

# × 네트워크프로그래밍 기초

한성대학교 컴퓨터공학부

신 성



# TCP/IP 소개

## ■ TCP/IP 프로토콜

- TCP는 Transmission Control Protocol
- 두 시스템 간에 신뢰성 있는 데이터의 전송을 관장하는 프로토콜
- TCP에서 동작하는 응용프로그램 사례
  - e-mail, FTP, 웹(HTTP) 등

프로토콜: 통신할 때 사용하는 약속, 예) 무전기의 '오버', 동시에 두 사람이 같이 말할 수 없다. 대화가 원활하게 이루어지지 않는다. 규칙이 필요

## ■ IP

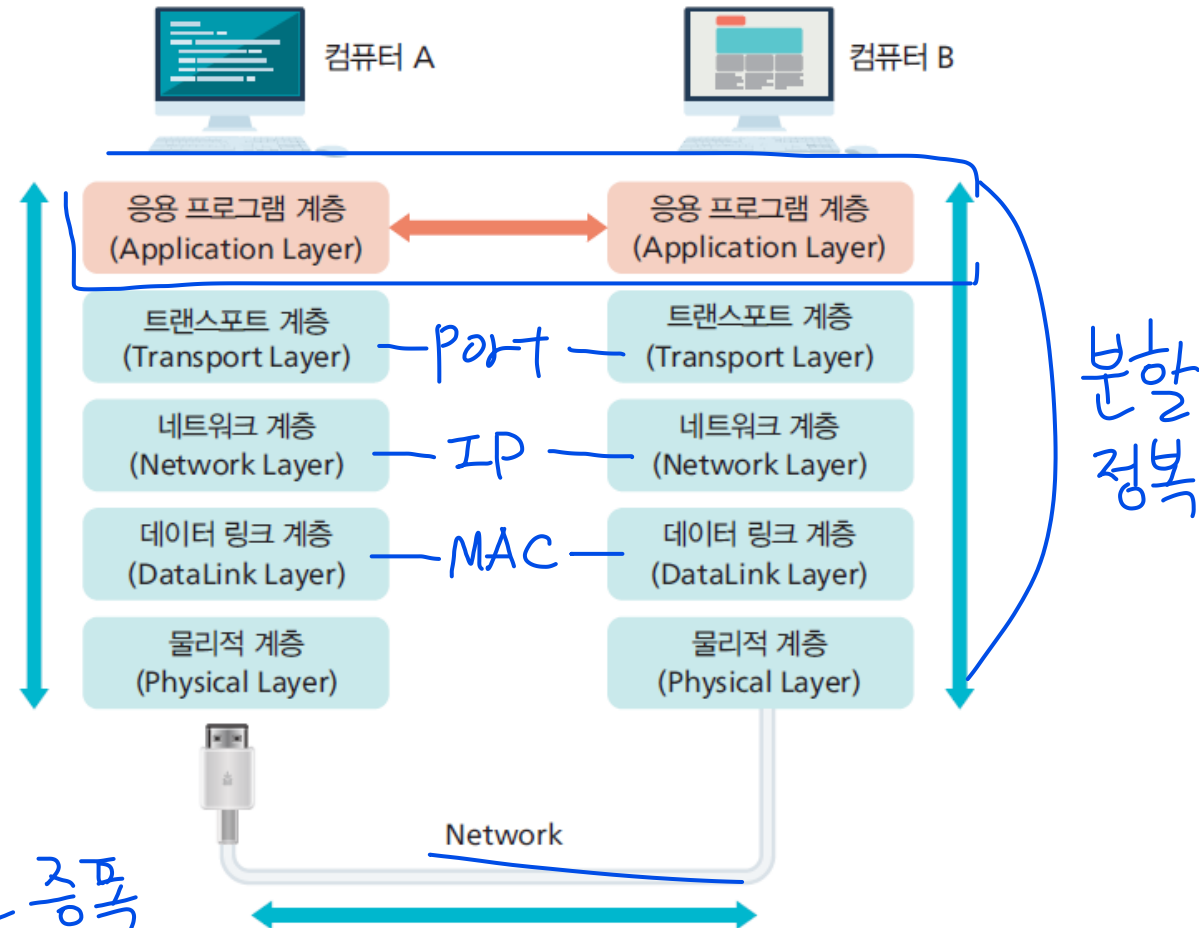
- Internet Protocol
- 패킷 교환 네트워크에서 송신 호스트와 수신 호스트가 데이터를 주고 받는 것을 관장하는 프로토콜
- TCP보다 하위 레벨 프로토콜

이와 같이 통신기기와 환경의 특성에 따라 대화자 간에 약속을 정한 것이 프로토콜, 인터넷에 연결된 컴퓨터 간에는 TCP/IP라는 프로토콜 사용, 대표적인 것으로 http, ftp 모두 TCP/IP를 기반으로 하는 프로토콜

4인  
전송

# 프로토콜

- 프로토콜(Protocol)은 컴퓨터 간에 상호통신을 할 때 데이터를 원활하고 신뢰성 있게 주고 받기 위해 필요한 약속을 규정하는 것이다.



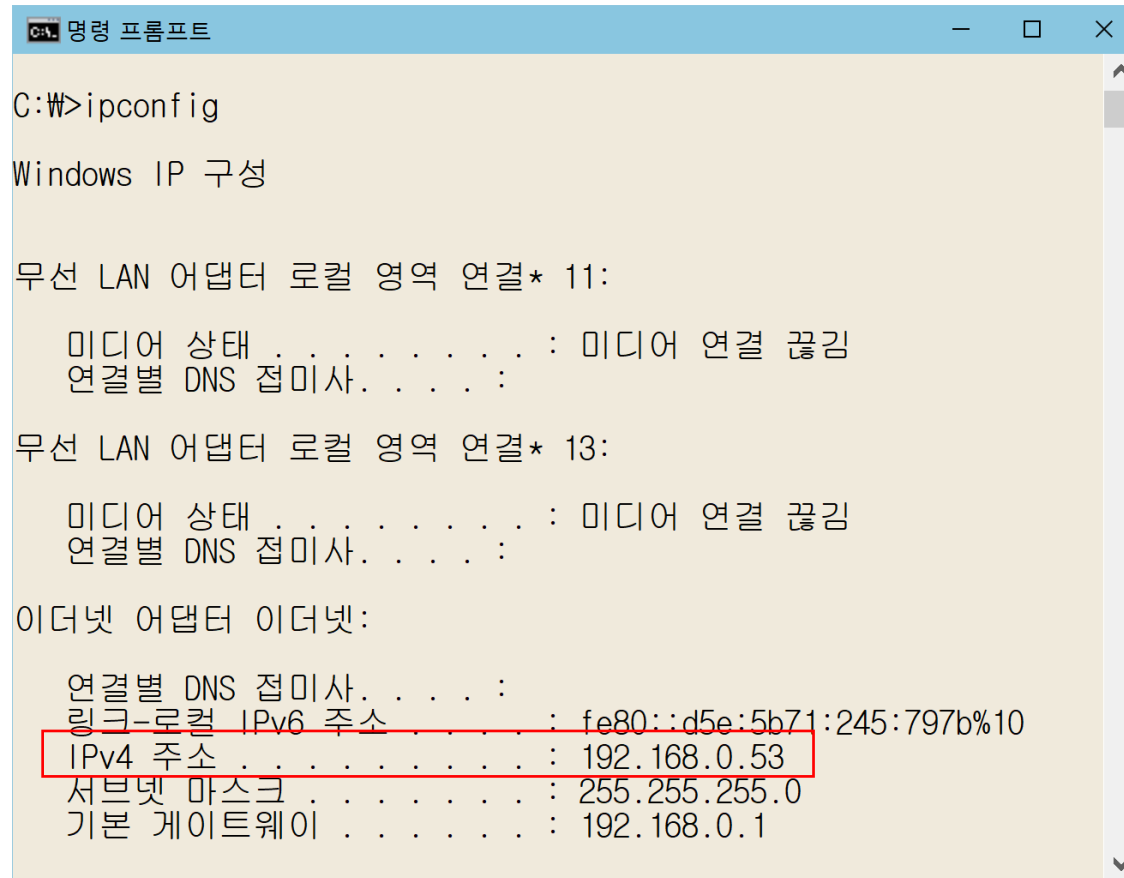
# IP 주소

## ▪ IP 주소

- 네트워크 상에서 유일하게 식별될 수 있는 컴퓨터 주소
  - 숫자로 구성된 주소
  - 4개의 숫자가 '.'으로 연결
    - 예) 192.156.11.15
- 숫자로 된 주소는 기억하기 어려우므로 www.naver.com과 같은 문자열로 구성된 도메인 이름으로 바꿔 사용
  - DNS(Domain Name System)
    - 문자열로 구성된 도메인 이름을 숫자로 구성된 IP 주소로 자동 변환
- 현재는 32비트의 IP 버전 4(IPv4)가 사용되고 있음
  - IP 주소 고갈로 인해 128비트의 IP 버전 6(IPv6)이 점점 사용되는 추세

# 내 컴퓨터의 IP 주소 확인하기

- 내 컴퓨터의 윈도우에서 명령창을 열어 ipconfig 명령 수행



```
C:\>ipconfig

Windows IP 구성

무선 LAN 어댑터 로컬 영역 연결* 11:

    미디어 상태 . . . . . : 미디어 연결 끊김
    연결별 DNS 접미사 . . . . . :

무선 LAN 어댑터 로컬 영역 연결* 13:

    미디어 상태 . . . . . : 미디어 연결 끊김
    연결별 DNS 접미사 . . . . . :

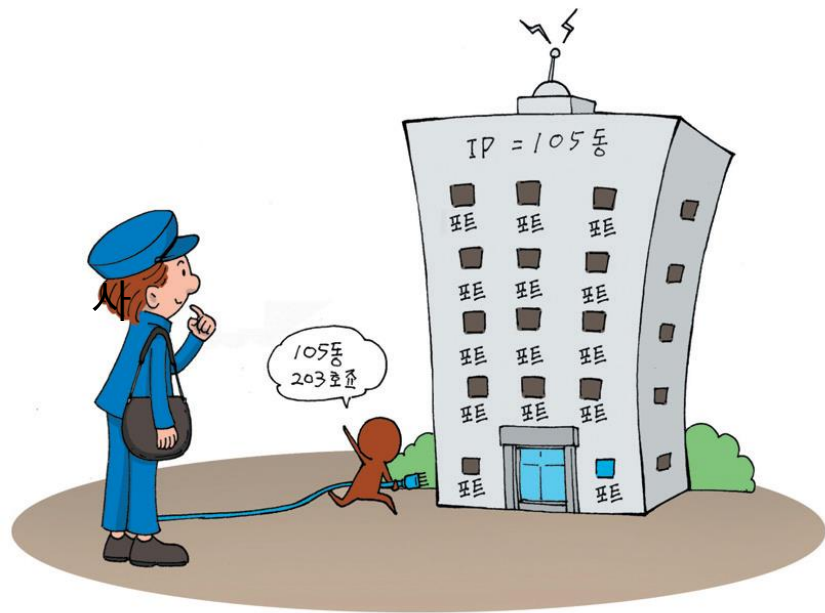
이더넷 어댑터 이더넷:

    연결별 DNS 접미사 . . . . . :
    링크-로컬 IPv6 주소 . . . . . : fe80::d5e:5b71:245:797b%10
    IPv4 주소 . . . . . : 192.168.0.53
    서브넷 마스크 . . . . . : 255.255.255.0
    기본 게이트웨이 . . . . . : 192.168.0.1
```

# 포트

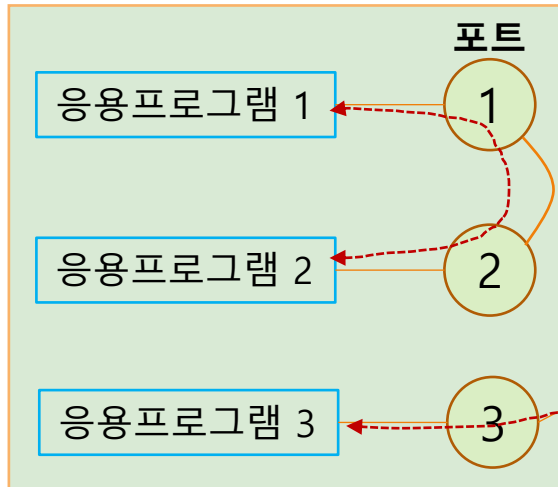
## ■ 포트

- 통신하는 프로그램 간에 가상의 연결단 포트 생성
  - IP 주소는 네트워크 상의 컴퓨터 또는 시스템을 식별하는 주소
  - 포트 번호를 이용하여 통신할 응용프로그램 식별
- 모든 응용프로그램은 하나 이상의 포트 생성 가능
  - 포트를 이용하여 상대방 응용프로그램과 데이터 교환
- 잘 알려진 포트(well-known ports)
  - 시스템이 사용하는 포트 번호
  - 잘 알려진 응용프로그램에서  
용하는 포트 번호
    - 0부터 1023 사이의 포트 번호
    - ex) SSH 22, HTTP 80, FTP 21
  - 잘 알려진 포트 번호는  
개발자가 사용하지 않는 것이 좋음  
(0부터 1023 사이의 포트 번호를 제외한 나머지 ~65535까지의 포트 사용)
    - 충돌 가능성 있음

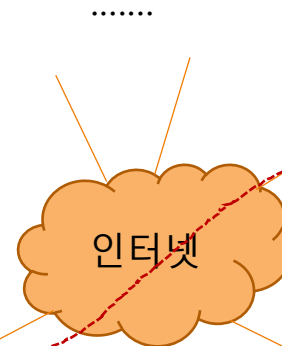


# 포트를 이용한 통신

컴퓨터1(IP: 203.1.1.110)



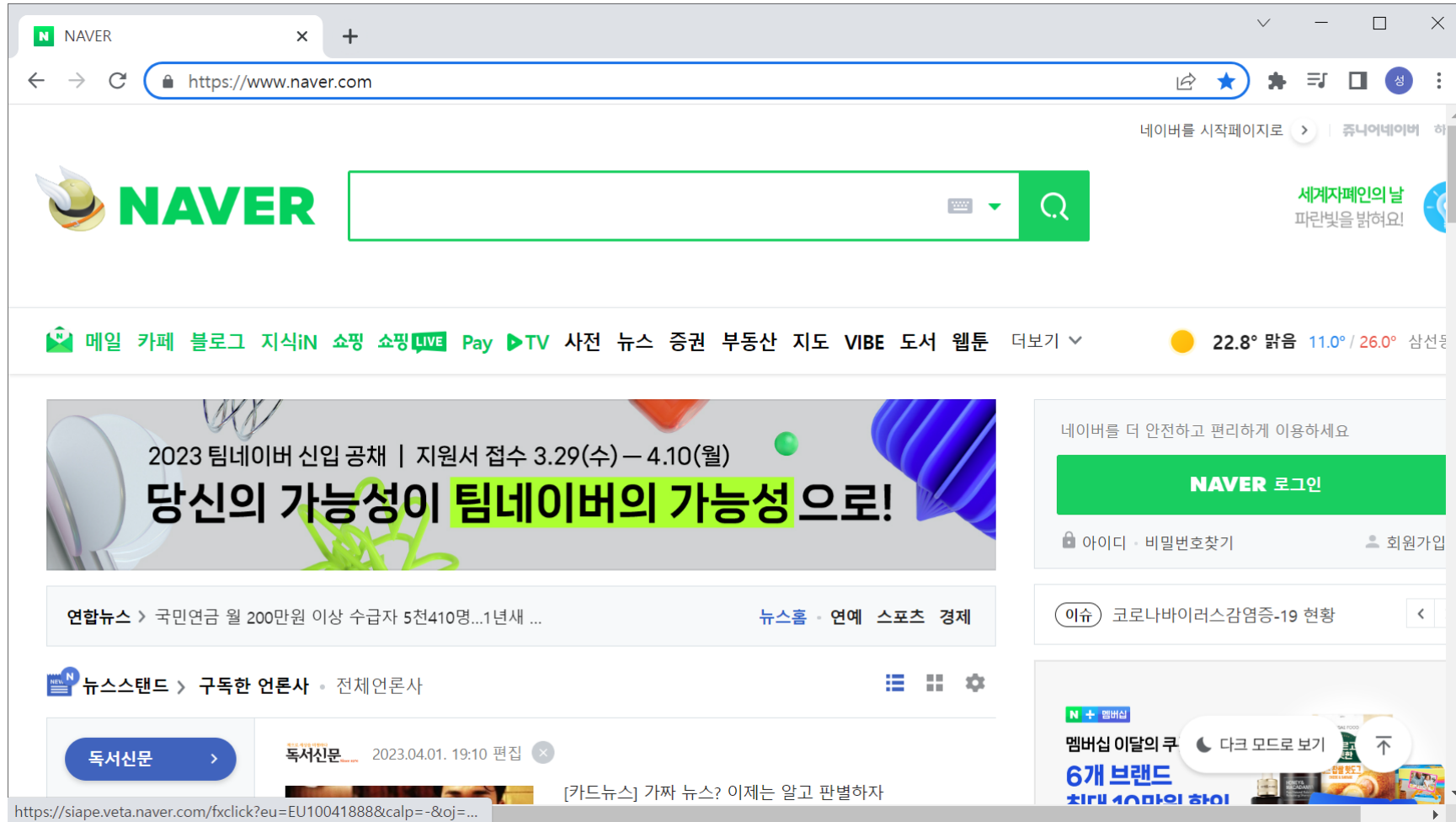
컴퓨터1(IP: 113.67.23.120)



# http 프로토콜의 예

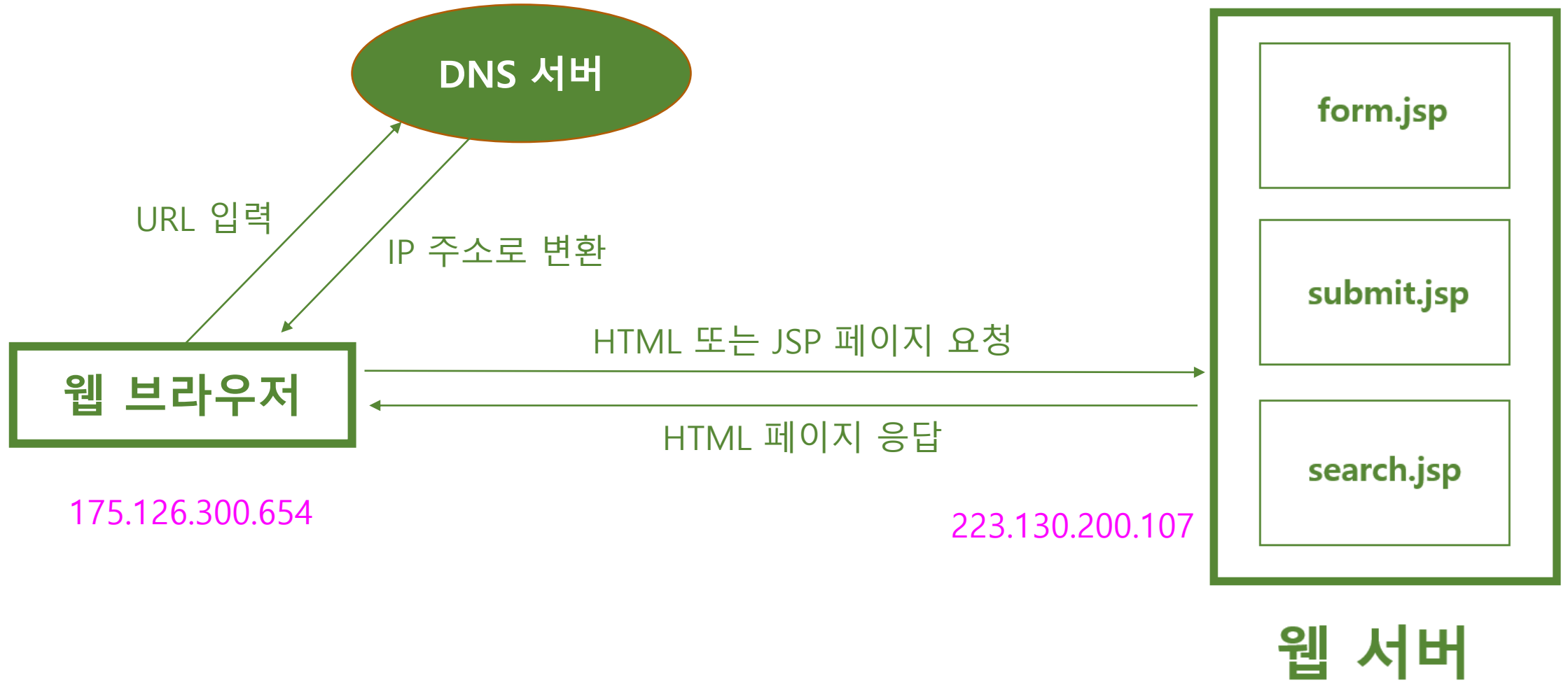
- 네이버의 예 `https://www.naver.com`

`http://서버주소:포트번호/경로명/파일명`

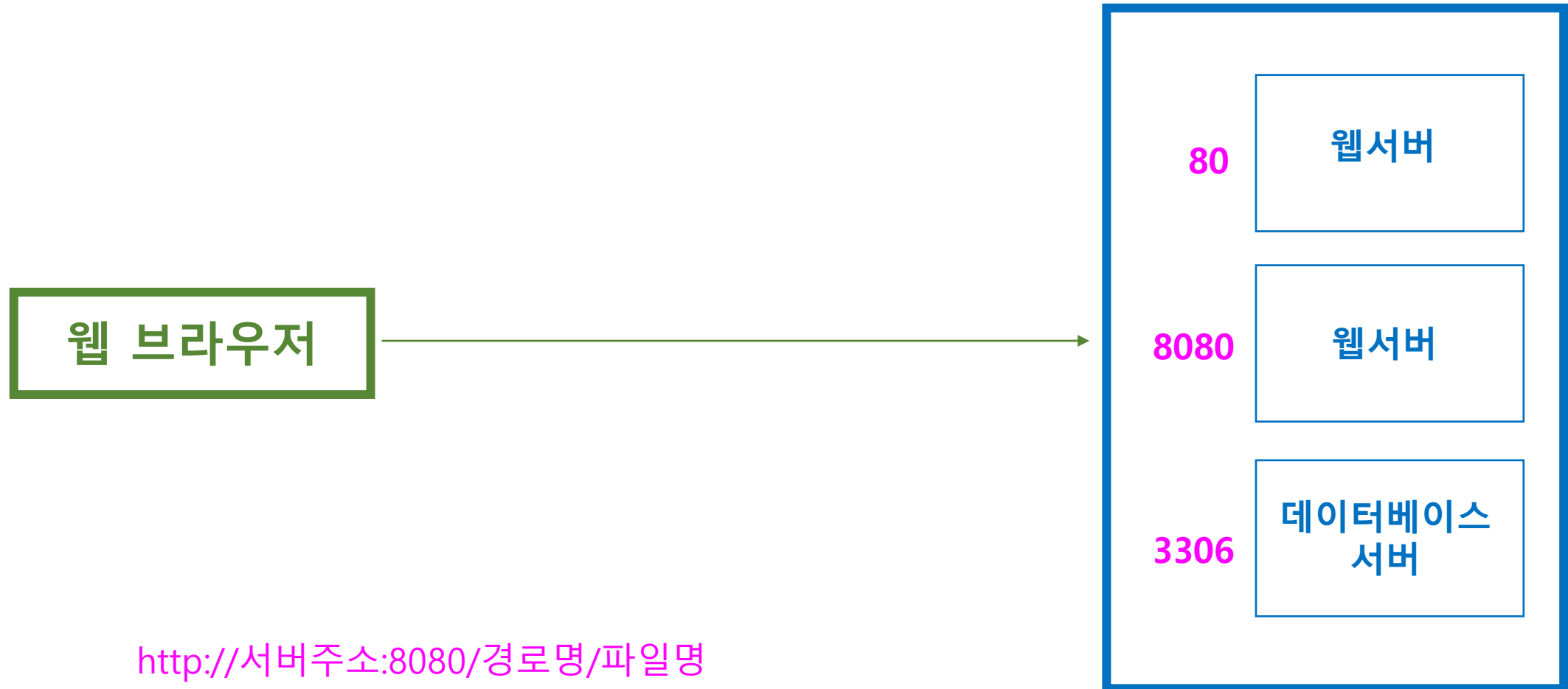




# IP주소와 도메인 네임



# 포트 번호



# 소켓 프로그래밍

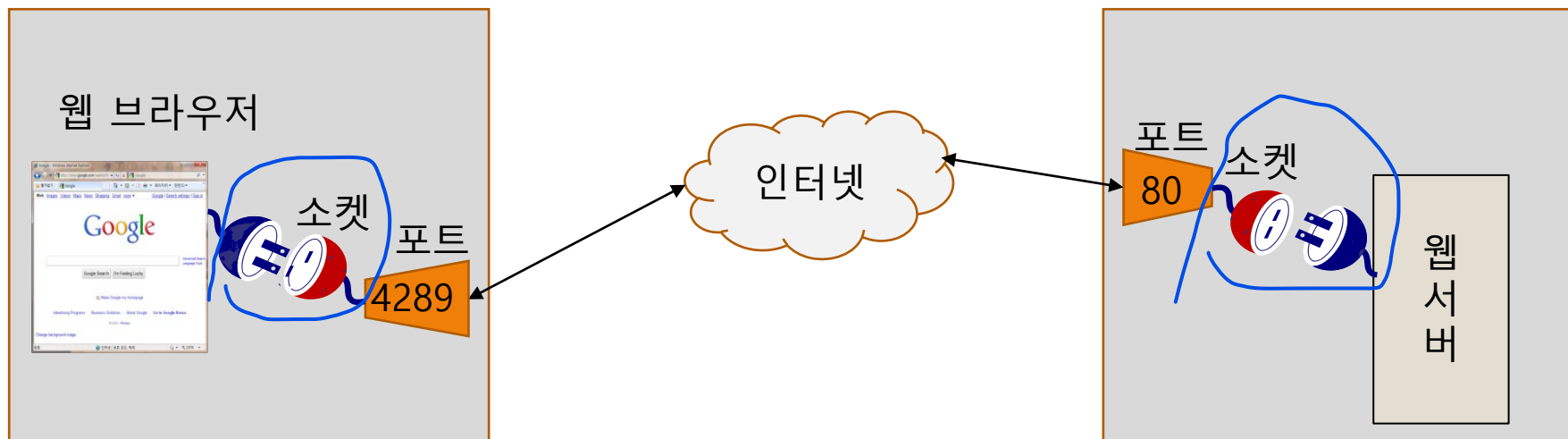
## ■ 소켓 (socket) ← 클래스

- TCP/IP 네트워크를 이용하여 쉽게 통신 프로그램을 작성하도록 지원하는 기반 기술
- 소켓
  - 두 응용프로그램 간의 양방향 통신 링크의 한쪽 끝 단
  - 소켓끼리 데이터를 주고받음
  - 소켓은 특정 IP 포트 번호와 결합
- 자바로 소켓 통신할 수 있는 라이브러리 지원
- 소켓 종류 : 서버 소켓과 클라이언트 소켓

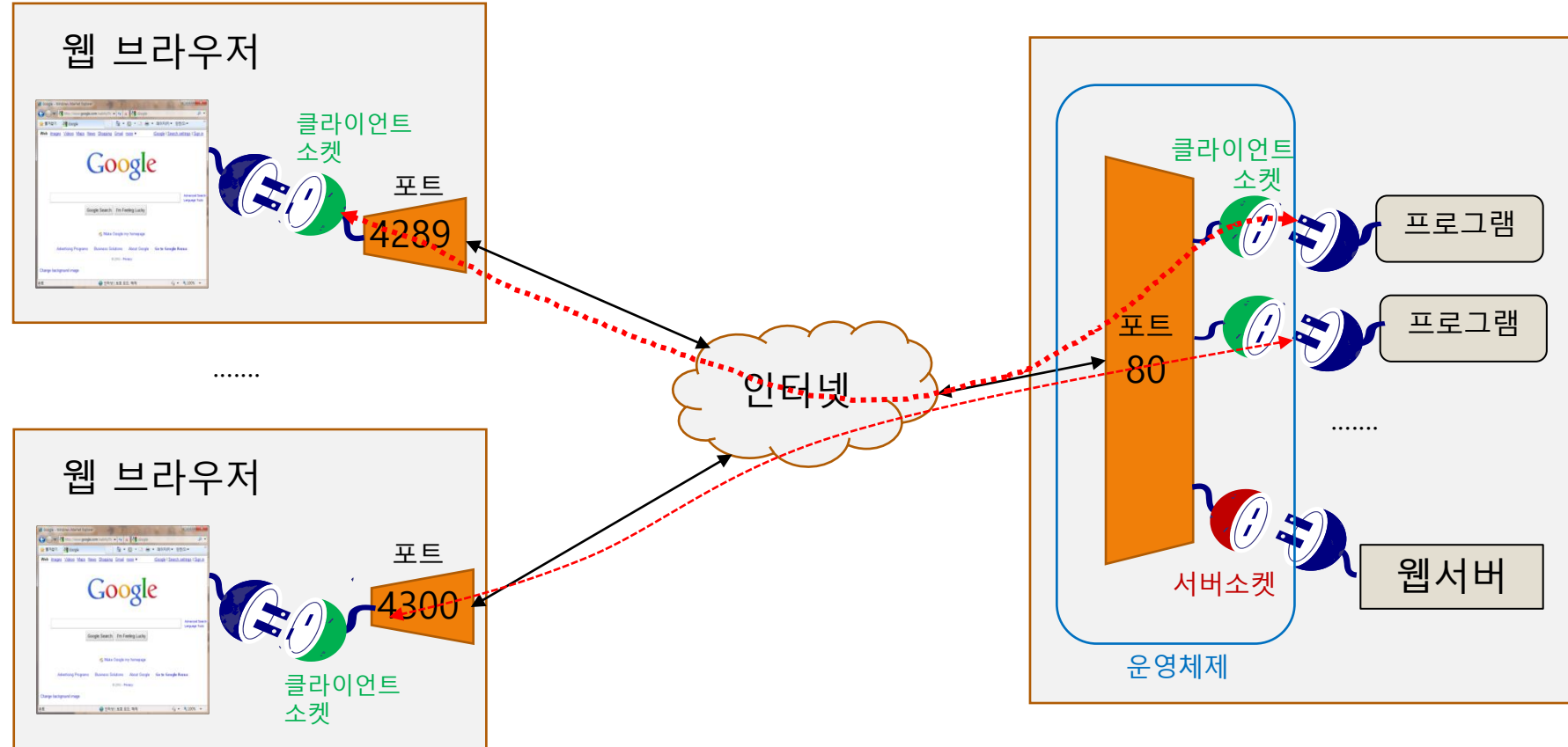
소켓은 콘센트와 같은 역할 수행

소켓을 연결하기만 하면 하위의 복잡한 프로토콜이나 네트워크 상태를 알 필요없이 연결된 소켓을 통해서 데이터를 주고받을 수 있음

서버 소켓 클래스와 (클라이언트) 소켓 클래스



# 소켓을 이용한 웹 서버와 클라이언트 사이의 통신 사례

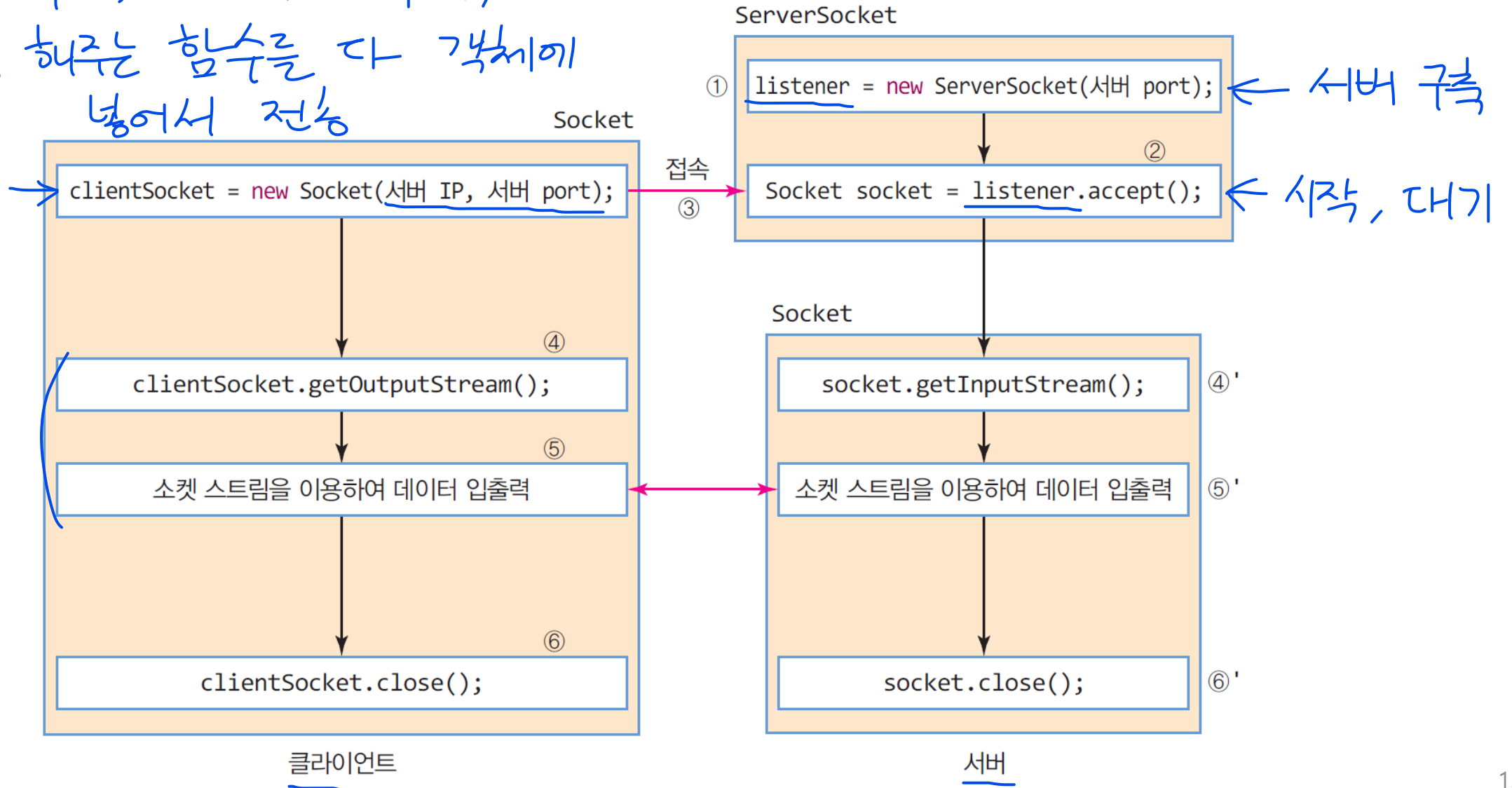


# 소켓을 이용한 서버 클라이언트 통신 프로그램의 전형적인 구조

서버 IP, 서버 port, 클라 IP, 클라 port,  
통신을 쉽게 해주는 함수를 다 객체에  
넣어서 전송

클라 구축,  
접속

23P



# Socket 클래스, 클라이언트 소켓

## ▪ Socket 클래스

- 클라이언트 소켓에 사용되는 클래스
- java.net 패키지에 포함
- 생성자

생성자	설명
Socket	연결되지 않은 상태의 소켓을 생성
Socket(InetAddress address, int port)	소켓을 생성하고, 지정된 IP 주소(addresss)와 포트 번호(port)에서 대기하는 원격 응용프로그램의 소켓에 연결
Socket(String host, int port)	소켓을 생성하여 지정된 호스트(host)와 포트 번호(port)에 연결한다. 호스트 이름이 null인 경우는 루프백(loopback) 주소로 가정

# Socket 클래스의 메소드

메소드	설명
<code>void bind(SocketAddress bindpoint)</code>	소켓에 로컬 IP 주소와 로컬 포트 지정
<code>void close()</code>	소켓을 닫는다.
<code>void connect(SocketAddress endpoint)</code>	소켓을 서버에 연결
<code>InetAddress getInetAddress()</code>	소켓에 연결된 서버 IP 주소 반환
<code>InputStream getInputStream()</code>	소켓에 연결된 입력 스트림 반환. 이 스트림을 이용하여 소켓이 상대방으로부터 받은 데이터를 읽을 수 있음
<code>InetAddress getLocalAddress()</code>	소켓이 연결된 로컬 주소 반환
<code>int getLocalPort()</code>	소켓의 로컬 포트 번호 반환
<code>int getPort()</code>	소켓에 연결된 서버의 포트 번호 반환
<code>OutputStream getOutputStream()</code>	소켓에 연결된 출력 스트림 반환. 이 스트림에 출력하면 소켓이 서버로 데이터 전송
<code>boolean isBound()</code>	소켓이 로컬 주소에 연결되어있으면 true 반환
<code>boolean isConnected()</code>	소켓이 서버에 연결되어 있으면 true 반환
<code>boolean isClosed()</code>	소켓이 닫혀있으면 true 반환
<code>void setSoTimeout(int timeout)</code>	데이터 읽기 타임아웃 시간 지정. 0이면 타임아웃 해제

# 클라이언트에서 소켓으로 서버에 접속하는 코드

- 클라이언트 소켓 생성  
및 서버에 접속

```
Socket clientSocket = new Socket("128.12.1.1", 5550);
```

- Socket 객체의 생성되면 곧 바로 128.12.1.1의 주소의 5550포트에 자동 접속

- 소켓으로부터 데이터를 전송할 입출력 스트림 생성

```
BufferedReader in = new BufferedReader(  
    new InputStreamReader(clientSocket.getInputStream()));  
BufferedWriter out = new BufferedWriter(  
    new OutputStreamWriter(clientSocket.getOutputStream()));
```

- 서버로 데이터 전송
  - flush()를 호출하면 스트림 속에 데이터를 남기지 않고 모두 전송

```
out.write("hello" + "\n");  
out.flush();
```

- 서버로부터 데이터 수신

```
String line = in.readLine(); (반을 때까지 대기)  
//서버로부터 한 행의 문자열 수신
```

- 네트워크 접속 종료

```
clientSocket.close();
```



# ServerSocket 클래스, 서버 소켓

## ▪ ServerSocket 클래스

- 서버 소켓에 사용되는 클래스, java.net 패키지에 포함
- 생성자

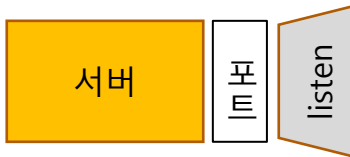
생성자	설명
ServerSocket(int port)	지정된 포트 번호(port)와 결합된 소켓 생성

- 메소드

메소드	설명
Socket accept()	클라이언트로부터 연결 요청을 기다리다 요청이 들어오면 수락하고 클라이언트와 데이터를 주고받을 새 Socket 객체를 반환
void close()	서버 소켓을 닫는다.
InetAddress getInetAddress()	서버 소켓의 로컬 IP 주소 반환
int getLocalPort()	서버 소켓의 로컬 포트 번호 반환
boolean isBound()	서버 소켓이 로컬 주소에 연결되어있으면 true 반환
boolean isClosed()	서버 소켓이 닫혀있으면 true 반환
void setSoTimeout(int timeout)	accept()가 대기하는 타임아웃 시간 지정. 0이면 무한정 대기

# 서버에 클라이언트가 연결되는 과정

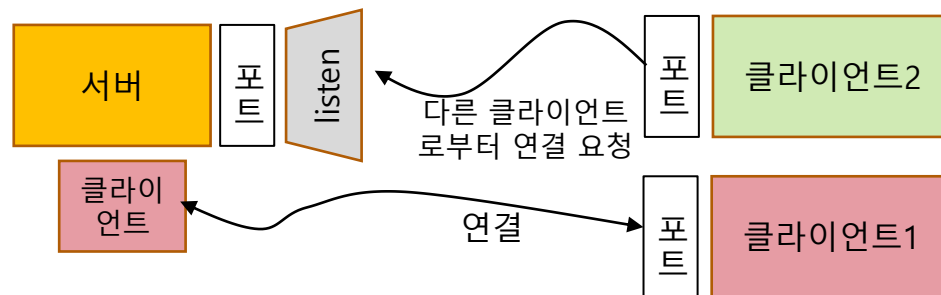
- 서버는 서버 소켓으로 들어오는 연결 요청을 기다림(listen)



- 클라이언트가 서버에게 연결 요청



- 서버가 연결 요청 수락(accept)
  - 새로운 클라이언트 소켓을 만들어 클라이언트와 통신하게 함
  - 그리고 다시 다른 클라이언트의 연결을 기다림



# 서버가 클라이언트와 통신하는 과정

- 서버 소켓 생성

- 서버는 접속을 기다리는 포트로 5550 선택

```
ServerSocket serverSocket = new ServerSocket(5550);
```

- 클라이언트로부터 접속 기다림

- accept() 메소드는 연결 요청이 오면 새로운 Socket 객체 반환
- 접속 후 새로 만들어진 Socket 객체를 통해 클라이언트와 통신

```
Socket socket = serverSocket.accept();
```

- 네트워크 입출력 스트림 생성

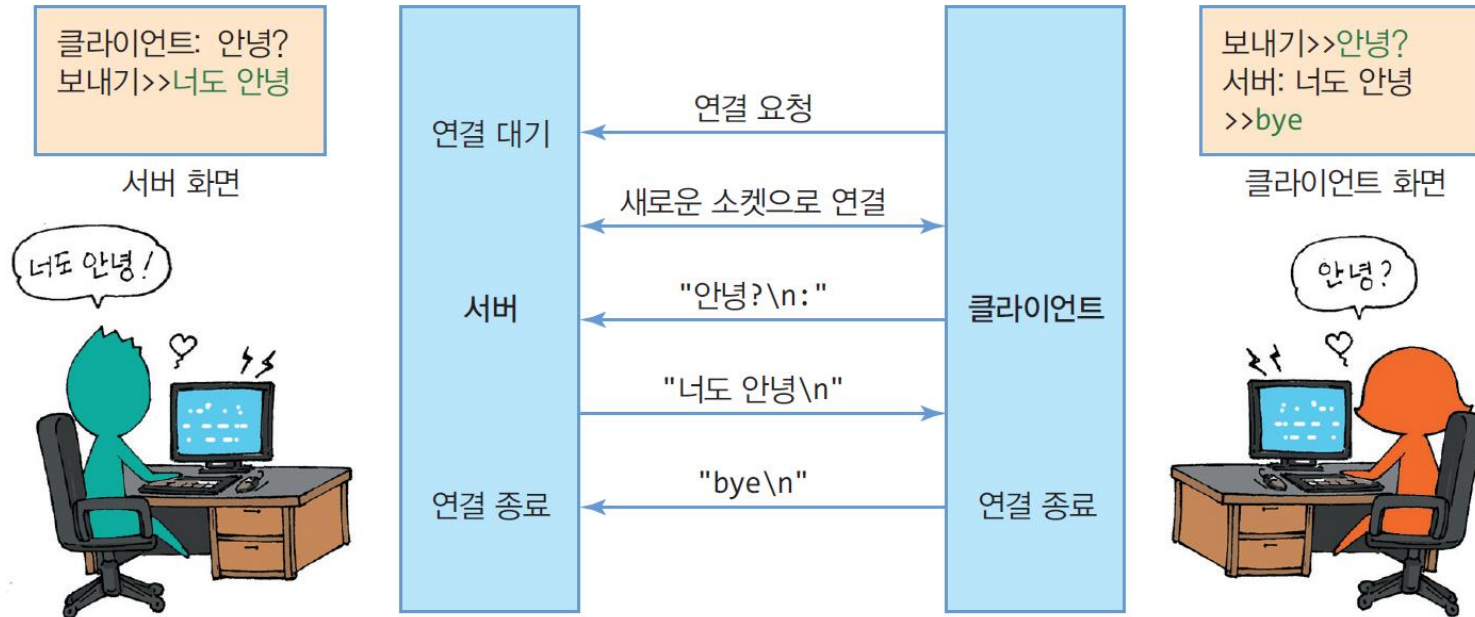
```
BufferedReader in = new BufferedReader(  
    new InputStreamReader(socket.getInputStream()));  
BufferedWriter out = new BufferedWriter(  
    new OutputStreamWriter(socket.getOutputStream()));
```

- Socket 객체의 getInputStream()과 getOutputStream() 메소드를 이용하여 입출력 데이터 스트림 생성

# [예제] 소켓을 이용한 서버/클라이언트 채팅 예제

## ■ 간단한 채팅 프로그램

- 서버와 클라이언트가 1:1로 채팅
- 클라이언트와 서버가 서로 한번씩 번갈아 가면서 문자열 전송
  - 문자열 끝에 "\n"을 덧붙여 보내고 라인 단위로 수신
- 클라이언트가 bye를 보내면 프로그램 종료



# 서버 프로그램 ServerEx.java

cmd |

[올려드린 실습 파일 참고]

실행은 .class 파일을  
(편하게 접근할 수 있도록)

C:\wjavatest 등에 옮기고

명령 프롬프트에서 실행

(명령프롬프트 2개 필요, 서버 실행, 클라이언트 실행)

```
import java.io.*;
import java.net.*;
import java.util.*;

public class ServerEx {
    public static void main(String[] args) {
        BufferedReader in = null;
        BufferedWriter out = null;
        ServerSocket listener = null;
        Socket socket = null;
        Scanner scanner = new Scanner(System.in); // 키보드에서 읽을 scanner 객체 생성
        try {
            → listener = new ServerSocket(9999); // 서버 소켓 생성
            System.out.println("연결을 기다리고 있습니다.....");
            → socket = listener.accept(); // 클라이언트로부터 연결 요청 대기
            System.out.println("연결되었습니다.");
            in = new BufferedReader(new InputStreamReader(socket.getInputStream()));
            out = new BufferedWriter(new OutputStreamWriter(socket.getOutputStream())); = printf
            while (true) {
                String inputMessage = in.readLine(); // 클라이언트로부터 한 행 읽기
                if (inputMessage.equalsIgnoreCase("bye")) {
                    System.out.println("클라이언트에서 bye로 연결을 종료하였음");
                    break; // "bye"를 받으면 연결 종료
                }
                System.out.println("클라이언트: " + inputMessage);
                System.out.print("보내기>>"); // 프롬프트
                String outputMessage = scanner.nextLine(); // 키보드에서 한 행 읽기
                out.write(outputMessage + "\n"); // 키보드에서 읽은 문자열 전송
                out.flush(); // out의 스트림 버퍼에 있는 모든 문자열 전송
            }
        } catch (IOException e) { System.out.println(e.getMessage());
        } finally {
            try {
                scanner.close(); // scanner 닫기
                socket.close(); // 통신용 소켓 닫기
                listener.close(); // 서버 소켓 닫기
            } catch (IOException e) { System.out.println("클라이언트와 채팅 중 오류가 발생했습니다."); }
        }
    }
}
```

# 클라이언트 프로그램

## ClientEx.java

[올려드린 실습 파일 참고]

실행은 .class 파일을  
(편하게 접근할 수 있도록)

C:\wjavatest 등에 옮기고

명령 프롬프트에서 실행

(명령프롬프트 2개 필요, 서버 실행,  
클라이언트 실행)

```
import java.io.*;
import java.net.*;
import java.util.*;

public class ClientEx {
    public static void main(String[] args) {
        BufferedReader in = null;
        BufferedWriter out = null;
        Socket socket = null;
        Scanner scanner = new Scanner(System.in); // 키보드에서 읽을 scanner 객체 생성
        try {
            → socket = new Socket("localhost", 9999); // 클라이언트 소켓 생성. 서버에 연결
            in = new BufferedReader(new InputStreamReader(socket.getInputStream()));
            out = new BufferedWriter(new OutputStreamWriter(socket.getOutputStream()));
            while (true) {
                System.out.print("보내기 >>"); // 프롬프트
                String outputMessage = scanner.nextLine(); // 키보드에서 한 행 읽기
                if (outputMessage.equalsIgnoreCase("bye")) {
                    out.write(outputMessage + "\n"); // "bye" 문자열 전송
                    out.flush();
                    break; // 사용자가 "bye"를 입력한 경우 서버로 전송 후 실행 종료
                }
                out.write(outputMessage + "\n"); // 키보드에서 읽은 문자열 전송
                out.flush(); // out의 스트림 버퍼에 있는 모든 문자열 전송
                String inputMessage = in.readLine(); // 서버로부터 한 행 수신
                System.out.println("서버: " + inputMessage);
            }
        } catch (IOException e) {
            System.out.println(e.getMessage());
        } finally {
            try {
                scanner.close();
                if(socket != null) socket.close(); // 클라이언트 소켓 닫기
            } catch (IOException e) {
                System.out.println("서버와 채팅 중 오류가 발생했습니다.");
            }
        }
    }
}
```

cmd 2

# 스트림

- (지금부터는) 다음 부분에 대한 설명

Scanner 를 쓰지 않고 데이터를 읽어들이는, (24, 29 p)  
한국어도 상관없이 문자씩...

InputStreamReader isr = new InputStreamReader(socket.getInputStream()); // 소켓 입력 스트림  
BufferedReader in = new BufferedReader(isr); ← 문자 '씩' 처리

무조건 1 byte

OutputStreamWriter osw = new OutputStreamWriter(socket.getOutputStream()); // 소켓 출력 스트림  
BufferedWriter out = new BufferedWriter(osw);

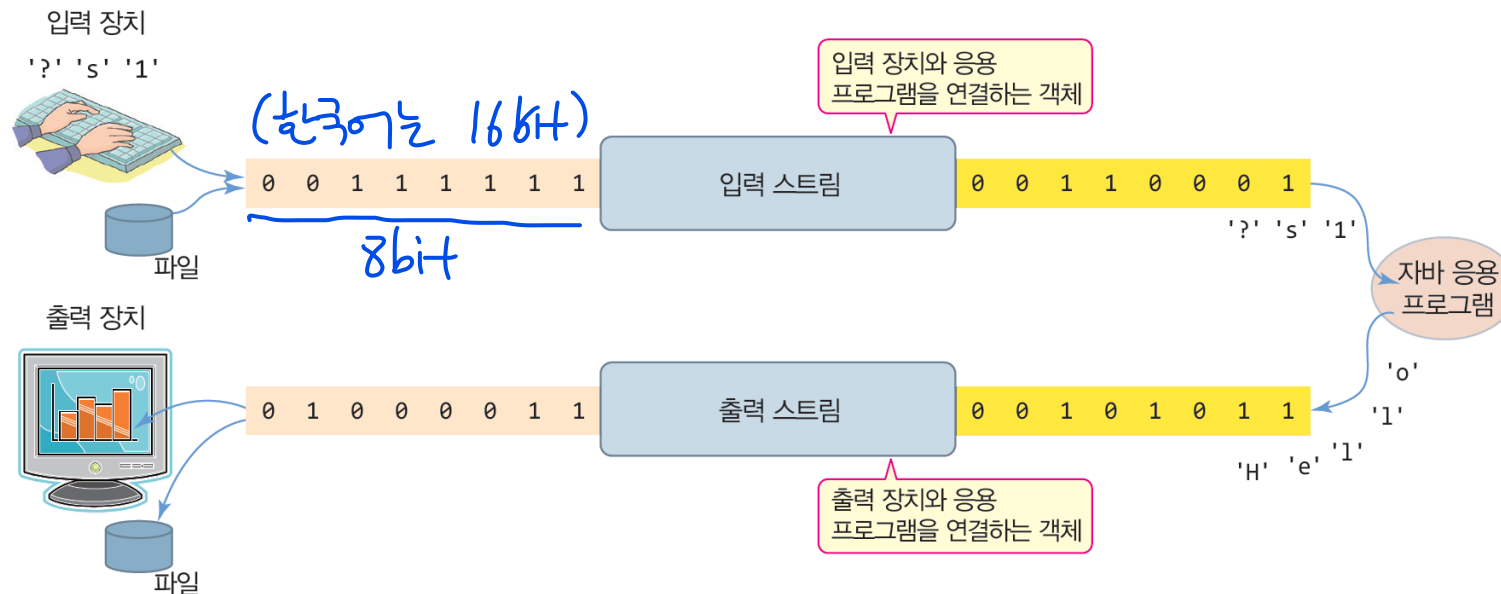
# 스트림

## ■ 스트림 입출력

- 버퍼를 가지고 순차적으로 이루어지는 입출력

## ■ 자바의 입출력 스트림

- 응용프로그램과 입출력 장치를 연결하는 소프트웨어 모듈
  - 입력 스트림 : 입력 장치로부터 자바 프로그램으로 데이터를 전달
  - 출력 스트림 : 출력 장치로 데이터 출력





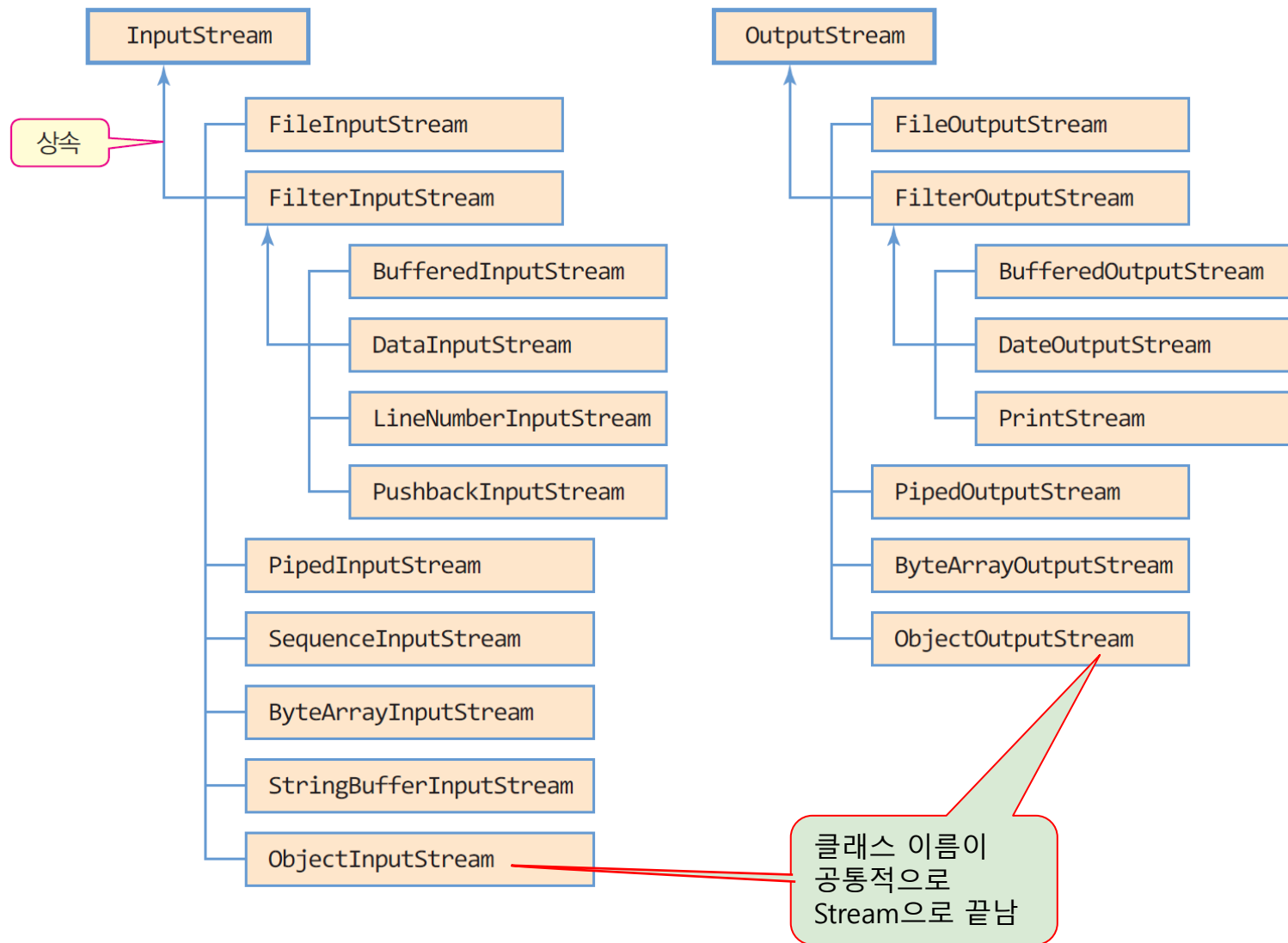
# 자바의 입출력 스트림 특징

- 스트림의 양끝에 입출력장치와 자바 응용프로그램 연결
- 스트림은 단방향
  - 입력과 출력을 동시에 하는 스트림 없음
- 입출력 스트림 기본 단위
  - 바이트 스트림의 경우 : 바이트
  - 문자 스트림의 경우 : 문자(자바에서는 문자1개 : 2 바이트)
- 선입선출 구조

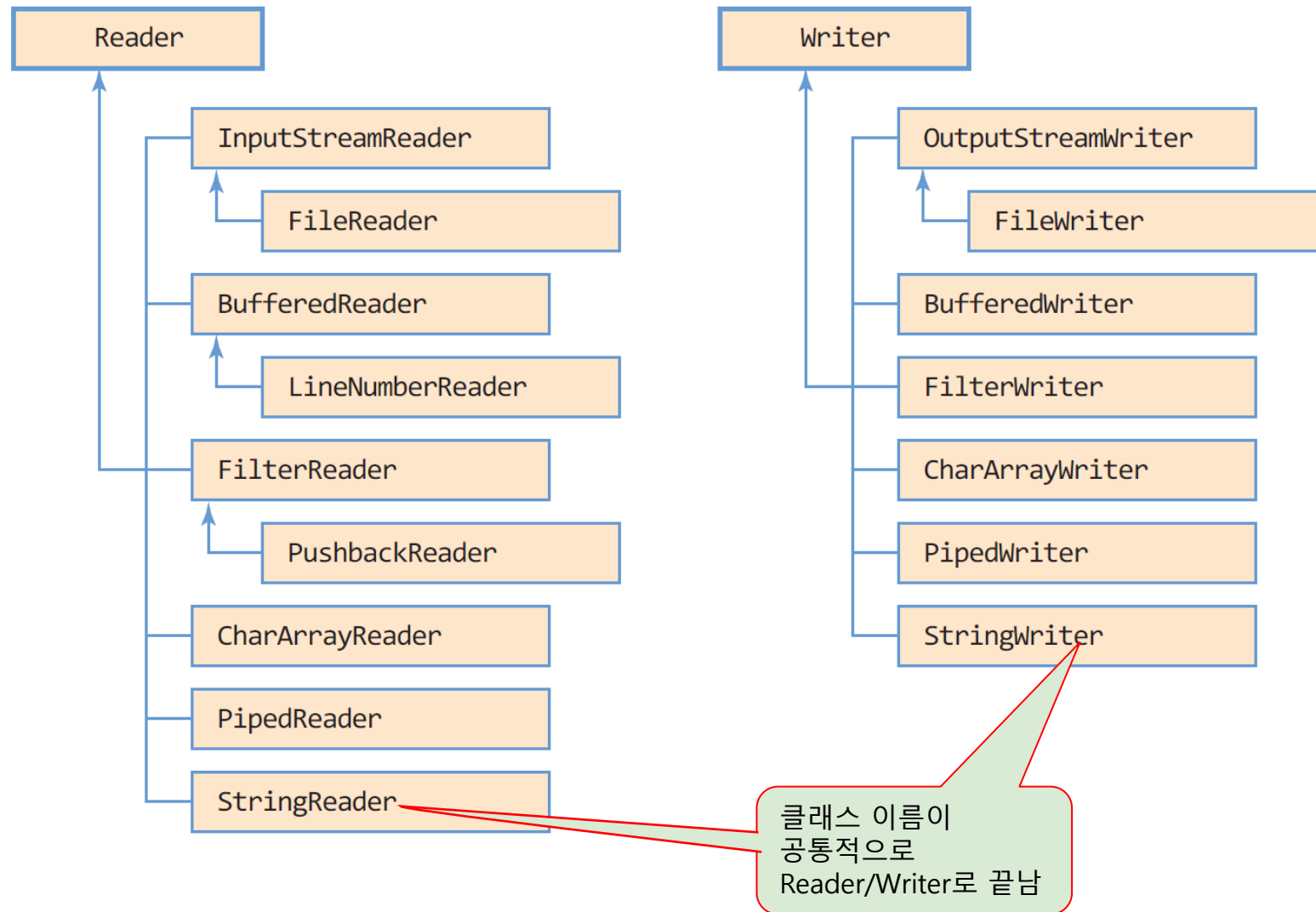
# 자바의 입출력 스트림 종류

- **바이트 스트림과 문자 스트림**
  - 바이트 스트림
    - 입출력되는 데이터를 단순 바이트로 처리
      - 예) 바이너리 파일을 읽는 입력 스트림
  - 문자 스트림
    - 문자만 입출력하는 스트림
    - 문자가 아닌 바이너리 데이터는 스트림에서 처리하지 못함
      - 예) 텍스트 파일을 읽는 입력 스트림
- **JDK는 입출력 스트림을 구현한 다양한 클래스 제공**
  - 다음 슬라이드

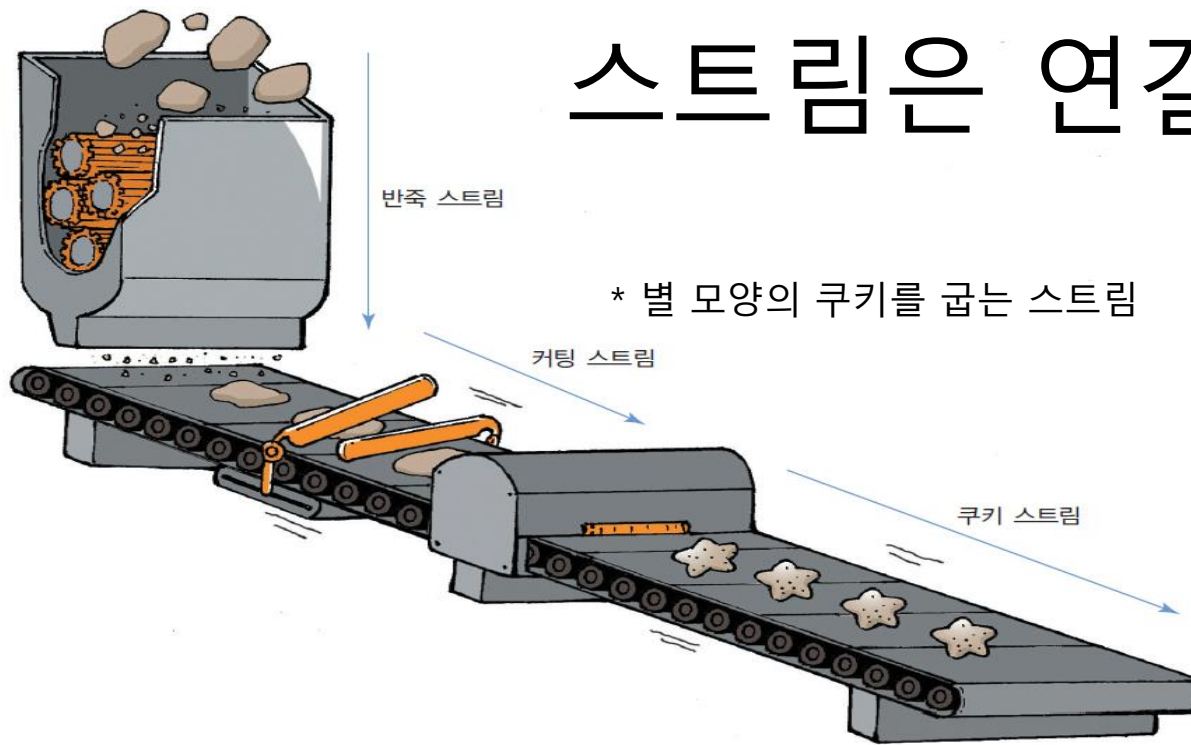
# JDK의 바이트 스트림 클래스 계층 구조



# JDK의 문자 스트림 클래스 계층 구조

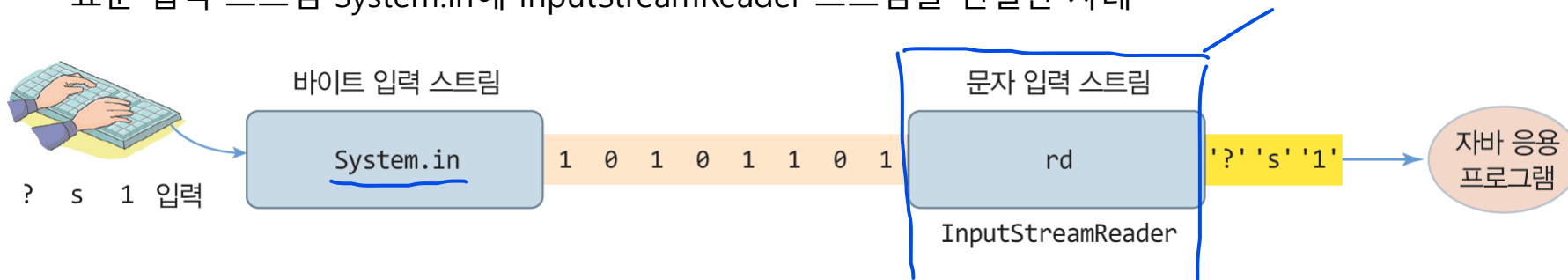


# 스트림은 연결될 수 있다



문자로 바꿔줌

\* 표준 입력 스트림 System.in에 InputStreamReader 스트림을 연결한 사례



```
InputStreamReader rd = new InputStreamReader(System.in);  
int c = rd.read(); // 키보드에서 문자 읽음
```

# 문자 스트림

## ■ 문자 스트림

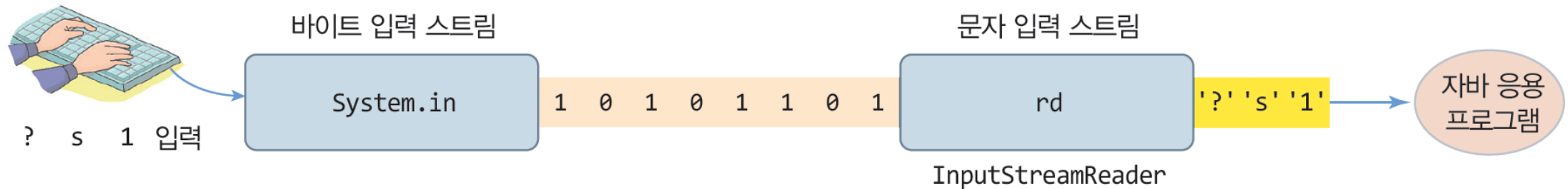
- 유니 코드(2바이트) 문자를 입출력 하는 스트림
  - 문자로 표현되지 않는 데이터는 다루지 못함
  - 이미지, 동영상과 같은 바이너리 데이터는 입출력 할 수 없음

## ■ 문자 스트림을 다루는 클래스

- Reader/Writer
- InputStreamReader/OutputStreamWriter
- FileReader/FileWriter
  - 텍스트 파일에서 문자 데이터 입출력

```
InputStreamReader isr = new InputStreamReader(System.in);  
BufferedReader in = new BufferedReader(isr);
```

System.in도 InputStream 클래스의 객체



출력 스트림의 경우에도 똑같이 *OutputStreamWriter()*와 *BufferedWriter()*가 존재

위와 똑같은 형태

```
InputStreamReader isr = new InputStreamReader(socket.getInputStream()); // 소켓 입력 스트림  
BufferedReader in = new BufferedReader(isr);      InputStream클래스의 객체
```

```
OutputStreamWriter osw = new OutputStreamWriter(socket.getOutputStream()); // 소켓 출력 스트림  
BufferedWriter out = new BufferedWriter(osw);
```

해보기 : 클라이언트에서 한 줄씩 텍스트를 입력 받아 서버로 보내고, 서버는 받은 텍스트를 출력하는 소켓 프로그램을 작성하세요. 클라이언트가 "bye"를 전송하면 클라이언트와 서버 모두 연결을 끊고 종료

```
C:\wjavatest>java EchoServer
서버입니다. 클라이언트를 기다립니다...
연결되었습니다.
... 안녕 자바 프로그램을 작성중이야
... 굉장히 재미있어
... 소켓 프로그래밍이 더 그래
... 다했다. 이제 자야겠어
접속을 종료합니다.

C:\wjavatest>_
```

```
C:\wjavatest>java EchoClient
서버에 접속하였습니다...
텍스트입력 >> 안녕 자바 프로그램을 작성중이야
텍스트입력 >> 굉장히 재미있어
텍스트입력 >> 소켓 프로그래밍이 더 그래
텍스트입력 >> 다했다. 이제 자야겠어
텍스트입력 >> 끝
연결을 종료합니다.

C:\wjavatest>
```



# Q&A

담당교수 : 신성

E-mail : [sihns@hansung.ac.kr](mailto:sihns@hansung.ac.kr)

연구실 : 우촌관 702호

휴대폰 번호 : 010-8873-8353