IP 주소와 포트(Port)

IP 주소 얻어 오기

```
package inetaddress;
import java.net.InetAddress;
import java.net.UnknownHostException;

public class InetAddressEx {

    public static void main(String[] args) {

        try {
            //내 컴퓨터
            InetAddress local = InetAddress.getLocalHost();
            System.out.println("내 컴퓨터 IP 주소 : " + local.getHostAddress());

            //서버 컴퓨터
            //InetAddress server = InetAddress.getByName("www.naver.com");
            //System.out.println(server);
            InetAddress[] servers = InetAddress.getAllByName("www.naver.com");
            for(InetAddress remote : servers)
                System.out.println(remote);
        } catch (UnknownHostException e) {
            e.printStackTrace();
        }
    }
}
```

내 컴퓨터 IP 주소 : 192.168.0.6

네이버 컴퓨터 IP 주소 : www.naver.com/223.130.195.200

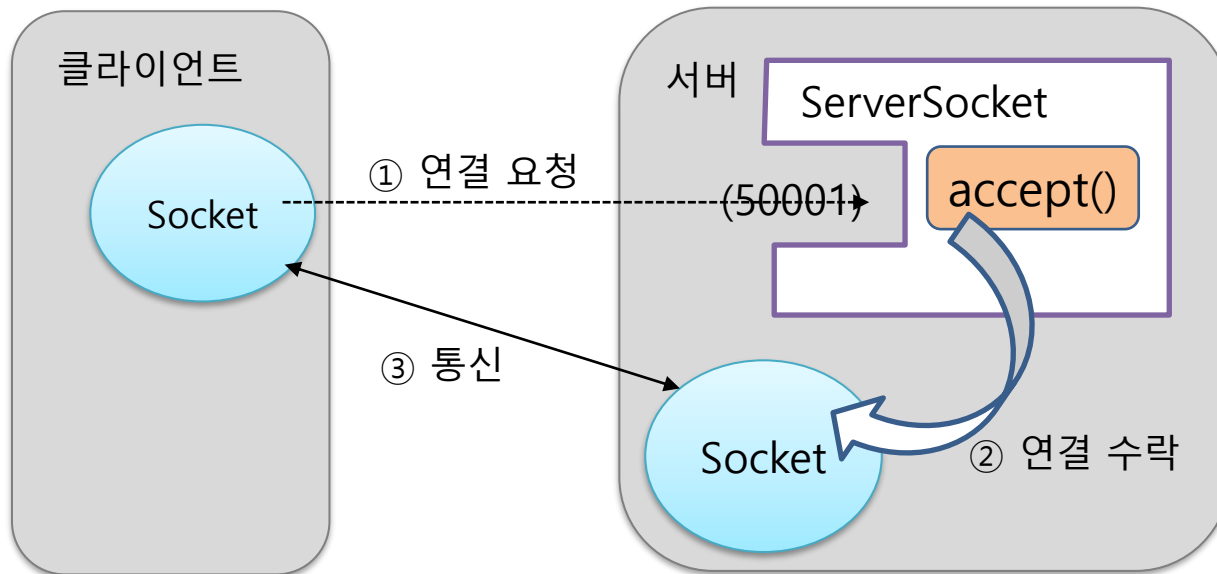
네이버 컴퓨터 IP 주소 : www.naver.com/223.130.195.95



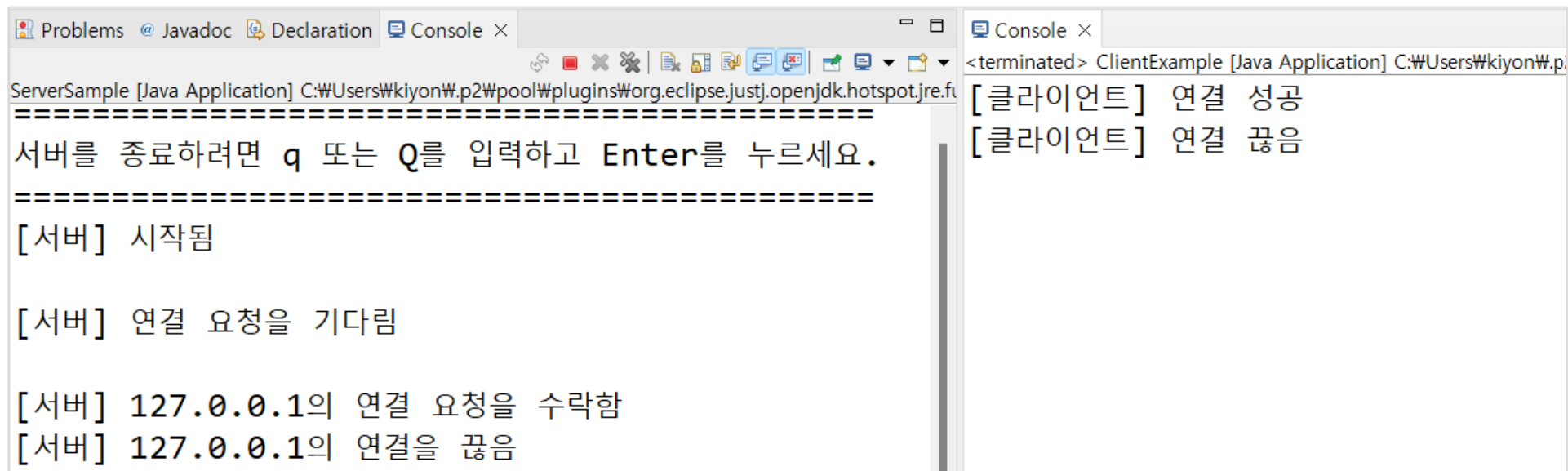
TCP 네트워크

TCP(Transmission Control Protocol)

- 연결 지향적 프로토콜 : 클라이언트와 서버가 연결된 상태에서 데이터를 주고 받는 프로토콜이다.
- 데이터를 정확하고 안정적으로 전달 - 데이터를 순차적으로 보내고 받을 때도 순차적으로 받음
- **Java.net API – ServerSocket 클래스, Socket 클래스**



TCP 네트워킹



```
Problems @ Javadoc Declaration Console x
ServerSample [Java Application] C:\Users\wkiyon\p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.f
=====
서버를 종료하려면 q 또는 Q를 입력하고 Enter를 누르세요.
=====
[서버] 시작됨

[서버] 연결 요청을 기다림

[서버] 127.0.0.1의 연결 요청을 수락함
[서버] 127.0.0.1의 연결을 끊음

Console x
<terminated> ClientExample [Java Application] C:\Users\wkiyon\p
[클라이언트] 연결 성공
[클라이언트] 연결 끊음
```



ServerSocket 생성과 연결 수락

ServerSocket 생성과 연결 수락 – 서버 프로그램

1. ServerSocket 생성과 포트 바인딩

```
ServerSocket serverSocket = new ServerSocket(50001);  
serverSocket.bind(new InetSocketAddress("localhost", 50001);
```

2. 연결 수락

```
try{  
    Socket socket = serverSocket.accept();  
}catch(Exception e){ }
```

3. 연결된 클라이언트 IP 주소 얻기

```
InetSocketAddress socketAddress =  
    (InetSocketAddress)socket.getRemoteSocketAddress();
```

2. 연결 끊기

```
try{  
    serverSocket.close();  
}catch(Exception e){ }
```



ServerSocket 연결 수락

연결 수락 - 서버 프로그램

```
public class ServerSample {
    //서버 소켓 객체 선언
    private static ServerSocket serverSocket;
    //main 스레드 - 키보드 입력 : 서버 종료하는 작업
    //작업 스레드 - 클라이언트 요청 받아서 수락하는 작업
    public static void main(String[] args) {
        System.out.println("=====");
        System.out.println("서버를 종료하려면 q 또는 Q를 입력하고 Enter를 누르세요.");
        System.out.println("=====");

        //TCP 서버 시작
        startServer();

        Scanner scanner = new Scanner(System.in);
        while(true) {
            String key = scanner.nextLine();
            if(key.toLowerCase().equals("q")) {
                break;
            }
        }
        scanner.close();
        //TCP 서버 종료
        stopServer();
    }
}
```



ServerSocket 연결 수락

```
public static void startServer() {  
    //작업 스레드 정의  
    Thread thread = new Thread() {  
        @Override  
        public void run() {  
            //ServerSocket 생성 및 Port 바인딩  
            try {  
                serverSocket = new ServerSocket(50001);  
                System.out.println("[서버] 시작됨"); //실행 : BindException 확인  
  
                while(true) { //여러 클라이언트의 연결 요청 수락을 위해 필요함  
                    System.out.println("\n[서버] 연결 요청을 기다림\n");  
                    //연결 수락  
                    Socket socket = serverSocket.accept();  
  
                    InetAddress isa =  
                        (InetAddress)socket.getRemoteSocketAddress();  
                    //String clientIp = isa.getHostName(); //컴퓨터 이름이 나올수 있음  
                    String clientIp = isa.getHostString();  
                    System.out.println("[서버] " + clientIp + "의 연결 요청을 수락함");  
                    //웹 브라우저에 ip주소:50001 입력 -> 콘솔에 IP 주소 출력됨
```

[서버] 연결 요청을 기다림

[서버] 192.168.35.183의 연결 요청을 수락함

[서버] 192.168.35.183의 연결을 끊음

[서버] 연결 요청을 기다림



ServerSocket 연결 수락

```
        //연결 끊기
        socket.close();
        System.out.println("[서버] " + clientIp + "의 연결을 끊음");
    }
} catch (IOException e) {
    //System.out.println("[서버] " + e.getMessage());
    System.out.println("[서버] " + e.toString()); //한 번 더 실행
}
}
};
thread.start();
}

public static void stopServer() {
    try {
        serverSocket.close();
    } catch (IOException e) {
        e.printStackTrace();
    }
}
```

* q를 입력하면 SocketException 발생함

```
q
[서버] 종료됨
[서버] Socket closed
java.net.SocketException: Socket closed
    at java.base/sun.nio.ch.NioSocketImpl.endAccept(NioSocketImpl.java:689)
    at java.base/sun.nio.ch.NioSocketImpl.accept(NioSocketImpl.java:762)
    at java.base/java.net.ServerSocket.implAccept(ServerSocket.java:675)
    at java.base/java.net.ServerSocket.platformImplAccept(ServerSocket.java:641)
    at java.base/java.net.ServerSocket.implAccept(ServerSocket.java:617)
    at java.base/java.net.ServerSocket.implAccept(ServerSocket.java:574)
    at java.base/java.net.ServerSocket.accept(ServerSocket.java:532)
    at server.ServerSample$1.run(ServerSample.java:47)
```



TCP 클라이언트

Socket 생성과 연결 요청 – 소켓 클라이언트

1. Socket 생성과 포트 바인딩 -> 서버에 연결 요청

```
Socket socket = new Socket("localhost", 50001);
```

2. 연결 끊기

```
socket.close();
```



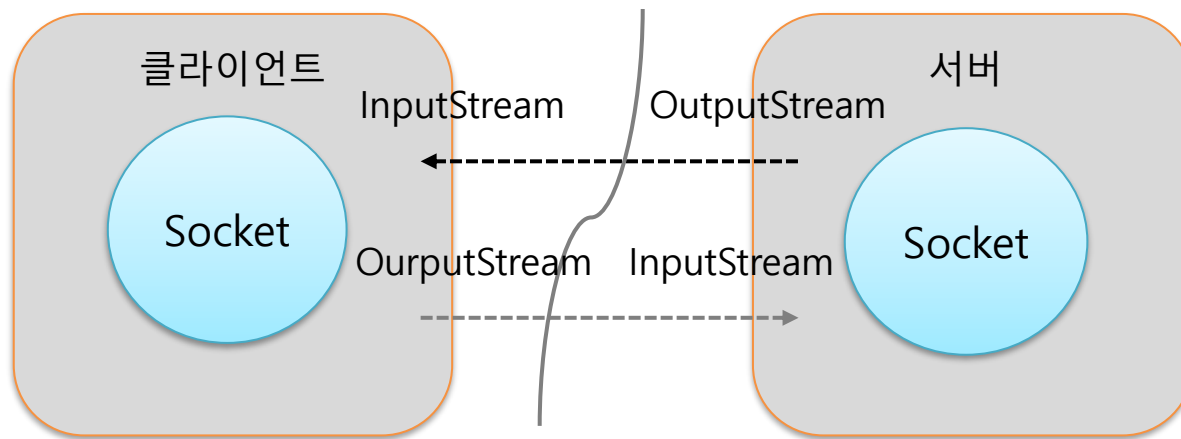
Socket 연결 요청

```
public class ClientExample {  
  
    public static void main(String[] args) {  
        try {  
            //Socket 생성과 동시에 연결 요청  
            Socket socket = new Socket("localhost", 50001);  
  
            System.out.println("[클라이언트] 연결 성공");  
  
            socket.close();  
            System.out.println("[클라이언트] 연결 끊음");  
        } catch (UnknownHostException e) {  
            //IP 또는 도메인 표기 방법이 잘못된 경우  
            System.out.println("UnknownHostException" + e.toString());  
        } catch (IOException e) {  
            //IP 또는 포트번호가 잘못된 경우  
            System.out.println("IOException" + e.toString());  
        }  
    }  
}
```



Socket 데이터 보내고 받기

Socket 데이터 통신



//입력스트림 얻기

InputStream is = socket.getInputStream()

//출력스트림 얻기

OutputStream os = socket.getOutputStream()



Socket 데이터 보내고 받기

Socket 데이터 통신

➤ 데이터 보내기(쓰기)

```
String data = "보낼 데이터";  
byte[ ] bytes = data.getBytes("UTF-8");  
OutputStream os = socket.getOutputStream()  
os.write(bytes);  
os.flush()
```

➤ 데이터 받기(읽기)

```
byte[ ] bytes = new byte[100];  
InputStream inputStream = socket.getInputStream()  
int readByteCount = inputStream.read(bytes);  
String data = new String(bytes, 0, readByteCount, "UTF-8")
```



Socket 데이터 보내고 받기

[클라이언트] 연결 성공
[클라이언트] 데이터 보냄: 오늘도 즐거운 하루 되세요~
[클라이언트] 데이터 받음: 오늘도 즐거운 하루 되세요~
[클라이언트] 연결 끊음

```
=====
서버를 종료하려면 q 또는 Q를 입력하고 Enter를 누르세요.
=====
[서버] 시작됨

[서버] 연결 요청을 기다림

[서버] 127.0.0.1의 연결 요청을 수락함
[서버] 받은 데이터를 다시 보냄: 오늘도 즐거운 하루 되세요~
[서버] 127.0.0.1의 연결을 끊음
```



ServerSocket 데이터 보내고 받기

Socket 데이터 통신 – 서버 프로그램 (EcoServer.java)

```
while(true) {
    System.out.println("\n[서버] 연결 요청을 기다림\n");
    Socket socket = serverSocket.accept();

    InetAddress isa = (InetAddress)socket.getRemoteSocketAddress();
    //String clientIp = isa.getHostName(); //컴퓨터 이름이 나올수 있음
    String clientIp = isa.getHostString();
    System.out.println("[서버] " + clientIp + "의 연결 요청을 수락함");
    //-----
    //데이터 받기
    InputStream is = socket.getInputStream();
    byte[] bytes = new byte[1024]; //1KB
    int readByteCount = is.read(bytes); //읽은 바이트 수
    String message = new String(bytes, 0, readByteCount, "utf-8"); //디코딩 문자셋

    //데이터 보내기
    OutputStream os = socket.getOutputStream();
    bytes = message.getBytes("utf-8"); //인코딩 문자셋
    os.write(bytes);
    os.flush();
    System.out.println("[서버] 받은 데이터를 다시 보냄: " + message);
    //-----
}
```



ServerSocket 데이터 보내고 받기

보조 스트림 사용하기 – DataInputStream, DataOutputStream

```
//데이터 받기
DataInputStream dis = new DataInputStream(socket.getInputStream());
String message = dis.readUTF();

//데이터 보내기
DataOutputStream dos = new DataOutputStream(socket.getOutputStream());
dos.writeUTF(message);
dos.flush();
System.out.println("[서버] 받은 데이터를 다시 보냄: " + message);
//-----
```



Socket 데이터 보내고 받기

Socket 데이터 통신 – 클라이언트 프로그램(EcoClient.java)

```
try {
    Socket socket = new Socket("localhost", 50001);

    System.out.println("[클라이언트] 연결 성공");
    //-----
    //데이터 보내기
    String sendMessage = "오늘도 즐거운 하루 되세요~";
    OutputStream os = socket.getOutputStream();
    byte[] bytes = sendMessage.getBytes("utf-8");
    os.write(bytes);
    os.flush();
    System.out.println("[클라이언트] 데이터 보냄: " + sendMessage);

    //데이터 받기
    InputStream is = socket.getInputStream();
    bytes = new byte[1024];
    int readByteCount = is.read(bytes); //읽은 바이트 수
    //디코딩 문자셋
    String receiveMessage = new String(bytes, 0, readByteCount, "utf-8");
    System.out.println("[클라이언트] 데이터 받음: " + receiveMessage);
    //-----
}
```



Socket 데이터 보내고 받기

보조 스트림 사용하기 – DataInputStream, DataOutputStream

```
//데이터 보내기
String sendMessage = "오늘도 즐거운 하루 되세요~";
DataOutputStream dos = new DataOutputStream(socket.getOutputStream());
dos.writeUTF(sendMessage);
dos.flush();
System.out.println("[클라이언트] 데이터 보냄: " + sendMessage);

//데이터 받기
DataInputStream dis = new DataInputStream(socket.getInputStream());
String receiveMessage = dis.readUTF();
System.out.println("[클라이언트] 데이터 받음: " + receiveMessage);
//-----
```



JSON 데이터 형식

네트워크로 전달하는 데이터가 복잡할 수록 구조화된 형식이 필요하다.

네트워크에서 데이터 송신과 수신에 많이 사용되는 데이터 형식은 **JSON**과 **XML**이다.

JSON은 JavaScript Object Notation의 약자로 자바스크립트의 객체 형식을 기반으로 만들어졌고, 표기법은 아래와 같다.

```
{  
  "id": "sky123",  
  "name": "이하늘",  
  "age": 28,  
  "tel": {"mobile": "010-1234-5678", "home": "02-111-2222"},  
  "student": true,  
  "skill": ["java", "c", "c++"]  
}
```



JSON 데이터 형식

객체 표기	<pre>{ "속성명" : 속성값 "속성명" : 속성값 ... }</pre>	<p>속성명: 반드시 쌍따옴표("")로 감싸야함 <i>속성값으로 가능한 것</i></p> <ul style="list-style-type: none">- 문자열, 숫자, true/false- 객체{ }- 배열[...]
배열 표기	<pre>[요소1, 요소2]</pre>	<p><i>요소로 가능한 것</i></p> <ul style="list-style-type: none">- 문자열, 숫자, true/false- 객체{ }- 배열[...]



JSON 데이터 형식

JSON 라이브러리

JSON을 만들고 해석할 수 있는 라이브러리이다.

Maven Repository > json검색 > json in java > 20230628버전 > homepage > git
으로 이동



image credit: Ismael Pérez Ortiz

JSON in Java [package org.json] [↗](#)

maven-central v20230618

[Click here if you just want the latest release jar file.](#)

Overview [↗](#)



JSON 데이터 형식



JSON In Java

JSON is a light-weight, language independent, data interchange form JSON encoders/decoders in Java. It also includes the capability to cor a reference implementation. There is a large number of JSON packag one. Until then, choose carefully.

License	Public
Categories	JSON Libraries
Tags	json format
Ranking	#91 in MvnRepository (See Top Artifacts) #5 in JSON Libraries
Used By	5,411 artifacts

Central (24) Jahia (1) Redhat GA (3) Redhat EA (1) EmergyaPub (1)
PNT (61) OW2 Public (1) ICM (2)

Version	Vulner
20230618	
20230227	



Home » org.json » json » 20230618



JSON In Java » 20230618

JSON is a light-weight, language indepen JSON encoders/decoders in Java. It also in a reference implementation. There is a lar one. Until then, choose carefully.

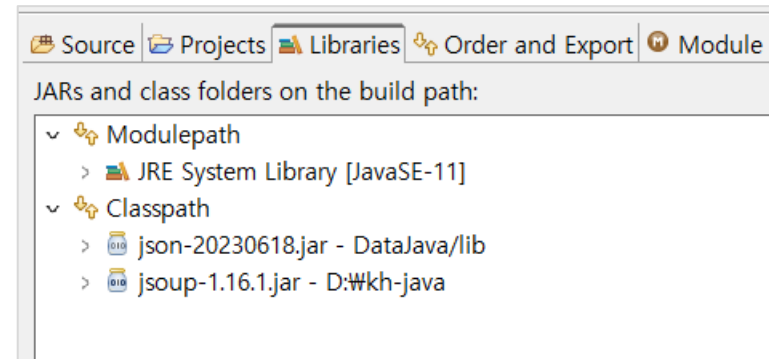
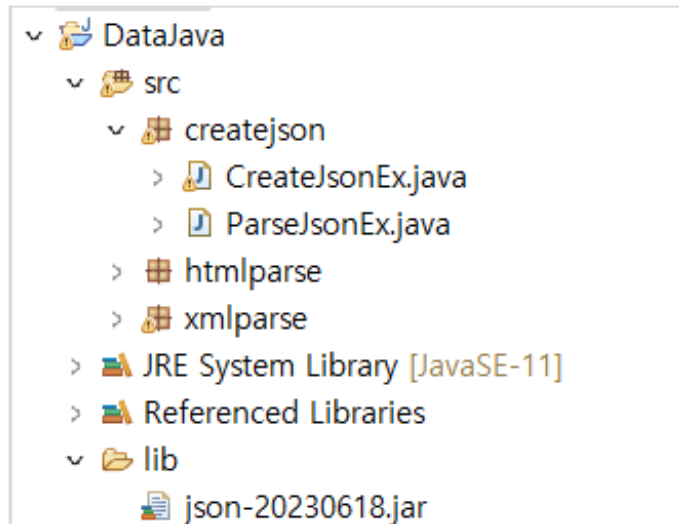
License	Public
Categories	JSON Libraries
Tags	json format
HomePage	https://github.com/douglasc
Date	2023-06-18



JSON 데이터 형식

이클립스에 세팅하기

프로젝트 > 우클릭 > 폴더 > lib 생성 > json jar 파일 복사 > 우클릭 > Add to Build Path



JSON 데이터 형식

JSON 관련 주요 클래스

클래스	용 도
JSONObject	JSON으로 객체를 생성하거나 파싱할 때 사용
JSONArray	JSON으로 배열을 생성하거나 파싱할 때 사용



JSON 데이터 실습 예제

JSON 만들기 - 회원 정보

```
import org.json.JSONArray;
import org.json.JSONObject;

public class CreateJsonEx {

    public static void main(String[] args) throws IOException {
        //JSON 객체 생성
        JSONObject root = new JSONObject();

        //속성 추가
        root.put("id", "sky123");
        root.put("name", "이하늘");
        root.put("age", 28);
        root.put("student", true);

        //객체 속성 추가
        JSONObject tel = new JSONObject();
        tel.put("home", "02-111-2222");
        tel.put("mobile", "010-1234-5678");
        root.put("tel", tel);
    }
}
```



```
{"student":true,"skill":["java","c","c++"],"name":"이하늘","tel":{"mobile":"010-1234-5678"}}
```

JSON 데이터 실습 예제

```
//배열 속성 추가
JSONArray skill = new JSONArray();
skill.put("java");
skill.put("c");
skill.put("c++");
root.put("skill", skill);

//JSON 열기
String json = root.toString(); //문자열로 얻기(생성)
System.out.println(json);

//파일로 저장
//출력 스트림 객체 생성
Writer writer = new FileWriter("C:/jsontdata/member.json",
    Charset.forName("utf-8"));
writer.write(json); //데이터 쓰기

writer.flush();
writer.close();
}
```

```
{"student":true,"skill":["java","c","c++"],"name":"이하늘","tel":{"mobile":"010-1234-5678","home":"02-111-2222"},"id":"sky123","age":28}
```



JSON 데이터 실습 예제

JSON 파싱(해석)하기 – 문자열 출력

```
public class ParseJsonEx {  
  
    public static void main(String[] args) throws IOException {  
        //파일로부터 JSON 열기  
        //BufferedReader의 readLine() 사용  
        BufferedReader br = new BufferedReader(  
            new FileReader("C:/jsondata/member.json", Charset.forName("utf-8")));  
  
        String json = br.readLine(); //한 행 읽기  
        br.close();  
  
        //JSON 파싱(해석)  
        JSONObject root = new JSONObject(json);  
    }  
}
```

```
id: sky123  
name: 이하늘  
age: 28  
student: true  
home: 02-111-2222  
mobile: 010-1234-5678  
skill: java, c, c++,
```



JSON 데이터 실습 예제

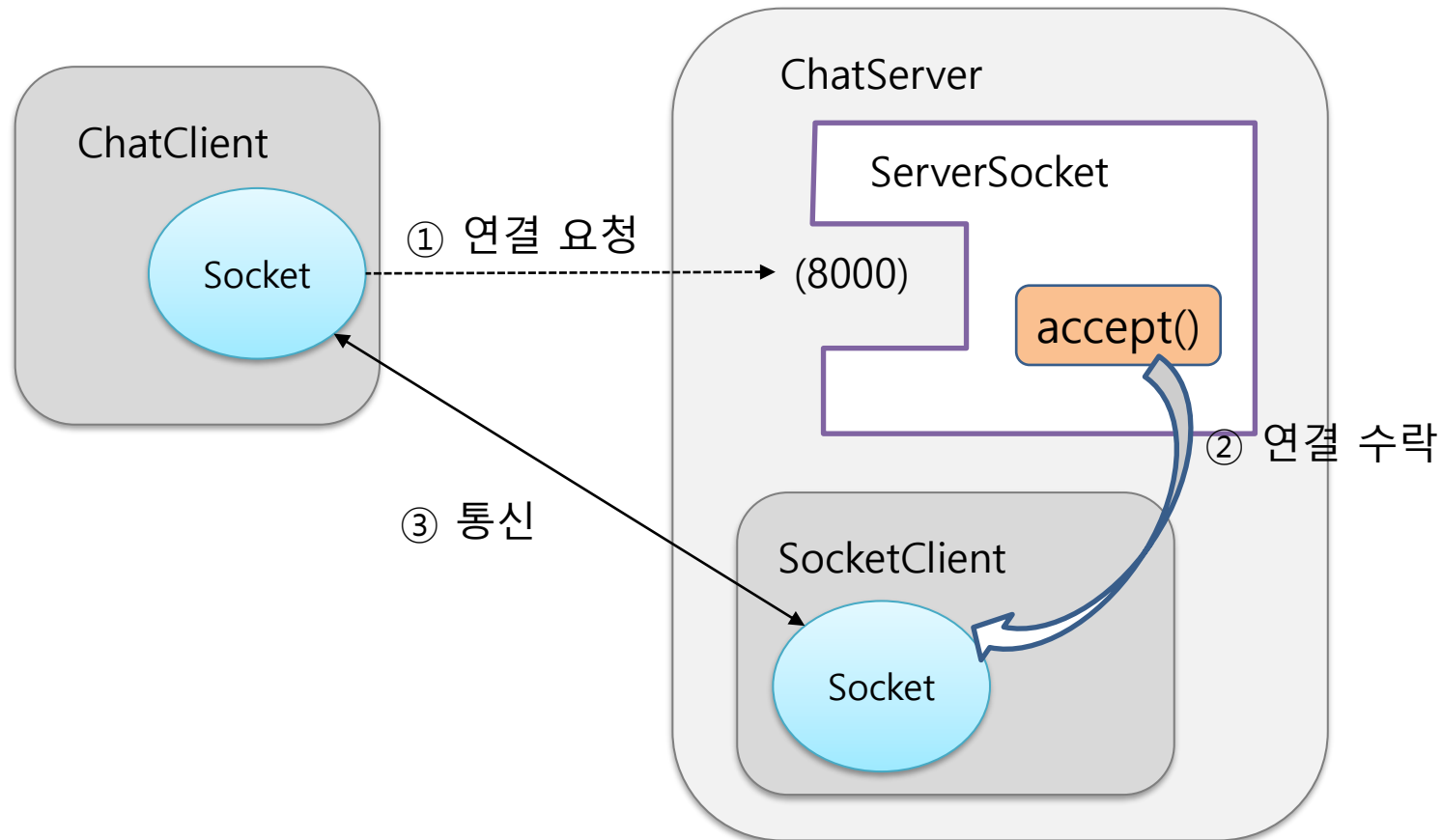
```
//속성 정보 읽기
System.out.println("id: " + root.getString("id"));
System.out.println("name: " + root.getString("name"));
System.out.println("age: " + root.getInt("age"));
System.out.println("student: " + root.getBoolean("student"));

//객체 속성 정보 읽기
JSONObject tel = root.getJSONObject("tel");
System.out.println("home: " + tel.getString("home"));
System.out.println("mobile: " + tel.getString("mobile"));

//배열 속성 정보 읽기
JSONArray skill = root.getJSONArray("skill");
System.out.print("skill: ");
for(int i=0; i<skill.length(); i++) {
    System.out.print(skill.get(i) + ", ");
}
}
```



TCP 채팅 프로그램



TCP 채팅 프로그램

클래스	용 도
ChatServer	<ul style="list-style-type: none">- 채팅 서버 실행 클래스- ServerSocket을 생성하고 8000에 바인딩- ChatClient 연결 수락 후 SocketClient 생성
SocketClient	<ul style="list-style-type: none">- ChatClient와 1:1로 통신
ChatClient	<ul style="list-style-type: none">- 채팅 클라이언트 실행 클래스- ChatServer에 연결 요청- SocketClient와 1:1로 통신



채팅 서버

[클라이언트] 서버에 연결됨
대화명 입력:

[서버] 시작됨

=====
서버를 종료하려면 q 또는 Q를 입력하고 Enter를 누르세요.
=====

[클라이언트] 서버에 연결됨
대화명 입력: 자바웹

=====
보낼 메시지를 입력하고 Enter
채팅을 종료하려면 q 또는 Q를 입력하고 Enter
=====

[서버] 시작됨

=====
서버를 종료하려면 q 또는 Q를 입력하고 Enter를 누르세요.
=====

입장: 자바웹@127.0.0.1

현재 채팅자 수: 1



채팅 서버

[클라이언트] 서버에 연결됨

대화명 입력: 자바 웹

=====

보낼 메시지를 입력하고 Enter

채팅을 종료하려면 q 또는 Q를 입력하고 Enter

=====

<안드로이드 앱@127.0.0.1>입장하셨습니다.

[서버] 시작됨

=====

서버를 종료하려면 q 또는 Q를 입력하고 Enter를 누르세요.

=====

입장: 자바 웹@127.0.0.1

현재 채팅자 수: 1

입장: 안드로이드 앱@127.0.0.1

현재 채팅자 수: 2

Console ×

ChatClient [Java Application] C:\Users\wkiyon\p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_17.0.7.v20230425-1502\jre\bin\javaw.exe (2023. 10. 4. 오전 1:06:39) [pid: 13196]

대화명 입력: 자바 웹

=====

보낼 메시지를 입력하고 Enter

채팅을 종료하려면 q 또는 Q를 입력하고 Enter

=====

<안드로이드 앱@127.0.0.1>입장하셨습니다.



채팅 서버

[클라이언트] 서버에 연결됨

대화명 입력: 자바 웹

=====

보낼 메시지를 입력하고 Enter

채팅을 종료하려면 q 또는 Q를 입력하고 Enter

=====

<안드로이드 앱@127.0.0.1>입장하셨습니다.

안녕하세요

방가와요~^^

좋은 하루 보내세요~



Console ×

ChatClient [Java Application] C:\Users\Wkiyon\p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.wir

=====

<안드로이드 앱@127.0.0.1>입장하셨습니다.

안녕하세요

방가와요~^^

좋은 하루 보내세요~



채팅 서버

보낼 메시지를 입력하고 Enter

채팅을 종료하려면 q 또는 Q를 입력하고 Enter

=====

<안드로이드 앱@127.0.0.1>입장하셨습니다.

안녕하세요

방가와요~^^

좋은 하루 보내세요~

q

[클라이언트] 종료

[클라이언트] 서버에 연결 끊김

=====

입장: 자바 웹@127.0.0.1

현재 채팅자 수: 1

입장: 안드로이드 앱@127.0.0.1

현재 채팅자 수: 2

나감: 자바 웹@127.0.0.1

현재 채팅자 수: 1

Console ×

<terminated> ChatClient [Java Application] C:\Users\wkiyon\p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_17.0.7.v20230425-1502\jre\bin\javaw.exe (2023.1

방가와요~^^

좋은 하루 보내세요~

q

[클라이언트] 종료

[클라이언트] 서버에 연결 끊김



채팅 서버

ChatServer.java

```
import java.io.IOException;
import java.net.ServerSocket;
import java.net.Socket;
import java.util.Collection;
import java.util.Collections;
import java.util.HashMap;
import java.util.Hashtable;
import java.util.Map;
import java.util.Scanner;
import java.util.concurrent.ExecutorService;
import java.util.concurrent.Executors;

import org.json.JSONObject;

public class ChatServer {
    //필드
    ServerSocket serverSocket;
    //쓰레드 풀
    ExecutorService threadPool = Executors.newFixedThreadPool(100);

    ////멀티 스레드 환경 : 채팅방 맵 객체 - chatRoom
    //Map<String, SocketClient> chatRoom = new Hashtable<>();
```



채팅 서버

```
Map<String, SocketClient> chatRoom =
    Collections.synchronizedMap(new HashMap<>());

public void start() throws IOException {
    serverSocket = new ServerSocket(8000);
    System.out.println("[서버] 시작됨");

    //항상 연결을 준비함 - 스레드를 만들어야함
    Thread thread = new Thread(()->{
        //()->{} : Runnable 인터페이스의 run() 구현
        try {
            while(true) {
                //클라이언트의 연결 요청 수락(블로킹 해제) 및 소켓 객체 생성
                Socket socket = serverSocket.accept();
                SocketClient sc = new SocketClient(this, socket);
            }
        } catch (IOException e) { //블럭 비움
        }
    });
    thread.start();
}
```



채팅 서버

```
public void stop() {  
    try {  
        serverSocket.close();  
        threadPool.shutdown();  
        //반복자 - 내부 스트림 - stream() : 위의 Collection 사용해도 됨  
        /*Collection<SocketClient> socketClient = chatRoom.values();  
        for(SocketClient sc : socketClient) {  
            sc.close();  
        }*/  
        chatRoom.values().stream().forEach(sc -> sc.close());  
        System.out.println("[서버] 종료됨 ");  
    } catch (IOException e) {  
        e.printStackTrace();  
    }  
}
```



채팅 서버

```
public static void main(String[] args) {  
    try {  
        ChatServer chatServer = new ChatServer();  
        chatServer.start();  
  
        System.out.println("=====");  
        System.out.println("서버를 종료하려면 q 또는 Q를 입력하고 Enter를 누르세요.");  
        System.out.println("=====");  
  
        Scanner scanner = new Scanner(System.in);  
        while(true) {  
            String key = scanner.nextLine();  
            if(key.toLowerCase().equals("q")) {  
                break;  
            }  
        }  
        scanner.close();  
        chatServer.stop();  
    } catch (IOException e) {  
        System.out.println("[서버] " + e.getMessage());  
    }  
}
```



채팅 서버

ChatServer.java

```
//클라이언트 소켓 추가
public void addSocketClient(SocketClient socketClient) {
    //키 : 채팅방 이름
    String key = socketClient.chatName + "@" + socketClient.clientIp;
    chatRoom.put(key, socketClient);
    System.out.println("입장: " + key);
    System.out.println("현재 채팅자 수: " + chatRoom.size() + "\n");
}

//클라이언트 소켓 삭제
public void removeSocketClient(SocketClient socketClient) {
    String key = socketClient.chatName + "@" + socketClient.clientIp;
    chatRoom.remove(key);
    System.out.println("나감: " + key);
    System.out.println("현재 채팅자 수: " + chatRoom.size() + "\n");
}
```



채팅 서버

ChatServer.java

```
//모든 클라이언트에게 보내기 - json
public void sendToAll(SocketClient sender, String message) {
    JSONObject root = new JSONObject();
    root.put("clientId", sender.clientId);
    root.put("chatName", sender.chatName);
    root.put("message", message);
    String json = root.toString();

    //key:chatName, value=socketClient
    //values() 리턴 타입 - Collection - 외부 반복자
    Collection<SocketClient> socketClient = chatRoom.values();
    for(SocketClient sc : socketClient) {
        //발신자와 동일하면 보내지 말고 다음 클라이언트를 보냄
        if(sc == sender) continue;
        sc.send(json);
    }
}
```



채팅 서버

```
//서버측에서 통신하는 클래스 정의
public class SocketClient {
    ChatServer chatServer;
    Socket socket;
    String clientIp;
    String chatName;
    DataInputStream dis;
    DataOutputStream dos;

    public SocketClient(ChatServer chatServer, Socket socket) {
        try {
            this.chatServer = chatServer;
            this.socket = socket;
            dis = new DataInputStream(socket.getInputStream());
            dos = new DataOutputStream(socket.getOutputStream());

            InetAddress isa =
                (InetAddress) socket.getRemoteSocketAddress();
            this.clientIp = isa.getHostAddress();
            //this.clientIp = isa.getHostString();
            receive();
        } catch (IOException e) {
        }
    }
}
```

SocketClient.java



채팅 서버

```
//메서드 : JSON 받기
public void receive() {
    //항상 받을 준비를 함(대기)
    chatServer.threadPool.execute(()->{
        try {
            while(true) {
                String receiveJson = dis.readUTF(); //읽을때 연결이 끊기면 예외 발생()

                //JSONObject로 파싱 - command(명령:요청내용) 속성의 구조
                //{ "command": "incoming", "data": "chatName" }
                //{ "command": "message", "data": "xxxx(메시지)" }
                JSONObject jsonObject = new JSONObject(receiveJson);
                String command = jsonObject.getString("command"); //command 속성값 얻기

                switch(command) {
                    case "incoming": //채팅방 입장
                        this.chatName = jsonObject.getString("data");
                        chatServer.sendToAll(this, "입장하셨습니다."); //SocketClient.this
                        chatServer.addSocketClient(this);
                        break;
                }
            }
        } catch (IOException e) {
            //예외 처리
        }
    });
}
```



채팅 서버

SocketClient.java

```
        case "message": //메시지 보냄
            String message = jsonObject.getString("data"); //data 속성값 얻기
            chatServer.sendToAll(this, message);
            break;
        }
    }
} catch (IOException e) {
    //연결이 끊겼을때 예외 발생
    chatServer.sendToAll(this, "나가셨습니다.");
    chatServer.removeSocketClient(this); //현재 소켓 클라이언트 삭제
}
});
}
```



채팅 서버

SocketClient.java

```
//메서드 : JSON 보내기
public void send(String json) {
    try {
        dos.writeUTF(json);
        dos.flush();
    } catch (IOException e) {
    }
}

public void close() {
    try {
        socket.close();
    } catch (IOException e) {
    }
}
}
```



채팅 서버

ChatClient.java

```
public class ChatClient {  
    //필드  
    Socket socket;  
    DataInputStream dis;  
    DataOutputStream dos;  
    String chatName;  
  
    //연결 요청 - TCP 이므로  
    public void connect() throws IOException{  
        //상대방 host 입력  
        socket = new Socket("localhost", 8000);  
        dis = new DataInputStream(socket.getInputStream());  
        dos = new DataOutputStream(socket.getOutputStream());  
        System.out.println("[클라이언트] 서버에 연결됨");  
    }  
}
```



채팅 서버

```
public void receive() {  
    //항상 받을 준비함 - 스레드를 만들어야함  
    Thread thread = new Thread(()->{  
        while(true) {  
            try {  
                String json = dis.readUTF();  
  
                //읽은 데이터를 파싱함  
                //어떤 컴퓨터에서 어떤 누가 어떤 메시지를 보냄  
                JSONObject root = new JSONObject(json);  
                String clientId = root.getString("clientId");  
                String chatName = root.getString("chatName");  
                String message = root.getString("message");  
                System.out.println("<" + chatName + "@" + clientId + "> " + message);  
            } catch (IOException e) {  
                System.out.println("[클라이언트] 서버에 연결 끊김");  
                System.exit(0); //process 완전 종료  
            }  
        }  
    });  
    thread.start();  
}
```



채팅 서버

```
public void send(String json) throws IOException{
    dos.writeUTF(json);
    dos.flush();
}

public void unconnect() throws IOException{
    socket.close();
}

public static void main(String[] args) {
    try {
        ChatClient chatClient = new ChatClient();
        chatClient.connect();

        Scanner scanner = new Scanner(System.in);
        System.out.print("대화명 입력: ");
        chatClient.chatName = scanner.nextLine();

        //command: 입장, data: 채팅방 이름
        JSONObject jsonObject = new JSONObject();
        jsonObject.put("command", "incoming");
        jsonObject.put("data", chatClient.chatName);
        String json = jsonObject.toString();
    }
}
```



채팅 서버

```
chatClient.send(json); //데이터를 보냄
chatClient.receive(); //받을 준비함

System.out.println("=====");
System.out.println("보낼 메시지를 입력하고 Enter");
System.out.println("채팅을 종료하려면 q 또는 Q를 입력하고 Enter");
System.out.println("=====");
while(true) {
    String message = scanner.nextLine(); //채팅 문자 입력
    if(message.toLowerCase().equals("q")) {
        break;
    }else {
        //command: 메시지 보냄
        jsonObject = new JSONObject();
        jsonObject.put("command", "message");
        jsonObject.put("data", message);
        json = jsonObject.toString();
        chatClient.send(json);
    }
}
```



채팅 서버

ChatClient.java

```
        scanner.close();
        chatClient.disconnect();    //연결 끊음
    }catch(Exception e) {
        System.out.println("[클라이언트] 서버 연결 안됨");
    }
    System.out.println("[클라이언트] 종료");
}
```

