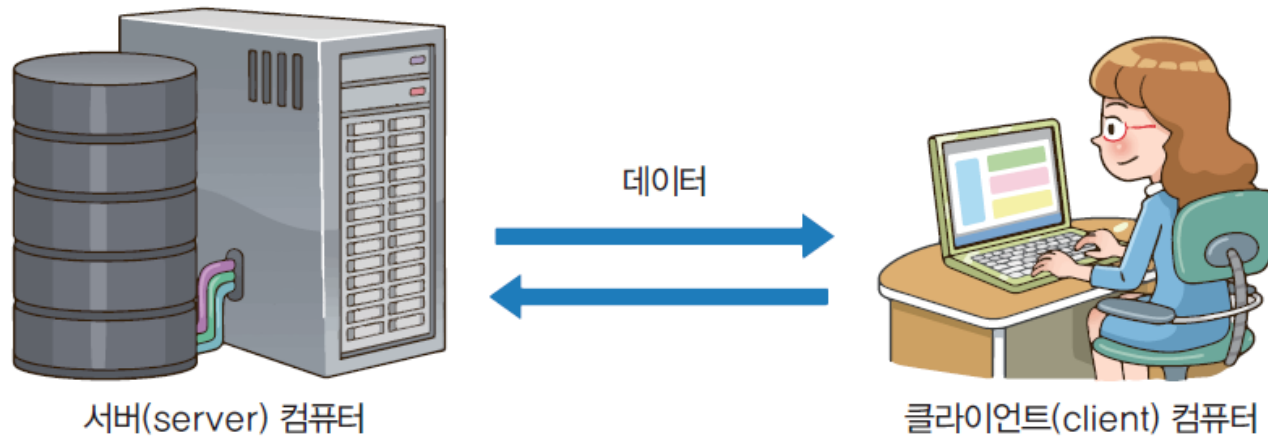




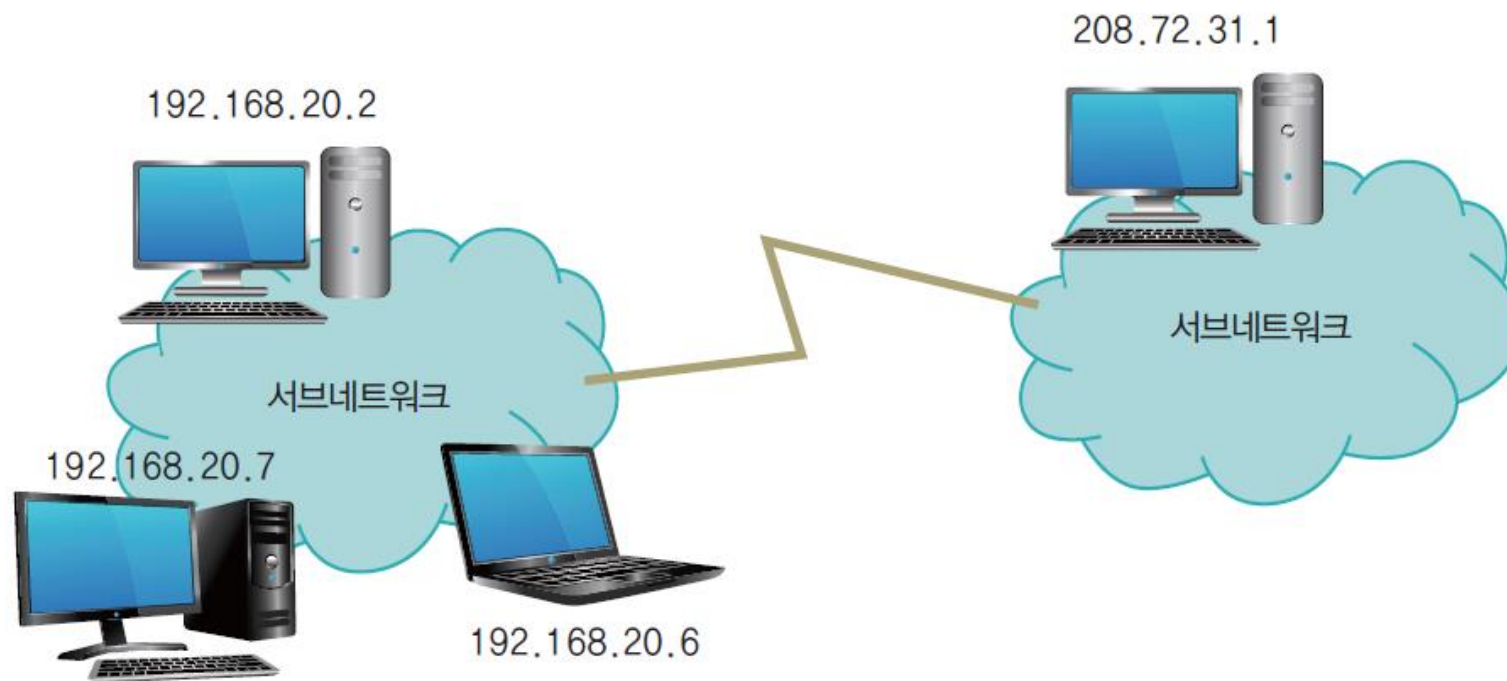
서버와 클라이언트

- 서버(Server): 사용자들에게 서비스를 제공하는 컴퓨터
- 클라이언트(Client): 서버에게 서비스를 요청해서 사용하는 컴퓨터
- (예) 웹서버와 클라이언트



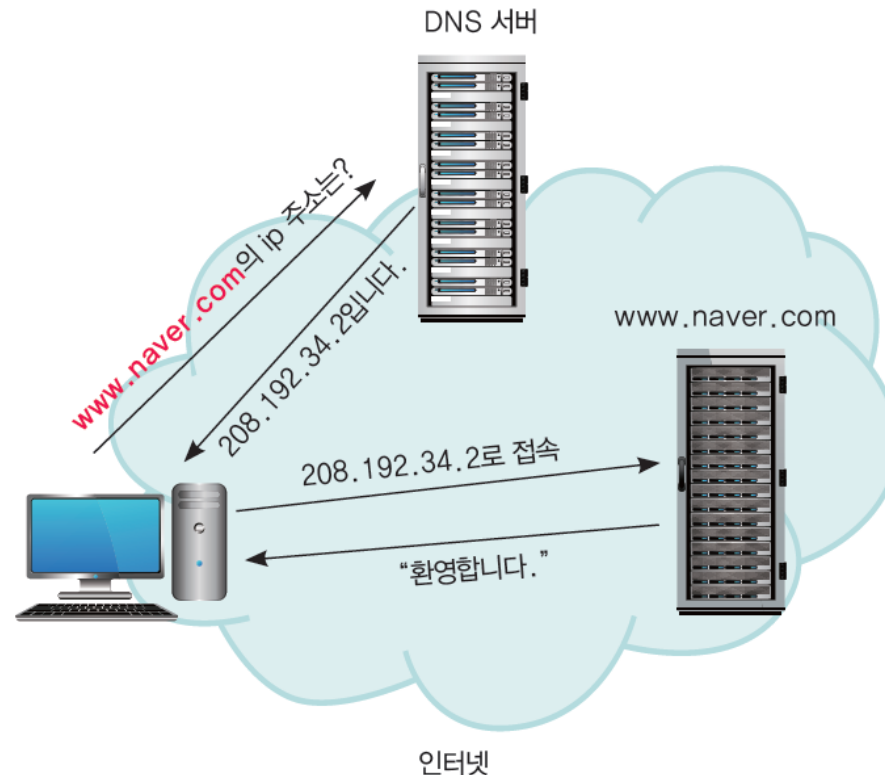
IP 주소

- IP 주소: 인터넷에서 컴퓨터의 주소



호스트 이름, DNS, URL

- DNS(Domain Name System): 숫자 대신 기호를 사용하는 주소
- DNS 서버: 기호 주소를 숫자 주소가 변환해주는 서버
- URL(Uniform Resource Locator): 인터넷 상의 자원을 나타내는 약속



URL

- **URL(Uniform Resource Locator)**은 인터넷 상의 파일이나 데이터베이스같은 자원에 대한 주소를 지정하는 방법이다



예제

```
public class host2ip
{
    public static void main ( String[] args ) throws IOException
    {
        String hostname = "www.naver.com";
        try
        {
            InetAddress address = InetAddress.getByName(hostname);
            System.out.println("IP 주소: " + address.getHostAddress());
        }
        catch ( UnknownHostException e )
        {
            System.out.println(hostname + "의 IP 주소를 찾을 수 없습니다. ");
        }
    }
}
```



IP 주소: 125.209.222.142

웹에서 파일 다운로드

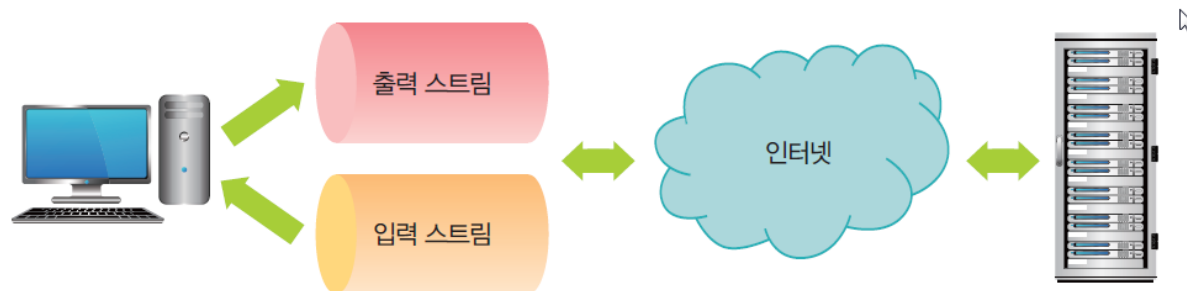
- **java.net.URL을 이용하여** 우리의 프로그램과 인터넷 상의 원격 컴퓨터를 연결한다. 그리고 원격 컴퓨터가 가지고 있는 자원에 접근할 수 있다.

전체적인 구조



형식

```
try {  
    URL testURL = new URL("http://www.naver.com/");  
    // 여기에 더 많은 코드가 들어간다.  
  
} catch (MalformedURLException e){  
    // 예외 처리  
}
```



예제 (네이버 파일 받기)

```
public class URLConnectionReader {  
    public static void main(String[] args) throws Exception {  
        URL site = new URL("http://www.naver.com/");  
        URLConnection url = site.openConnection();  
        BufferedReader in = new BufferedReader(  
            new InputStreamReader(  
                url.getInputStream()));  
  
        String inLine;  
  
        while ((inLine = in.readLine()) != null)  
            System.out.println(inLine);  
        in.close();  
    }  
}
```



```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"  
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">  
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="ko" lang="ko">  
<head>  
<meta http-equiv="Content-Type" content="text/html; charset=euc-kr" />  
...
```


예제(구글 파일받기)

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

public class HttpURLTest {
    public static void main(String[] args) throws Exception {
        HttpURLTest http = new HttpURLTest();

        String site = "http://www.google.com/search?q=java";

        URL url = new URL(site);
        HttpURLConnection conn = (HttpURLConnection) url.openConnection();

        conn.setRequestMethod("GET");
        conn.setRequestProperty("User-Agent", "Mozilla/5.0");

        int resCode = conn.getResponseCode();

        BufferedReader in = new BufferedReader(new InputStreamReader(
                                                    conn.getInputStream()));

        String inputLine;
        StringBuffer output = new StringBuffer();

        while((inputLine = in.readLine()) != null) {
            output.append(inputLine);
        }
        in.close();

        System.out.println(output);
    }
}
```

Lab: 웹에서 이미지 파일 다운로드하기

- 웹에 있는 특정한 이미지 파일을 한정된 버퍼를 사용하여 다운로드하는 프로그램을 작성하여 보자. 버퍼의 크기는 2048 바이트로 한다.



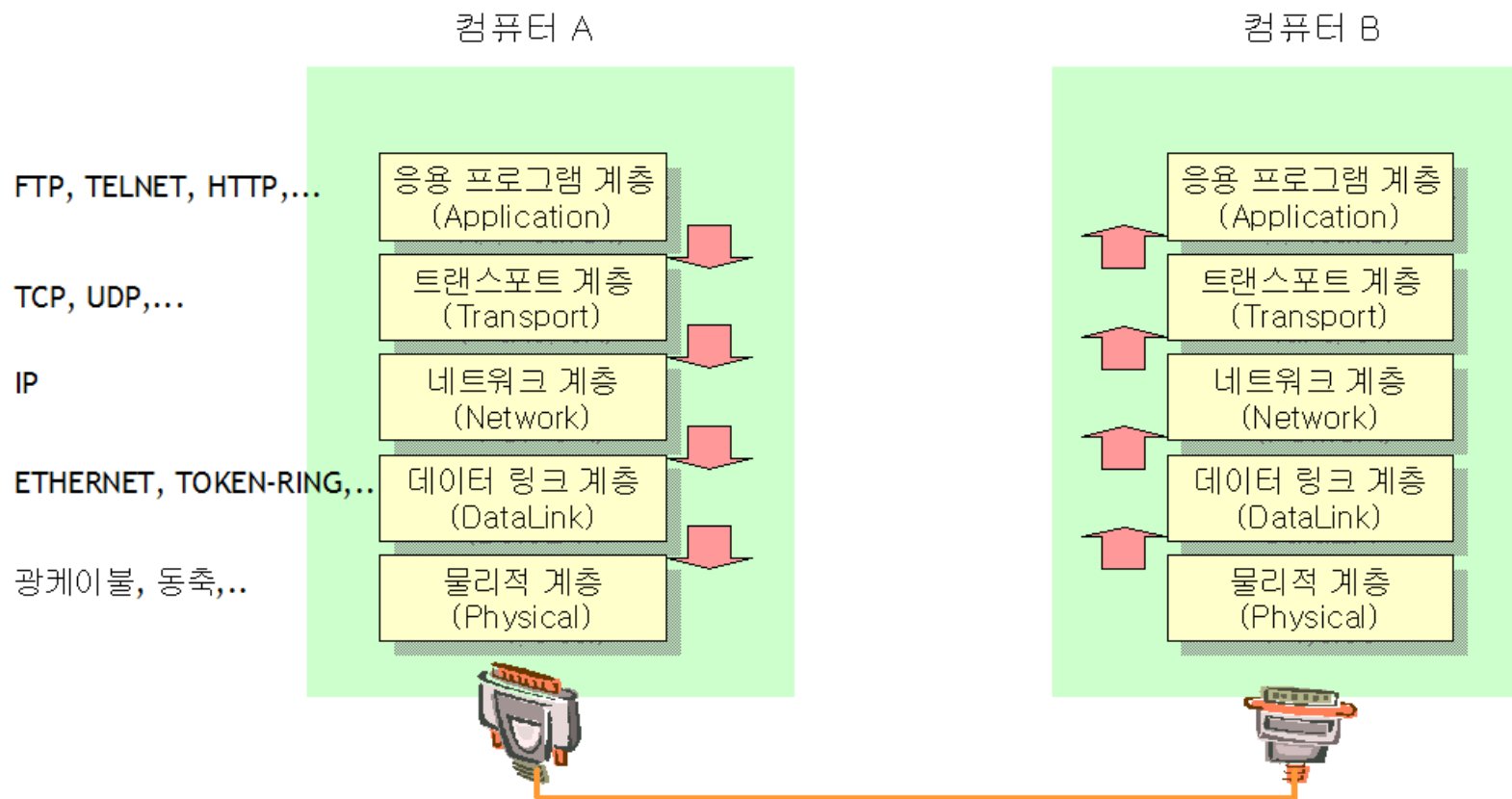
<http://www.oracle.com/us/hp07-bg121314-openworld-2x-2280475.jpg> 사이트에서 이
미지를 다운로드합니다.
2048바이트 만큼 읽었음!
2048바이트 만큼 읽었음!
...
1924바이트 만큼 읽었음!

예제

```
public class DownloadImage {  
    public static void main(String[] args) throws Exception {  
        String website = "http://www.oracle.com/us/hp07-bg121314-openworld-2x-2280475.jpg";  
        System.out.println("" + website + "사이트에서 이미지를 다운로드합니다.");  
        URL url = new URL(website);  
        byte[] buffer = new byte[2048];  
        try (InputStream in = url.openStream();  
             OutputStream out = new FileOutputStream("test.jpg");) {  
            int length = 0;  
            while ((length = in.read(buffer)) != -1) {  
                System.out.println("" + length + "바이트 만큼 읽었음!");  
                out.write(buffer, 0, length);  
            }  
            in.close();  
            out.close();  
        } catch (Exception e) {  
            System.out.println("예외: " + e.getMessage());  
        }  
    }  
}
```


프로토콜

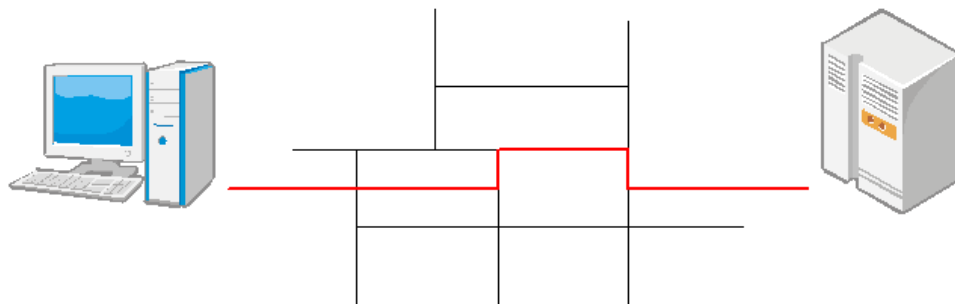
- 프로토콜(protocol): 통신을 하기 위한 약속



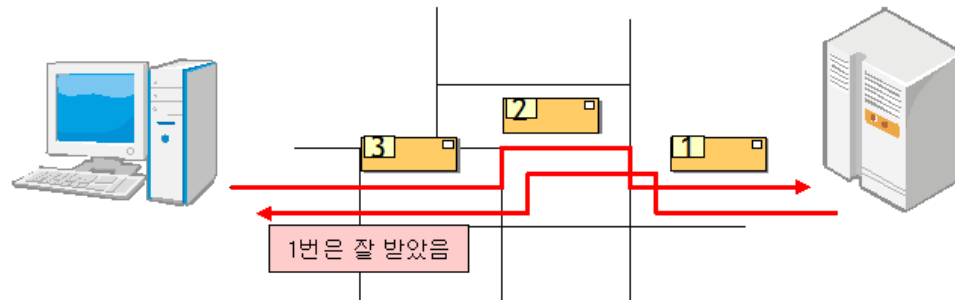
TCP

- TCP(Transmission Control Protocol)는 신뢰성있게 통신하기 위하여 먼저 서로 간에 연결을 설정한 후에 데이터를 보내고 받는 방식

(1) 먼저 가능한 경로 중에서 하나가 결정된다.



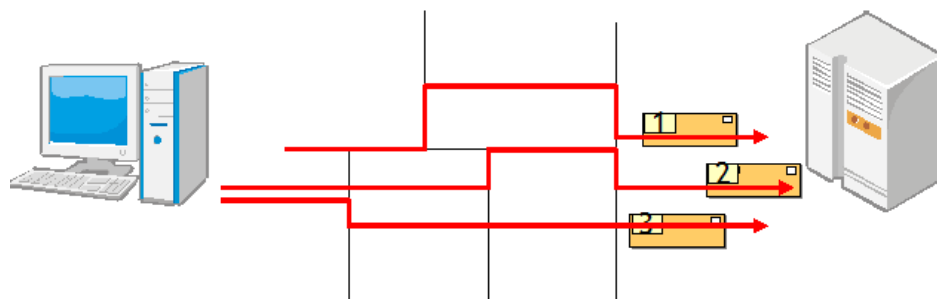
(2) 데이터는 패킷으로 나누어지고 패킷에 주소 붙여서 전송한다.



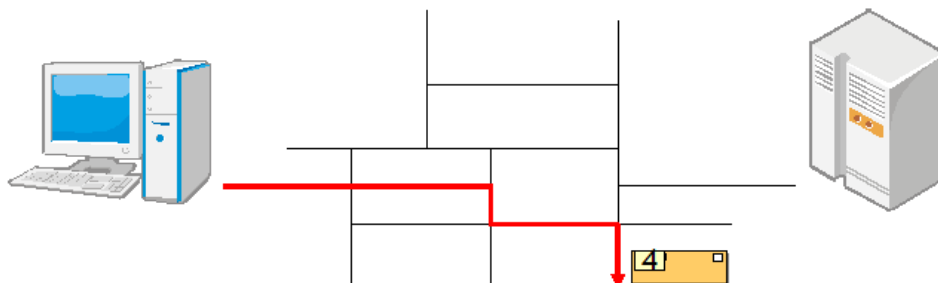
UDP

- UDP(User Datagram Protocol)는 데이터를 몇 개의 고정 길이의 패킷(데이터그램이라고 불린다)으로 분할하여 전송

(1) 데이터를 패킷으로 나누어서 패킷에 주소를 붙이고 전송한다.



(2) 패킷의 순서가 지켜지지 않으며 패킷이 분실될 수도 있다.



TCP/IP 소개

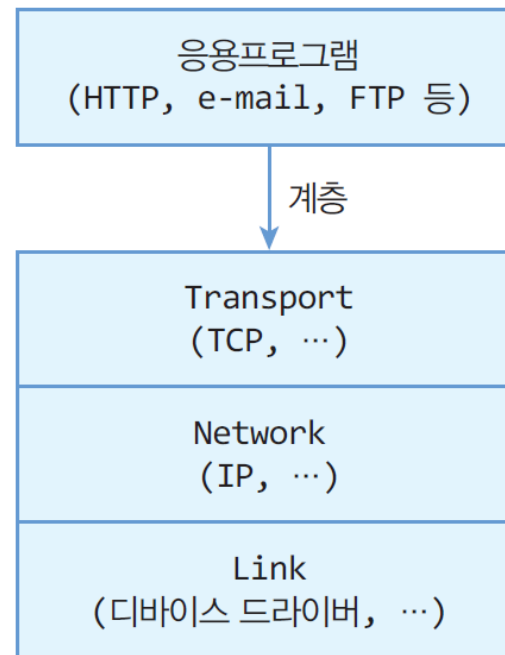
16

□ 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

□ IP

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



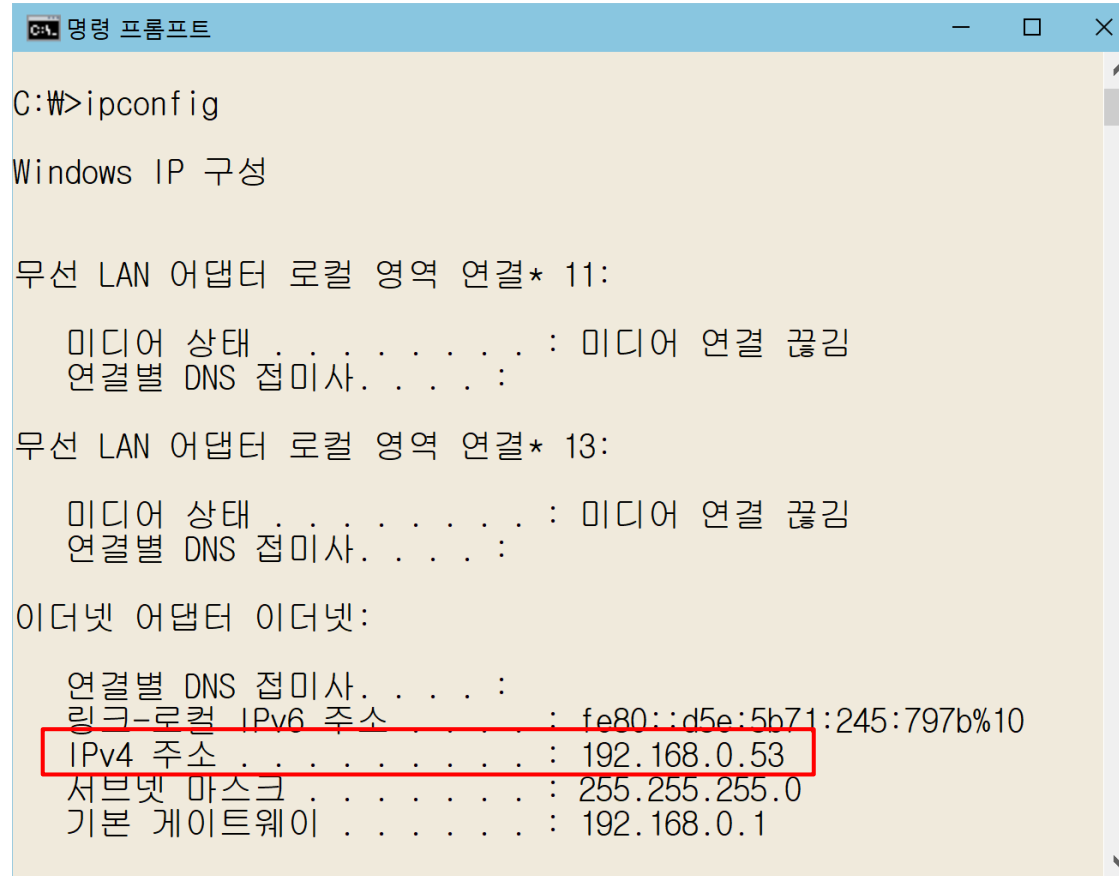
□ IP 주소

- ▣ 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

18

- 내 컴퓨터의 윈도우에서 명령창을 열어 ipconfig 명령 수행



```
C:\>명령 프롬프트

C:\W>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
```

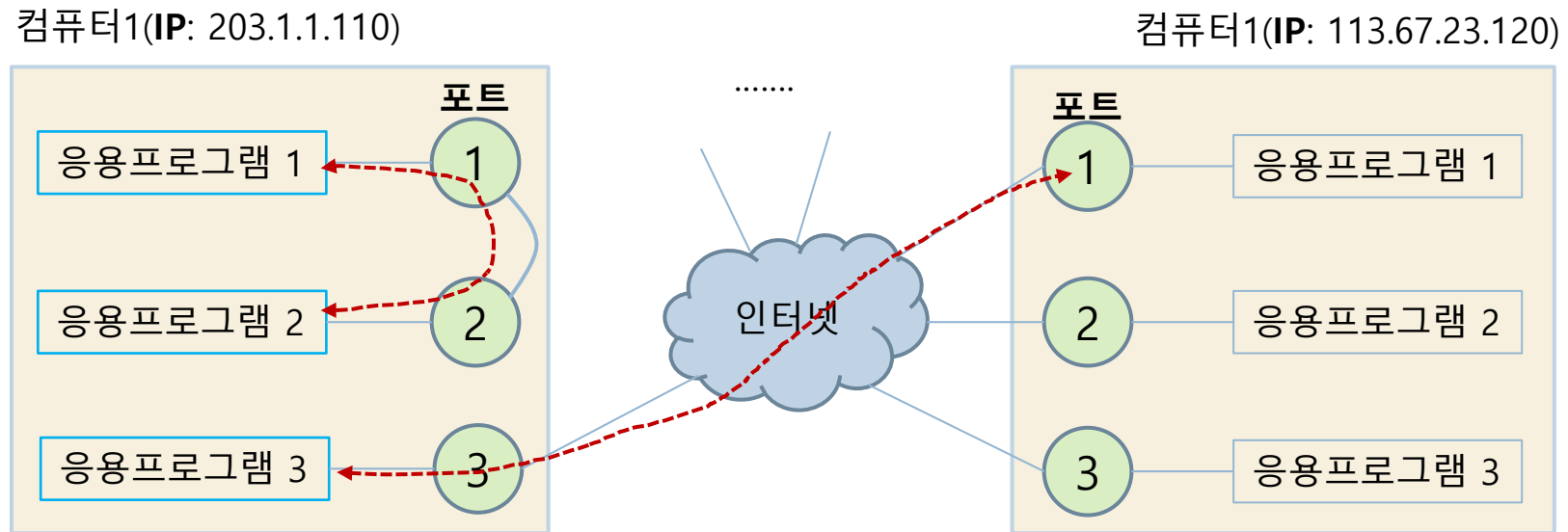
□ 포트

- 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 사용하는 포트 번호
 - 0부터 1023 사이의 포트 번호
 - ex) SSH 23, HTTP 80, FTP 21
 - 잘 알려진 포트 번호는 개발자가 사용하지 않는 것이 좋음
 - 충돌 가능성 있음



포트를 이용한 통신

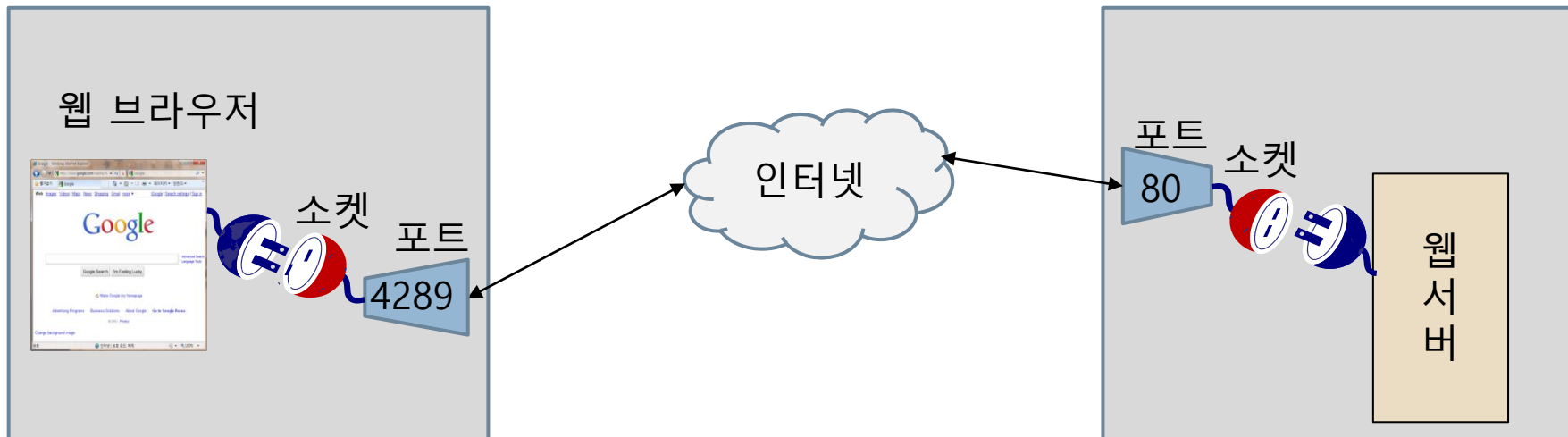
20



소켓 프로그래밍

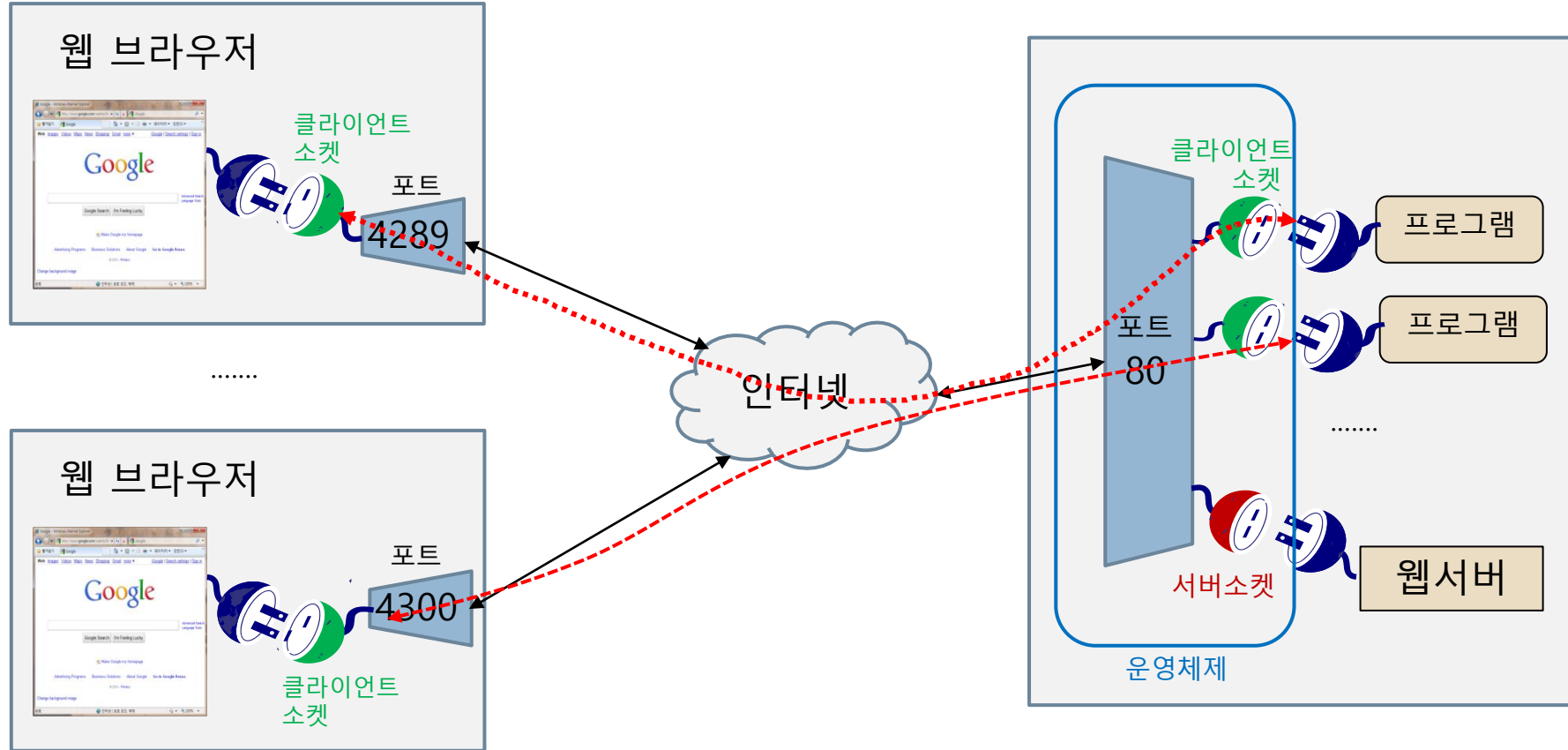
21

- 소켓 (socket)
 - ▣ 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



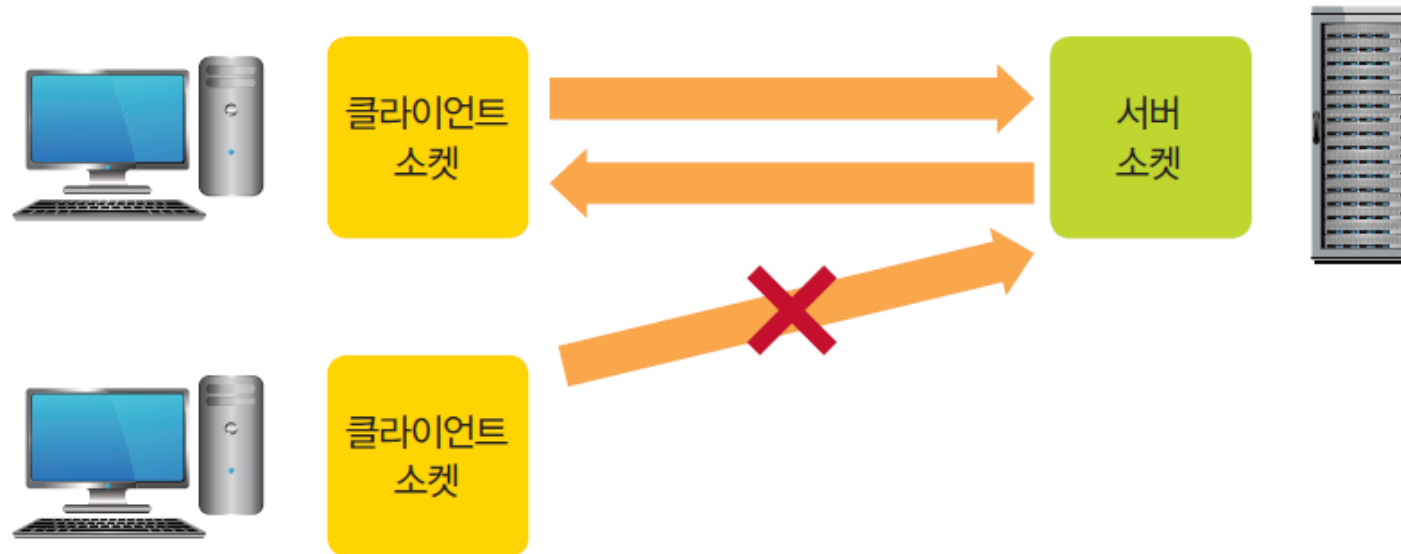
소켓을 이용한 웹 서버와 클라이언트 사이의 통신 사례

22

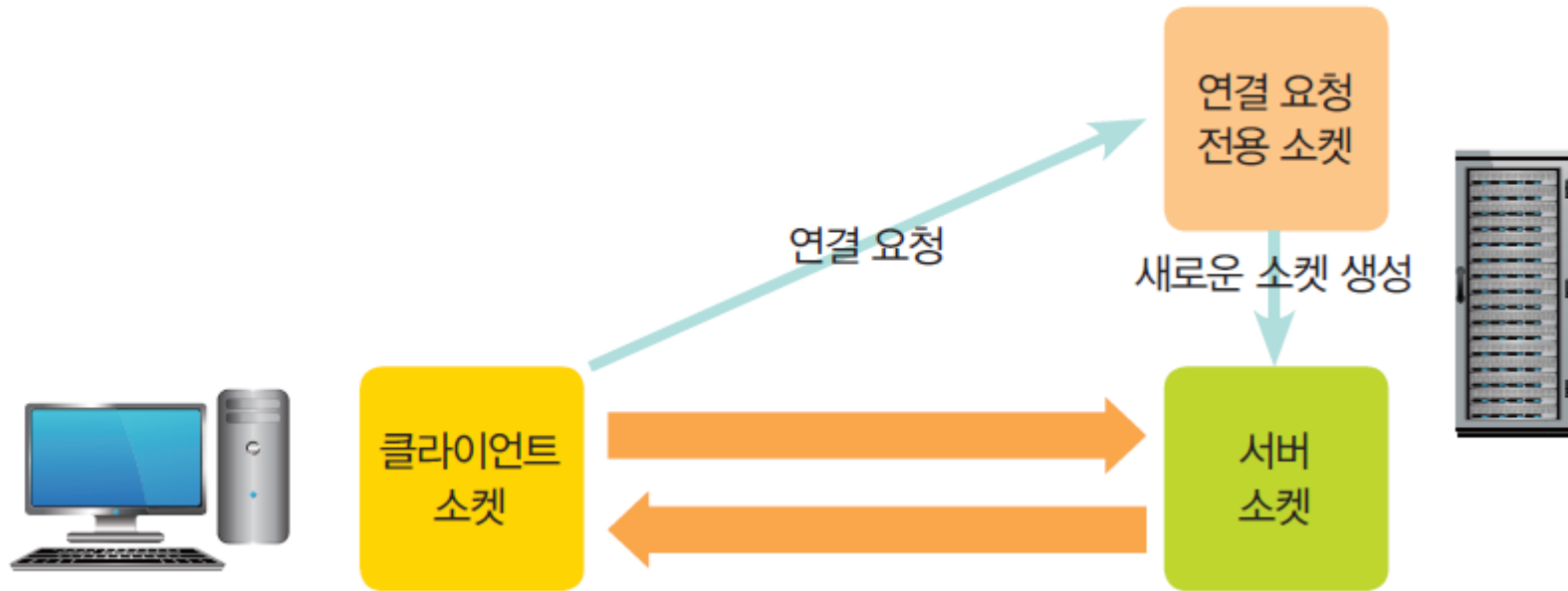


서버와 클라이언트 제작

- 서버가 하나의 소켓만을 사용한다면 문제가 발생한다.

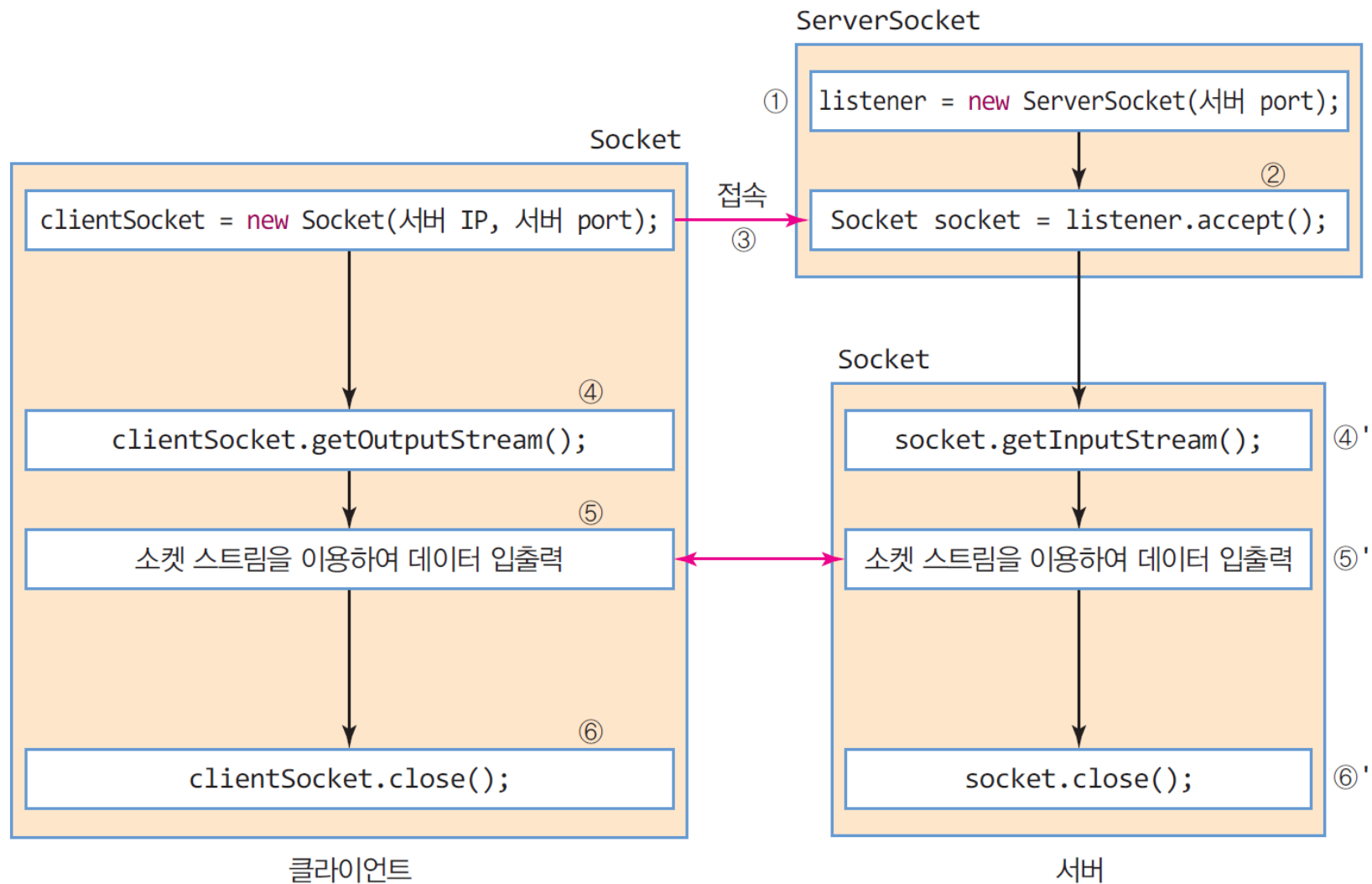


서버는 연결 요청만을 받는 소켓을 따로 가지고 있다.



소켓을 이용한 서버 클라이언트 통신 프로그램의 전형적인 구조

25



Socket 클래스, 클라이언트 소켓

26

- 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

27

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

28

- 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();  
//서버로부터 한 행의 문자열 수신
```

- 네트워크 접속 종료

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

ServerSocket 클래스, 서버 소켓

29

□ ServerSocket 클래스

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

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

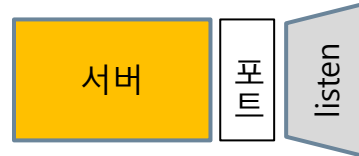
▣ 메소드

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

30

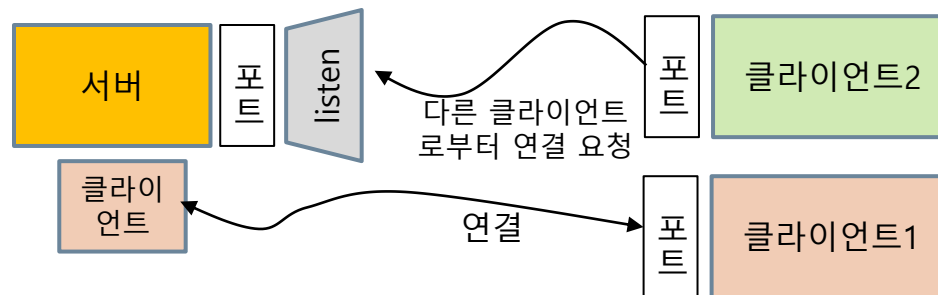
- 서버는 서버 소켓으로 들어오는 연결 요청을 기다림(listen)



- 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

31

▣ 서버 소켓 생성

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

- 서버는 접속을 기다리는 포트로 5550 선택

▣ 클라이언트로부터 접속 기다림

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

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

▣ 네트워크 입출력 스트림 생성

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

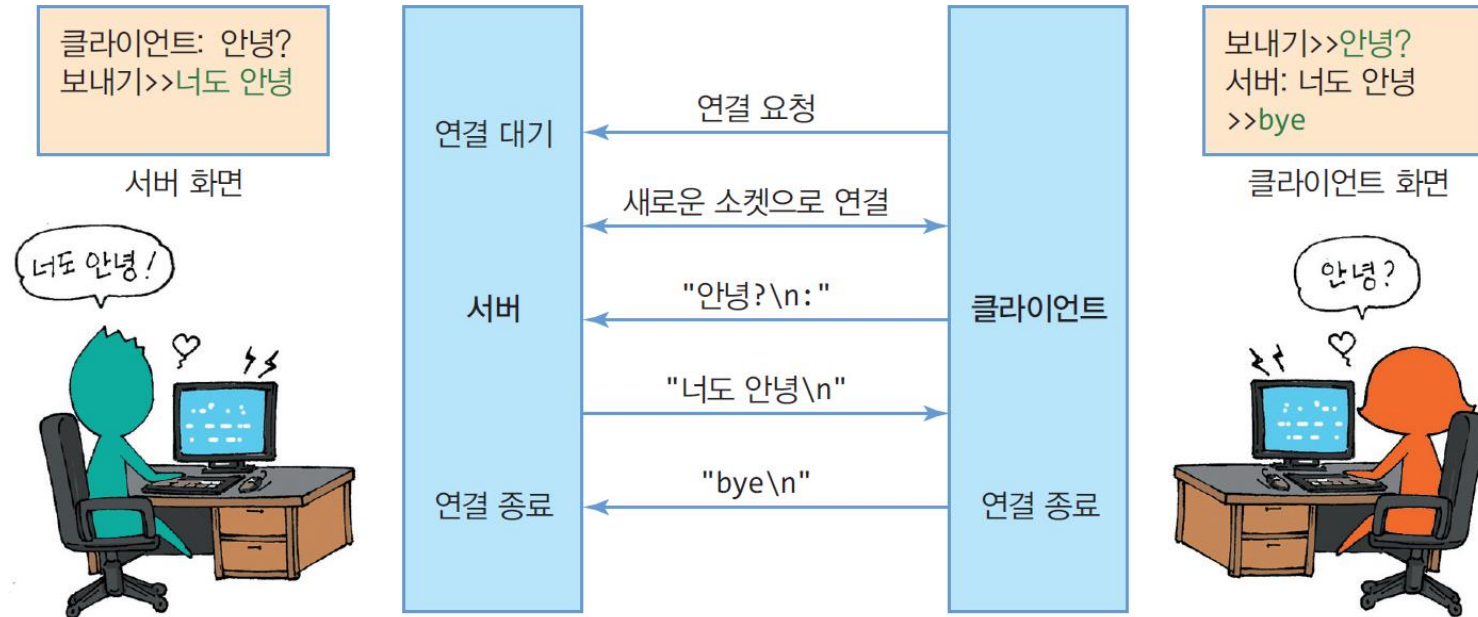
- Socket 객체의 getInputStream()과 getOutputStream() 메소드를 이용하여 입출력 데이터 스트림 생성

소켓을 이용한 서버/클라이언트 채팅 예제

32

□ 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

```
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()));
            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

```
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.");
            }
        }
    }
}
```

수식 계산 서버-클라이언트 만들기 실습

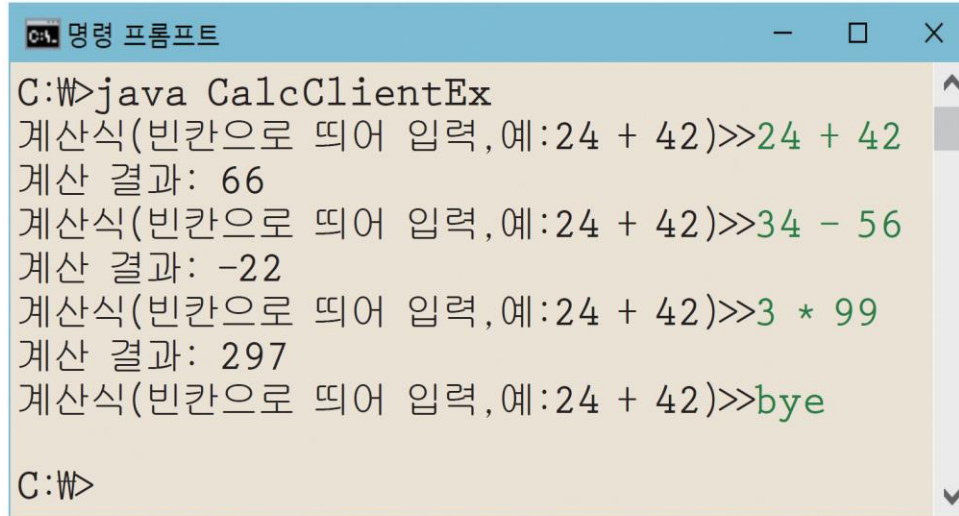
35

□ 문제 개요

- ▣ 서버 클라이언트는 1:1 통신
- ▣ 서버를 먼저 실행시키고 클라이언트를 실행시켜 서버에 접속
- ▣ 클라이언트는 사용자로부터 수식을 입력 받아 서버로 전송
- ▣ 연산자는 +, -, *의 3가지만 허용하고 정수 연산만 가능
- ▣ 서버가 식을 받으면 식을 서버의 화면에 출력하고, 계산하여 결과를 클라이언트로 전송
- ▣ 클라이언트는 서버로부터 받은 답을 화면에 출력
- ▣ 클라이언트와 서버는 전송할 데이터를 문자열로 만들고 "\n"을 덧붙여 전송하며, 라인 단위로 송수신
- ▣ 클라이언트가 "bye"를 보내면 양쪽 모두 종료

실행 예시

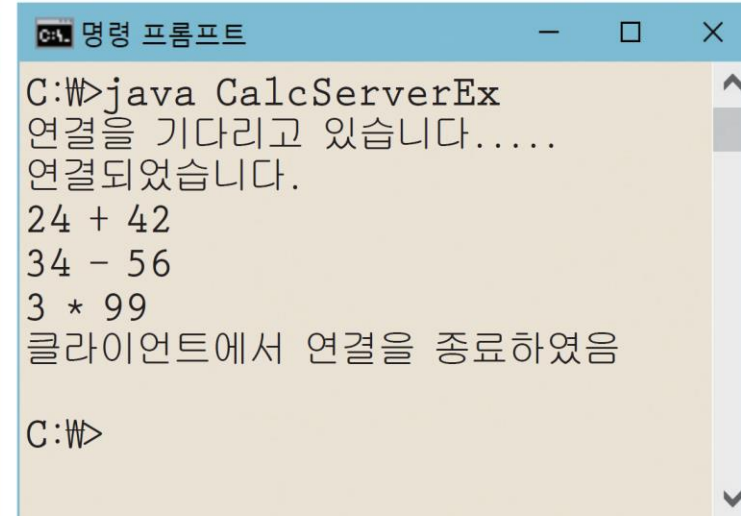
36



```
명령 프롬프트
C:\W>java CalcClientEx
계산식(빈칸으로 띄어 입력, 예: 24 + 42)>>24 + 42
계산 결과: 66
계산식(빈칸으로 띄어 입력, 예: 24 + 42)>>34 - 56
계산 결과: -22
계산식(빈칸으로 띄어 입력, 예: 24 + 42)>>3 * 99
계산 결과: 297
계산식(빈칸으로 띄어 입력, 예: 24 + 42)>>bye

C:\W>
```

(a) 계산 클라이언트의 실행



```
명령 프롬프트
C:\W>java CalcServerEx
연결을 기다리고 있습니다.....
연결되었습니다.
24 + 42
34 - 56
3 * 99
클라이언트에서 연결을 종료하였음

C:\W>
```

(b) 계산 서버의 실행

서버 프로그램 CalcServerEx.java

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

public class CalcServerEx {
    public static String calc(String exp) {
        StringTokenizer st = new StringTokenizer(exp, " ");
        if (st.countTokens() != 3) return "error";
        String res="";
        int op1 = Integer.parseInt(st.nextToken());
        String opcode = st.nextToken();
        int op2 = Integer.parseInt(st.nextToken());
        switch (opcode) {
            case "+": res = Integer.toString(op1 + op2);
                break;
            case "-": res = Integer.toString(op1 - op2);
                break;
            case "*": res = Integer.toString(op1 * op2);
                break;
            default : res = "error";
        }
        return res;
    }

    public static void main(String[] args) {
        BufferedReader in = null;
        BufferedWriter out = null;
        ServerSocket listener = null;
        Socket socket = null;
```

```
    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()));
        while (true) {
            String inputMessage = in.readLine();
            if (inputMessage.equalsIgnoreCase("bye")) {
                System.out.println("클라이언트에서 연결을 종료하였음");
                break; // "bye"를 받으면 연결 종료
            }
            System.out.println(inputMessage); // 받은 메시지를 화면에 출력
            String res = calc(inputMessage); // 계산. 계산 결과는 res
            out.write(res + "\n"); // 계산 결과 문자열 전송
            out.flush();
        }
    } catch (IOException e) {
        System.out.println(e.getMessage());
    } finally {
        try {
            if(socket != null) socket.close(); // 통신용 소켓 닫기
            if(listener != null) listener.close(); // 서버 소켓 닫기
        } catch (IOException e) {
            System.out.println("클라이언트와 채팅 중 오류가 발생했습니다.");
        }
    }
}
```

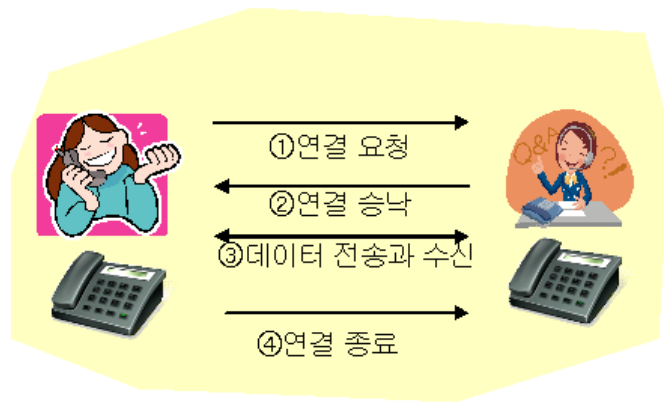
클라이언트 프로그램 CalcClientEx.java

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

public class CalcClientEx {
    public static void main(String[] args) {
        BufferedReader in = null;
        BufferedWriter out = null;
        Socket socket = null;
        Scanner scanner = new Scanner(System.in);
        try {
            socket = new Socket("localhost", 9999);
            in = new BufferedReader(new InputStreamReader(socket.getInputStream()));
            out = new BufferedWriter(new OutputStreamWriter(socket.getOutputStream()));
            while (true) {
                System.out.print("계산식(빈칸으로 띄어 입력,예:24 + 42)>>"); // 프롬프트
                String outputMessage = scanner.nextLine(); // 키보드에서 수식 읽기
                if (outputMessage.equalsIgnoreCase("bye")) {
                    out.write(outputMessage+"\\n"); // "bye" 문자열 전송
                    out.flush();
                    break; // 사용자가 "bye"를 입력한 경우 서버로 전송 후 연결 종료
                }
                out.write(outputMessage + "\\n"); // 키보드에서 읽은 수식 문자열 전송
                out.flush();
                String inputMessage = in.readLine(); // 서버로부터 계산 결과 수신
                System.out.println("계산 결과: " + 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.");
            }
        }
    }
}
```


UDP를 이용한 서버와 클라이언트

- DatagramSocket 클래스
 - ▣ DatagramSocket()은 UDP 프로토콜을 사용하는 소켓을 생성
- DatagramPacket 클래스
 - ▣ DatagramPacket()은 UDP 패킷을 생성한다.



TCP 프로토콜



UDP 프로토콜

Sender 클래스

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

public class Sender {
    public static void main(String[] args) throws IOException {

        DatagramSocket socket = null;
        socket = new DatagramSocket();
        String s = "우리는 여전히 우리 운명의 주인이다.";
        byte[] buf = s.getBytes();

        // "address"의 "port"에 있는 클라이언트에게 데이터를 보낸다.
        InetAddress address = InetAddress.getByName("127.0.0.1"); // 로컬 호스트
        DatagramPacket packet = new DatagramPacket(buf, buf.length, address,
            5000);
        socket.send(packet);
        socket.close();
    }
}
```

Receiver 클래스

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

public class Receiver {
    public static void main(String[] args) throws IOException {

        byte[] buf = new byte[256];

        DatagramSocket socket = new DatagramSocket(5000); // 포트 번호: 5000
        DatagramPacket packet = new DatagramPacket(buf, buf.length);
        socket.receive(packet);
        System.out.println(new String(buf));
    }
}
```

서버와 클라이언트의 실행

- 두개의 프로그램을 동시에 실행하여야 한다.



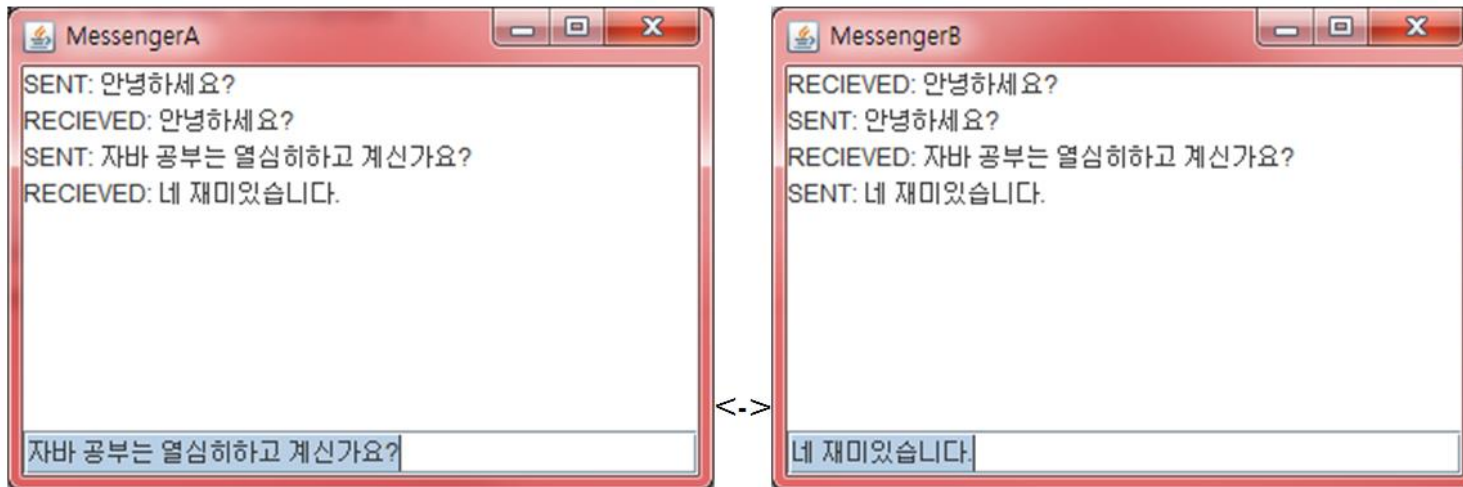
```
C> java Receiver
```

```
C> java Sender
```

우리는 여전히 우리 운명의 주인이다.

UDP를 이용한 서버와 클라이언트 작성

- 예제로 UDP 통신을 이용하여서 간단한 채팅을 할 수 있는 메신저를 작성하여 보자. 이 메신저는 정해진 상대와 텍스트를 주고 받을 수 있다.



MessengerA 클래스

```
public class MessengerA {  
    protected JTextField textField;  
    protected JTextArea textArea;  
    DatagramSocket socket;  
    DatagramPacket packet;  
    InetAddress address = null;  
    final int myPort = 5000;           // 수신용 포트 번호  
    final int otherPort = 6000; // 송신용 포트 번호  
  
    public MessengerA() throws IOException {  
        MyFrame f=new MyFrame();  
        address = InetAddress.getByName("127.0.0.1");  
        socket = new DatagramSocket(myPort);  
    }  
}
```

MessengerA 클래스

```
// 패킷을 받아서 텍스트 영역에 표시한다.  
public void process() {  
    while (true) {  
        try  
        {  
            byte[] buf = new byte[256];  
            packet = new DatagramPacket(buf, buf.length);  
            socket.receive(packet); // 패킷을 받는다.  
            // 받은 패킷을 텍스트 영역에 표시한다.  
            textArea.append("RECIEVED: " + new String(buf) + "\n");  
        }  
        catch (IOException ioException) {  
            ioException.printStackTrace();  
        }  
    }  
}
```

MessengerA 클래스

// 내부 클래스 정의

```
class MyFrame extends JFrame implements ActionListener {
```

```
    public MyFrame() {
```

```
        super("MessengerA");
```

```
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
```

```
        textField = new JTextField(30);
```

```
        textField.addActionListener(this);
```

```
        textArea = new JTextArea(10, 30);
```

```
        textArea.setEditable(false);
```

```
        add(textField, BorderLayout.PAGE_END);
```

```
        add(textArea, BorderLayout.CENTER);
```

```
        pack();
```

```
        setVisible(true);
```

```
    }
```

MessengerA 클래스

```
public void actionPerformed(ActionEvent evt) {
    String s = textField.getText();
    byte[] buffer = s.getBytes();
    DatagramPacket packet;

    // 패킷을 생성한다.
    packet = new DatagramPacket(buffer, buffer.length, address,
                                otherPort);

    try {
        socket.send(packet);           // 패킷을 보낸다.
    } catch (IOException e) {
        e.printStackTrace();
    }

    textArea.append("SENT: " + s + "\n");
    textField.selectAll();
    textArea.setCaretPosition(textArea.getDocument().getLength());
}

}

public static void main(String[] args) throws IOException {
    MessengerA m = new MessengerA();
    m.process();
}

}
```


MessengerB 클래스

// 다음의 몇 개의 문장만 제외하고 MessengerA와 동일

```
public class MessengerB {  
    ...  
    final int myPort = 6000;  
    final int otherPort = 5000;  
  
    public MessengerB() throws IOException {  
        ...  
    }  
    public static void main(String[] args) throws IOException {  
        MessengerB m = new MessengerB();  
        m.process();  
    }  
}
```