



TCP/IP 소개

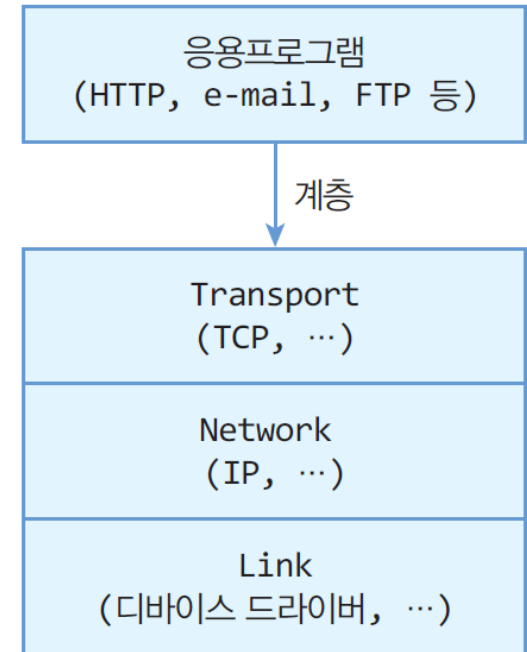
2

□ 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

3

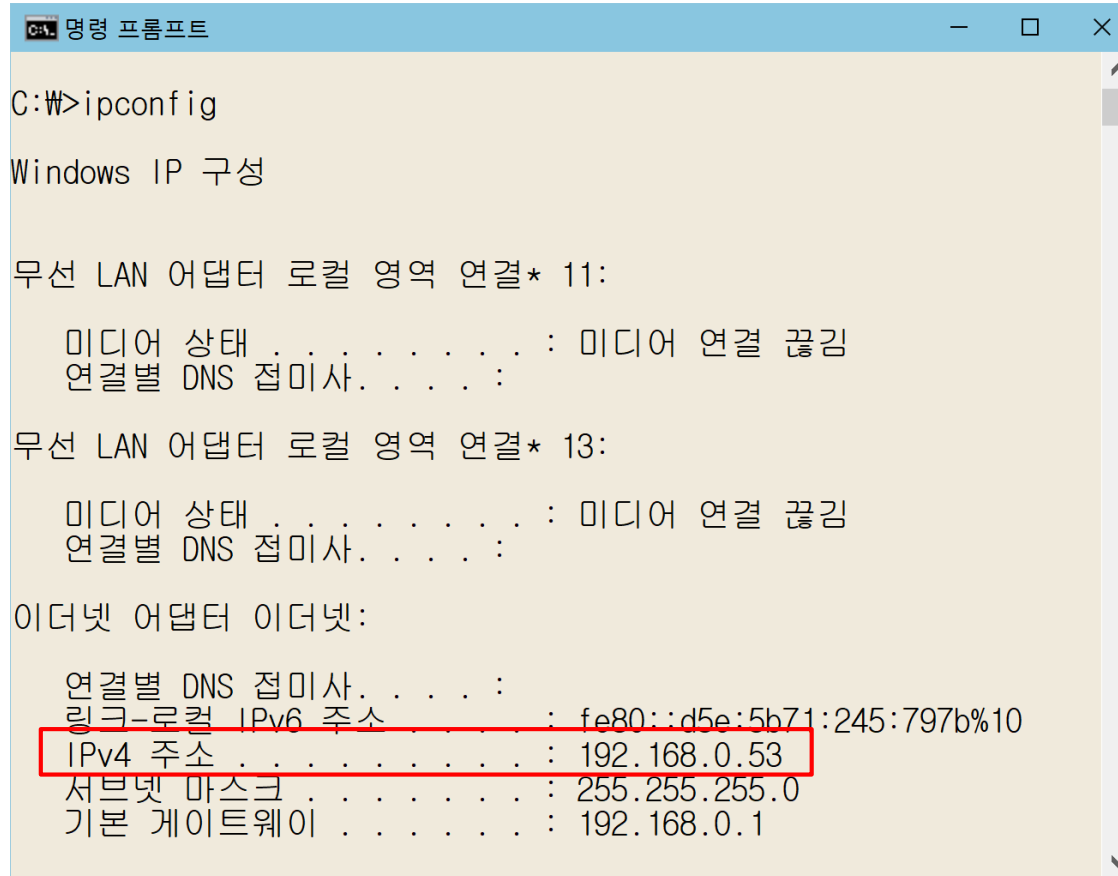
□ IP 주소

- 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

4

- 내 컴퓨터의 윈도우에서 명령창을 열어 ipconfig 명령 수행



```
C:\> 명령 프롬프트
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

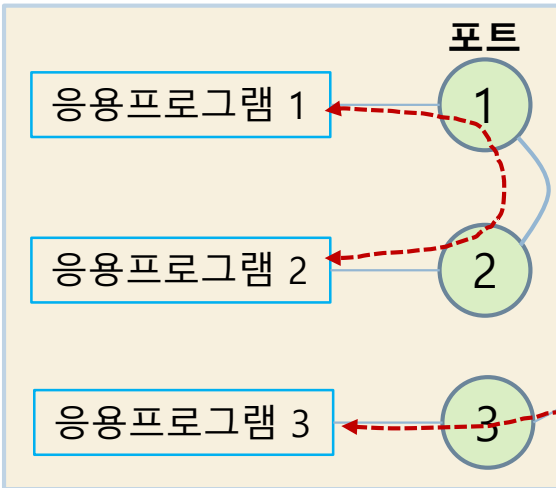
- 통신하는 프로그램 간에 가상의 연결단 포트 생성
 - IP 주소는 네트워크 상의 컴퓨터 또는 시스템을 식별하는 주소
 - 포트 번호를 이용하여 통신할 응용프로그램 식별
- 모든 응용프로그램은 하나 이상의 포트 생성 가능
 - 포트를 이용하여 상대방 응용프로그램과 데이터 교환
- 잘 알려진 포트(well-known ports)
 - 시스템이 사용하는 포트 번호
 - 잘 알려진 응용프로그램에서 사용하는 포트 번호
 - 0부터 1023 사이의 포트 번호
 - ex) SSH 23, HTTP 80, FTP 21
 - 잘 알려진 포트 번호는 개발자가 사용하지 않는 것이 좋음
 - 충돌 가능성 있음



포트를 이용한 통신

6

컴퓨터1(IP: 203.1.1.110)



컴퓨터1(IP: 113.67.23.120)

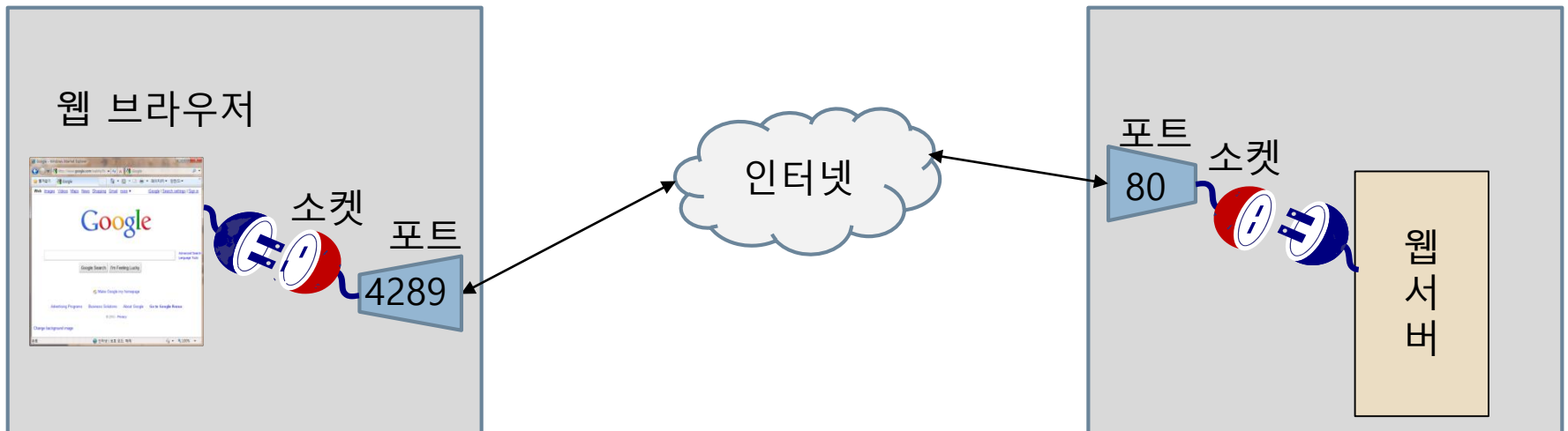


소켓 프로그래밍

7

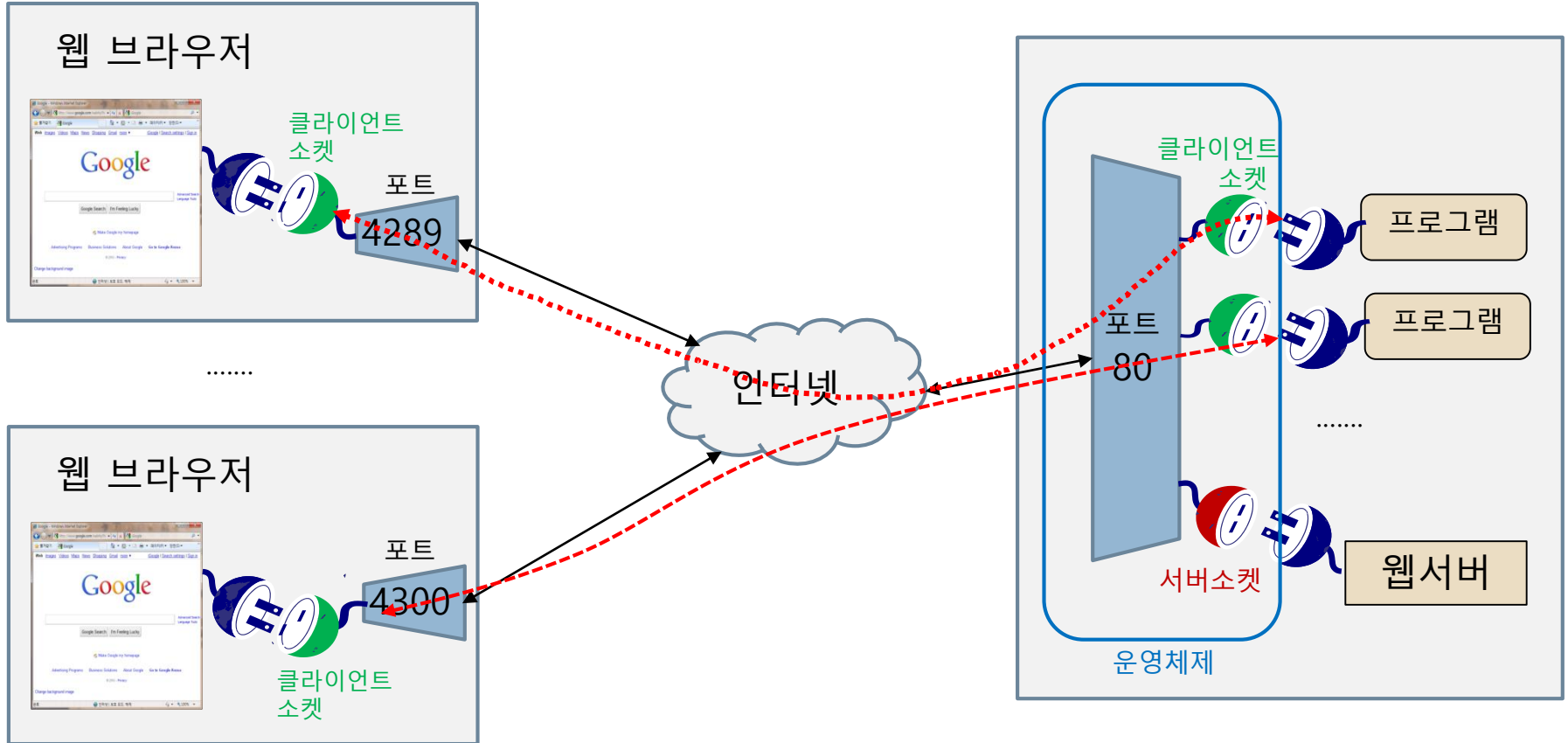
□ 소켓 (socket)

- 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

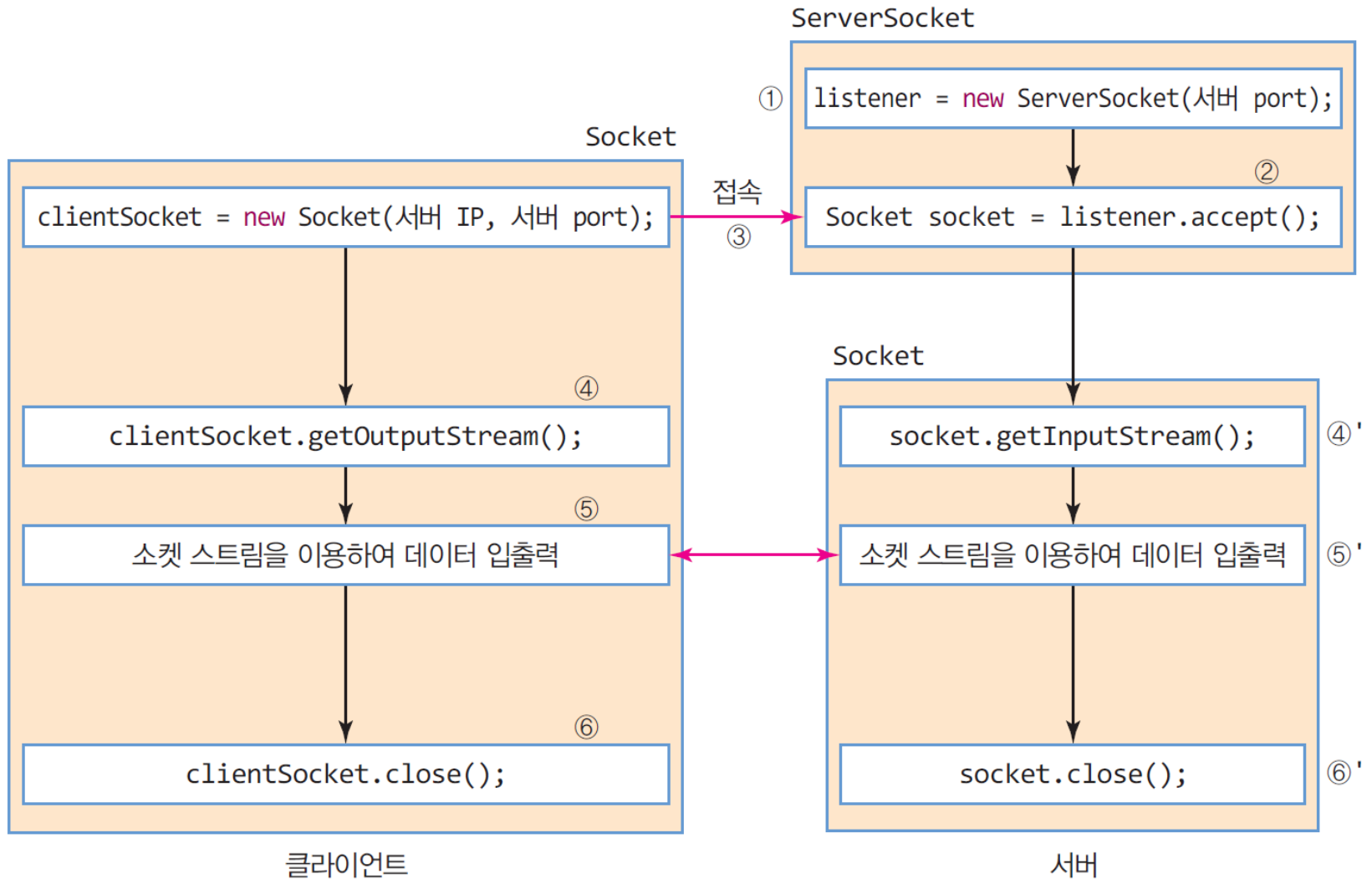


소켓을 이용한 웹 서버와 클라이언트 사이의 통신 사례

8



소켓을 이용한 서버 클라이언트 통신 프로그램의 전형적인 구조



Socket 클래스, 클라이언트 소켓

10

- 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

11

| 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

12

- 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

13

□ ServerSocket 클래스

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

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

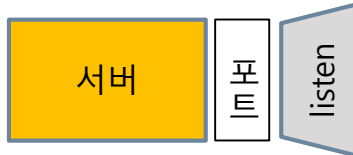
□ 메소드

| 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 |

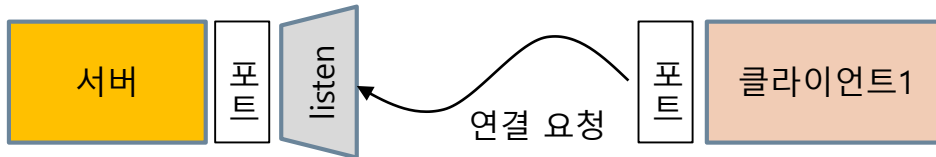
서버에 클라이언트가 연결되는 과정

14

- 서버는 서버 소켓으로 들어오는 연결 요청을 기다림(listen)

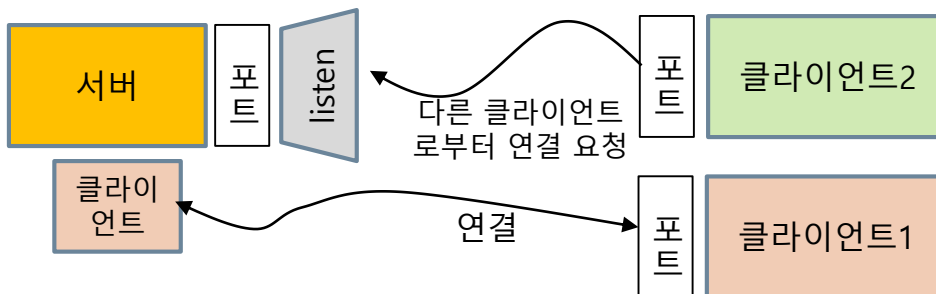


- 클라이언트가 서버에게 연결 요청



- 서버가 연결 요청 수락(accept)

- 새로운 클라이언트 소켓을 만들어 클라이언트와 통신하게 함
- 그리고 다시 다른 클라이언트의 연결을 기다림



서버가 클라이언트와 통신하는 과정

15

▣ 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()));
```

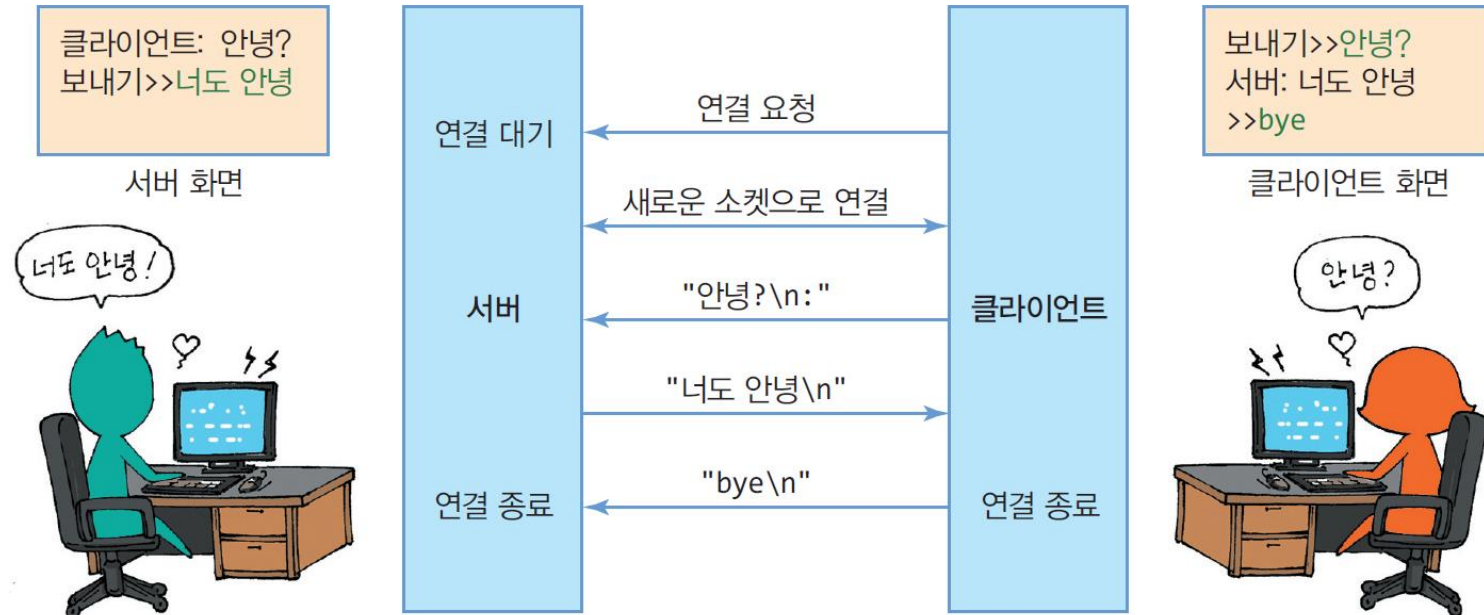
- Socket 객체의 getInputStream()과 getOutputStream() 메소드를 이용하여 입출력 데이터 스트림 생성

소켓을 이용한 서버/클라이언트 채팅 예제

16

□ 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 종료하였습니다.");
                    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

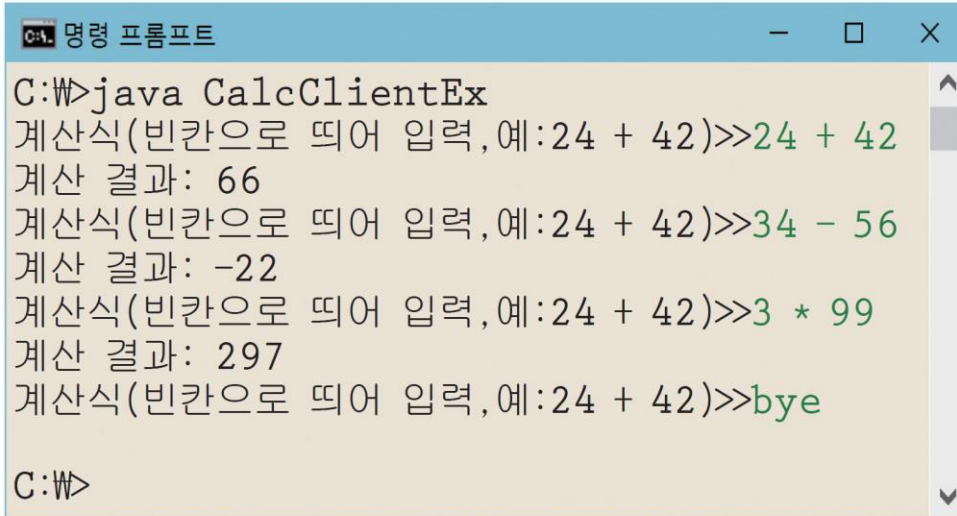
19

□ 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

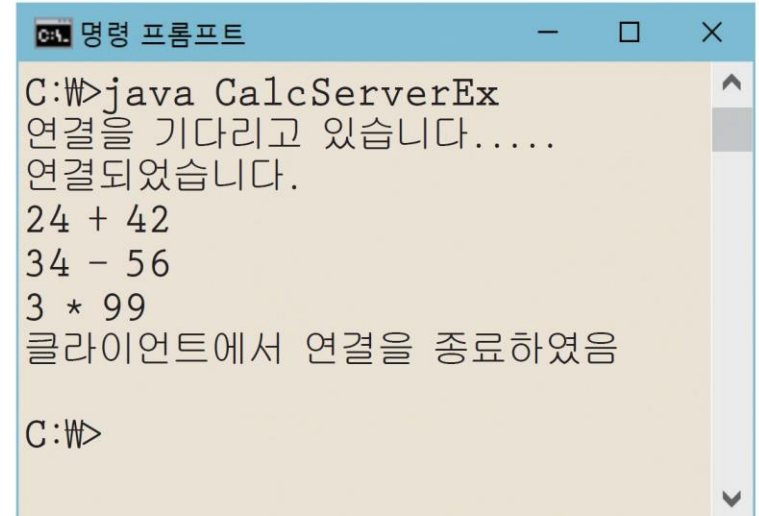
실행 예시

20



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