



사각형 캡처(R)

14

자바 소켓 프로그래밍

학습 목표

1. 소켓 통신에 대한 이해
2. 자바로 간단한 소켓 통신 프로그램 작성

TCP/IP 소개

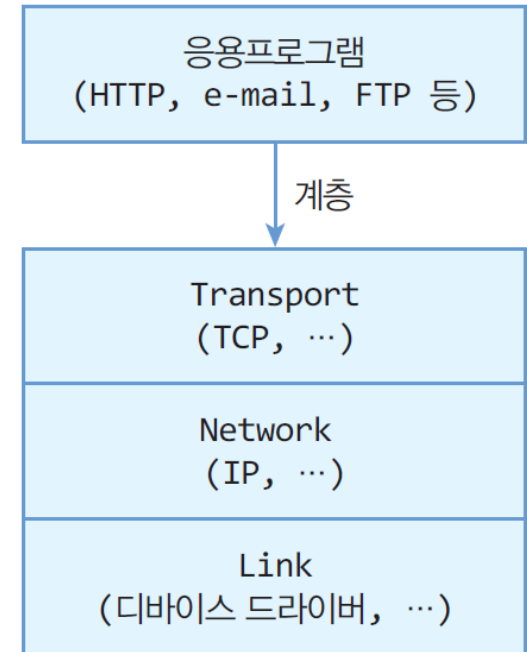
3

□ TCP/IP 프로토콜

- ▣ 두 시스템 간에 데이터가 손상없이 안전하게 전송되도록 하는 통신 프로토콜
- ▣ TCP에서 동작하는 응용프로그램 사례
 - e-mail, FTP, 웹(HTTP) 등

□ TCP/IP 특징

- ▣ 연결형 통신
 - 한 번 연결 후 계속 데이터 전송 가능
- ▣ 보낸 순서대로 받아 응용프로그램에게 전달



IP 주소

4

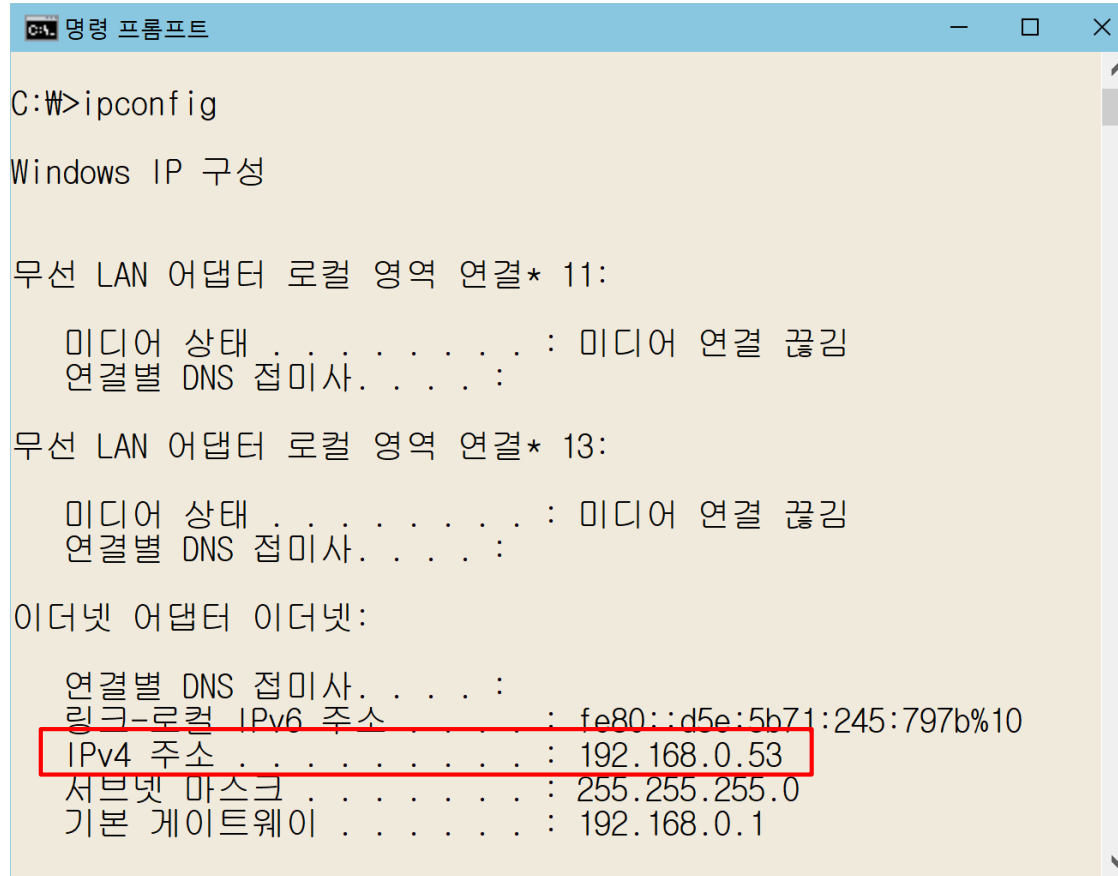
□ IP 주소

- 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
- 자신의 IP 주소를 간단히 localhost라는 이름으로 사용 가능

내 컴퓨터의 IP 주소 확인하기

5

- 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:

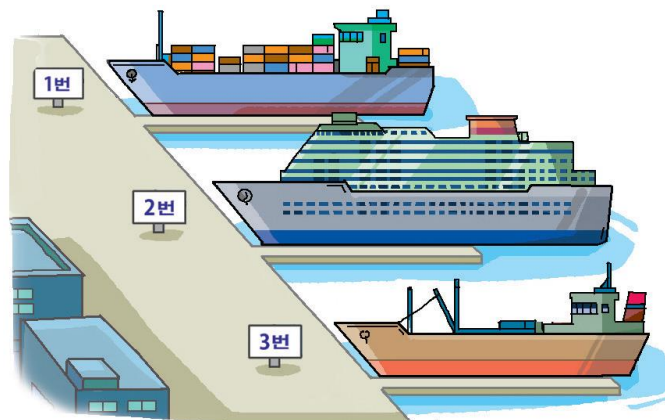
    미디어 상태 . . . . . : 미디어 연결 끊김
    연결별 DNS 접미사 . . . . :

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
```

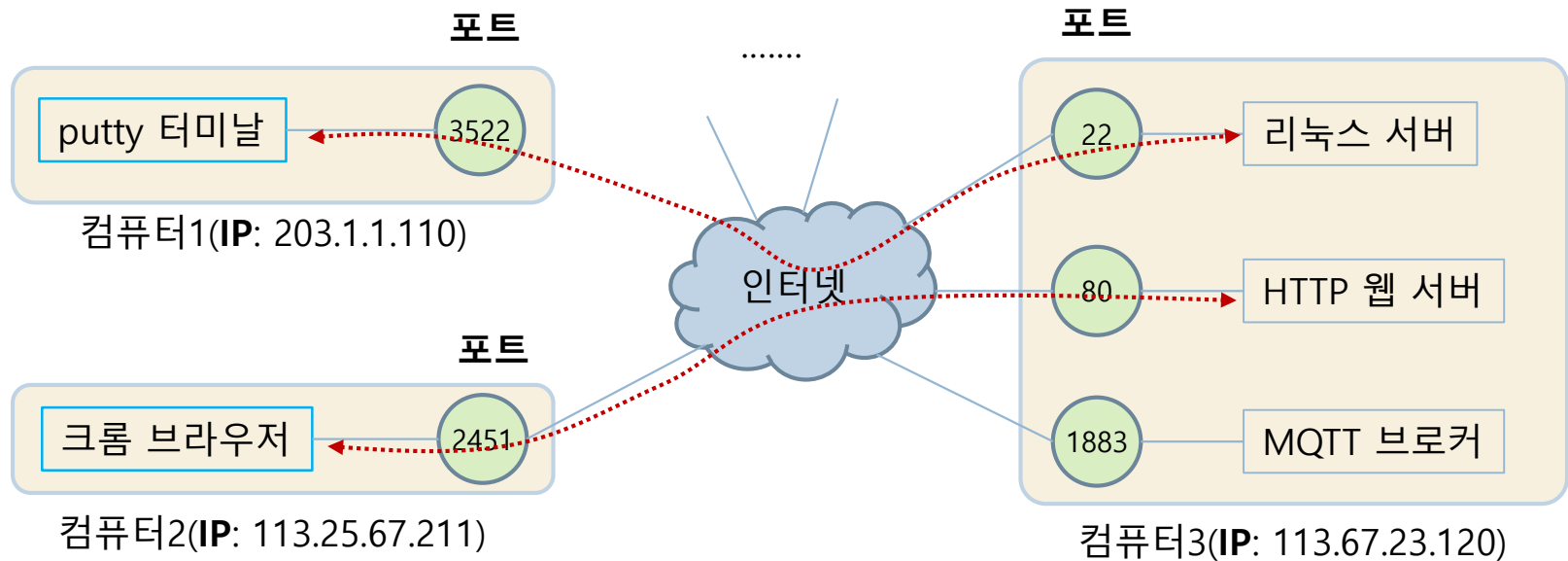
□ 포트

- 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 22, HTTP 80, FTP 21
 - 잘 알려진 포트 번호는 개발자가 사용하지 않는 것이 좋음
 - 충돌 가능성 있음



포트를 이용한 통신

7

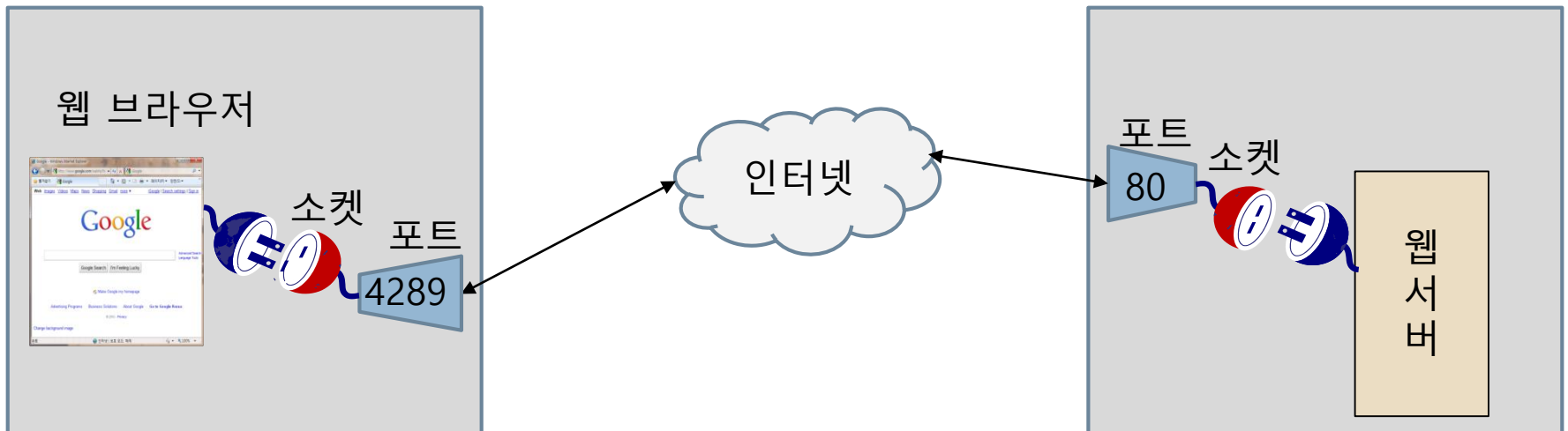


소켓 프로그래밍

8

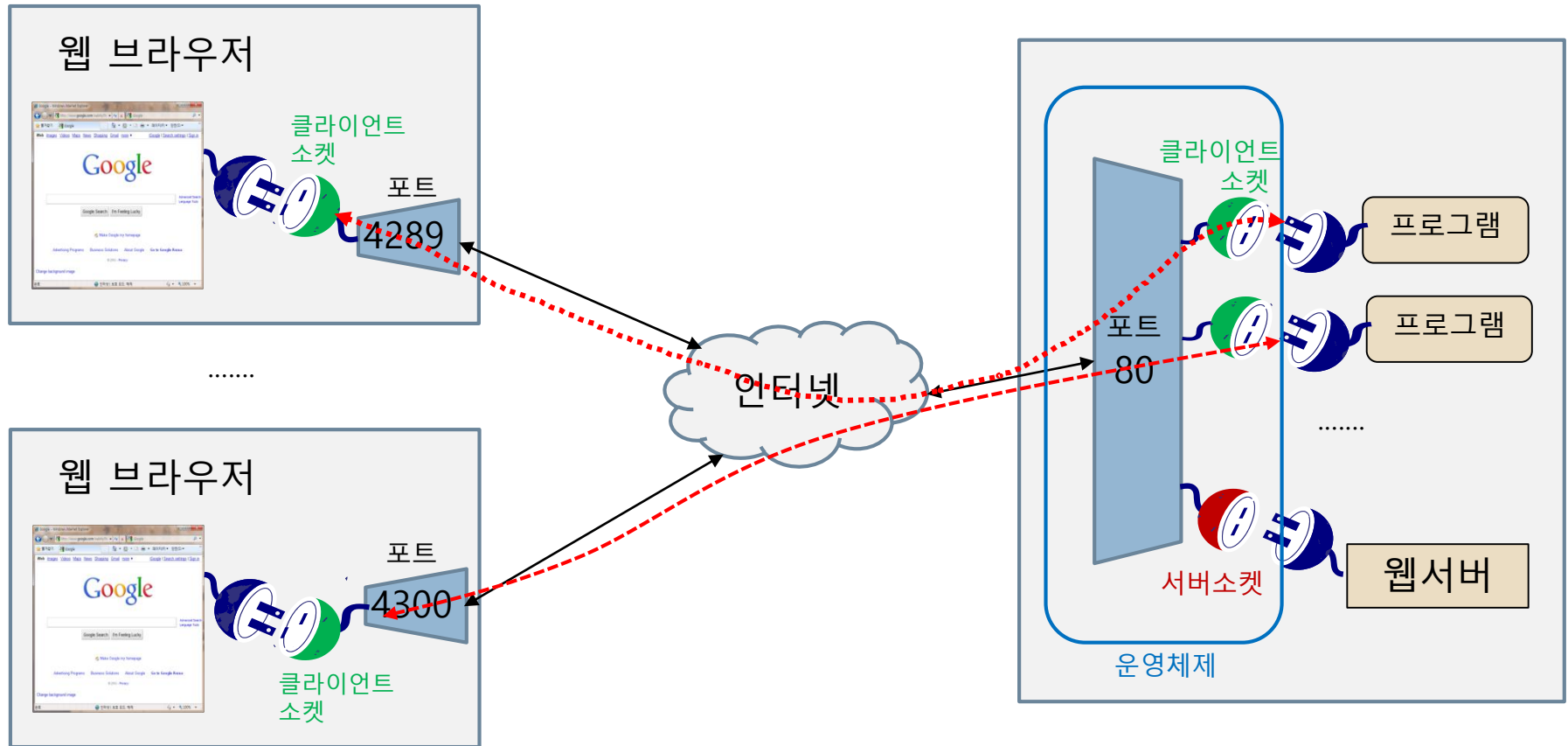
□ 소켓 (socket)

- 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



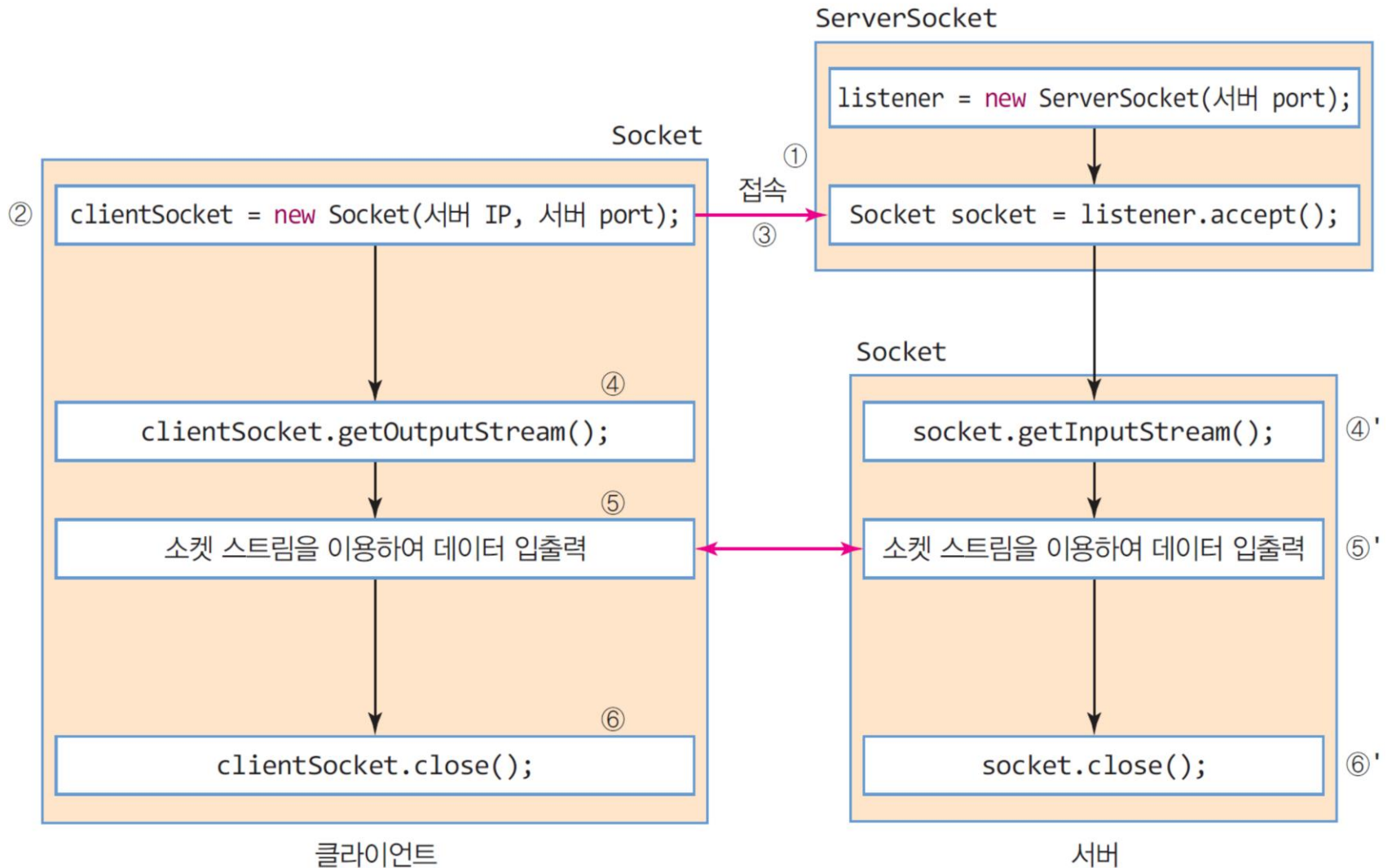
소켓을 이용한 웹 서버와 클라이언트 사이의 통신 사례

9



소켓을 이용한 서버 클라이언트 통신 프로그램의 전형적인 구조

10



Socket 클래스, 클라이언트 소켓

11

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

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

Socket 클래스의 메소드

12

메소드	설명
<code>bind(SocketAddress bindpoint)</code>	소켓에 로컬 IP 주소와 로컬 포트 지정
<code>void close()</code>	소켓을 닫는다.
<code>void connect(SocketAddress endpoint)</code>	서버에 연결
<code>InetAddress getAddress()</code>	연결된 서버 IP 주소 반환
<code>InputStream getInputStream()</code>	소켓의 입력 스트림 반환. 이 스트림을 이용하여 소켓이 상대방으로부터 받은 데이터를 읽을 수 있음
<code>InetAddress getLocalAddress()</code>	소켓의 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>	소켓의 출력 스트림 반환. 이 스트림에 출력하면 소켓이 상대방으로 데이터 전송
<code>boolean isBound()</code>	소켓이 로컬 주소에 결합되어 있으면 true 반환
<code>boolean isConnected()</code>	소켓이 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

13

- 클라이언트 소켓 생성 및 서버에 접속

```
Socket clientSocket = new Socket("128.12.1.1", 9999);
```

- Socket의 생성자에서 128.12.1.1의 주소의 9999포트에 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 데이터 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

14

□ ServerSocket 클래스

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

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

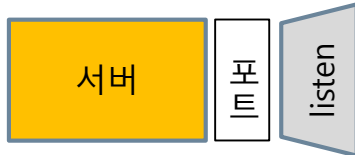
▣ 메소드

메소드	설명
Socket accept()	클라이언트로부터 연결 요청을 기다리다 요청이 들어오면 수락하고 클라이언트와 데이터를 주고받을 새 Socket 객체 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 IP 주소와 결합되어 있으면 true 반환
boolean isClosed()	서버 소켓이 닫혀있으면 true 반환
void setSoTimeout(int timeout)	accept()가 대기하는 타임아웃 시간 지정. 0이면 무한정 대기

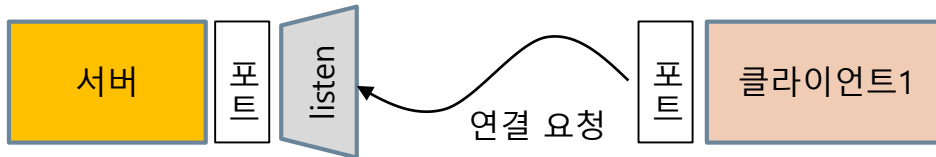
서버에 클라이언트가 연결되는 과정

15

- 서버는 서버 소켓으로 들어오는 연결 요청을 기다림(listen)

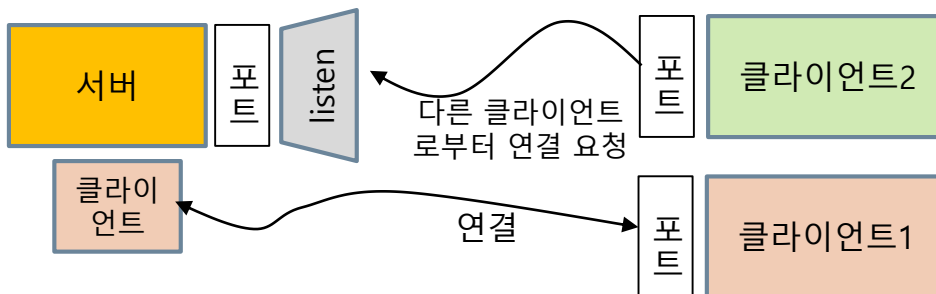


- 클라이언트가 서버에게 연결 요청



- 서버가 연결 요청 수락(accept)

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



서버가 클라이언트와 통신하는 과정

16

▣ 서버 소켓 생성

```
ServerSocket serverSocket = new ServerSocket(9999);
```

- 서버는 9999 포트에서 접속 기다리는 포트 9999 선택

▣ 클라이언트로부터 접속 기다림

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

- accept() 메소드는 접속 요청이 오면 접속 후 새 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()));
```

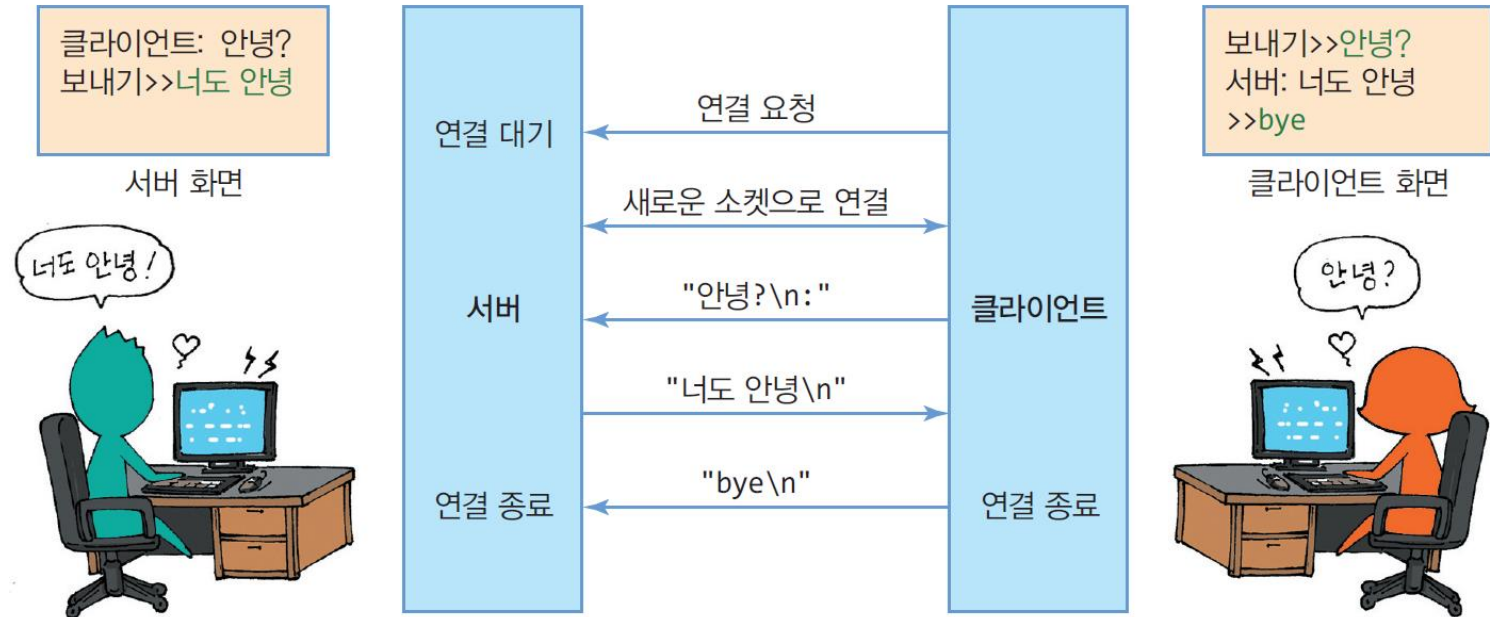
- Socket 객체의 getInputStream()과 getOutputStream() 메소드를 이용하여 입출력 데이터 스트림 생성

서버-클라이언트 채팅 프로그램 만들기

17

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

채팅 동작 과정

연결 대기

```
C:\W>java ServerEx
연결을 기다리고 있습니다.....
```

연결

```
C:\W>java ClientEx
보내기>>
```

```
C:\W>java ServerEx
연결을 기다리고 있습니다.....
연결되었습니다.
```

사용자 입력

"안녕\n" 전송

```
C:\W>java ClientEx
보내기>>안녕?
```

```
C:\W>java ServerEx
연결을 기다리고 있습니다.....
연결되었습니다.
클라이언트: 안녕?
보내기>>
```

"너도 안녕\n" 전송

사용자 입력

```
C:\W>java ClientEx
보내기>>안녕?
서버: 너도 안녕?
보내기>>
```

```
C:\W>java ServerEx
연결을 기다리고 있습니다.....
연결되었습니다.
클라이언트: 안녕?
보내기>>너도 안녕?
```

사용자 입력

프로그램 종료

"bye\n" 전송

```
C:\W>java ClientEx
보내기>>안녕?
서버: 너도 안녕?
보내기>>bye
C:\W>
```

```
C:\W>java ServerEx
연결되었습니다.
클라이언트: 안녕?
보내기>>너도 안녕?
클라이언트에서 bye로 연결을 종료하였음
C:\W>
```

프로그램 종료

수식 계산 서버-클라이언트 만들기 실습

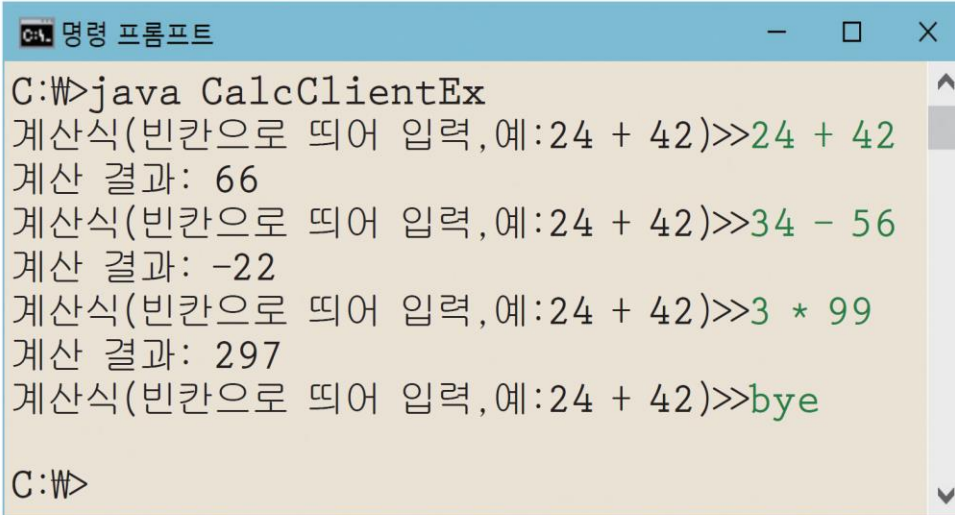
21

□ 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

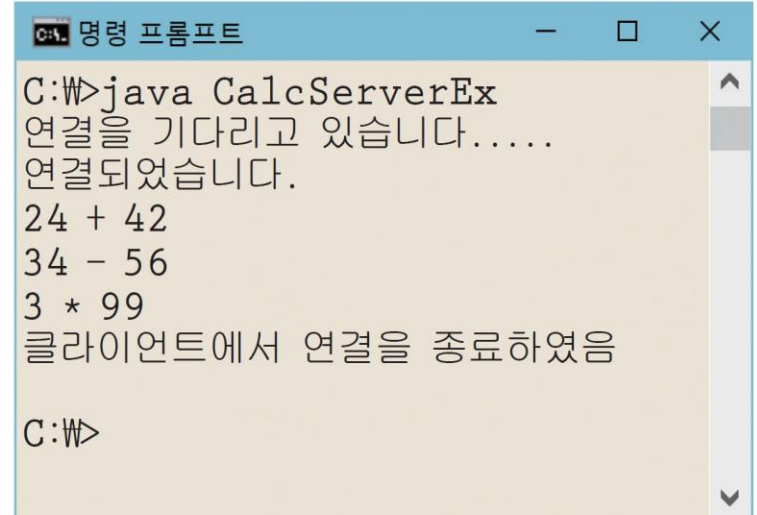
실행 예시

22



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

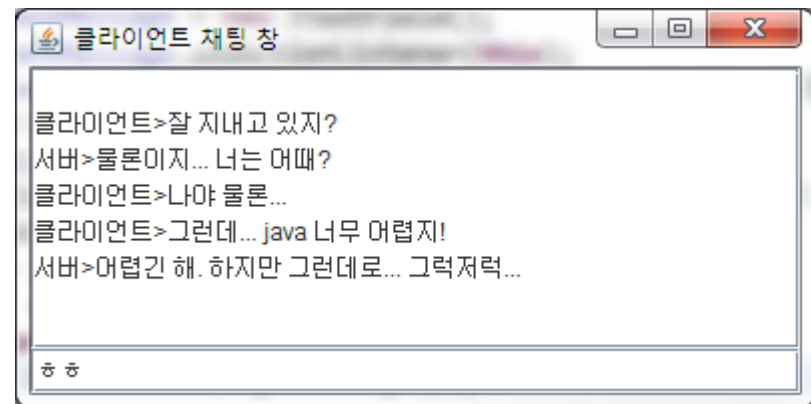
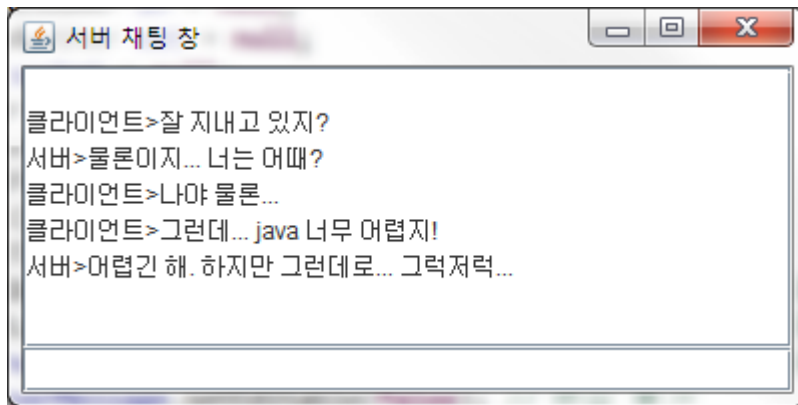
클라이언트 프로그램 CalcClientEx.java

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


TCP 예제: 채팅 서버와 클라이언트

25



소켓을 이용한 클라이언트 제작 순서

26

1. Socket 객체 생성
 - ▣ Socket client = **new** Socket(hostName, portNumber);
2. 소켓으로부터 스트림 객체를 얻는다.
 - ▣ InputStream input = client.getInputStream();
 - ▣ OutputStream output = client.getOutputStream();
3. 상호 대화 단계
 - ▣ read()와 write() 사용
4. 종료
 - ▣ close() 사용

소켓을 이용한 서버 제작 순서

27

1. ServerSocket 객체 생성
 - ▣ `ServerSocket server = new ServerSocket(portNumber, queueLength);`
2. `accept()` 메소드 호출
 - ▣ `Socket clientSocket = server.accept();`
3. 소켓으로부터 스트림 객체를 얻는다.
 - ▣ `InputStream input = clientSocket.getInputStream();`
 - ▣ `OutputStream output = clientSocket.getOutputStream();`
4. 상호 대화 단계
 - ▣ `read()`와 `write()` 사용
5. 종료
 - ▣ `close()` 사용

채팅 Server 전체구조

28

```
public class ChatServer extends JFrame implements ActionListener
{
    BufferedReader in = null;    // 클라이언트로 부터의 입력 스트림
    BufferedWriter out = null;   // 클라이언트로의 출력 스트림
    ServerSocket server = null;  // 서버소켓
    Socket socket = null;        // 소켓
    MsgViewer msgView;           // 메시지를 읽고, 보여주기 위한 컴포넌트
    JTextField msgInput;         // 메시지 입력을 위한 컴포넌트

    public ChatServer()
    {
        // 클라이언트에서 메시지 수신을 위한 스레드 생성
        // 서버 소켓 생성
        // 클라이언트로부터 연결 요청 대기
        // 클라이언트로부터의 입력 스트림 생성
        // 클라이언트로의 출력 스트림 생성
        // 클라이언트에서 메시지 수신을 위한 스레드 실행
    }

    private class MsgViewer extends JTextArea implements Runnable
    {
        public void run() {
            while (true) {
                // 클라이언트에서 한 행의 문자열 읽고, msgView에 문자열 붙여넣기
            }
        }
    }

    public void actionPerformed(ActionEvent e)
    {
        // 텍스트 필드에서 문자열 얻어와서 클라이언트로 문자열 전송
    }

    public static void main(String[] args) { new ChatClient(); }
}
```

채팅 Server 제작-1

29

```
public class ChatServer extends JFrame implements ActionListener
```

```
{
```

```
    BufferedReader in = null;  
    BufferedWriter out = null;  
    ServerSocket server = null;  
    Socket socket = null;
```

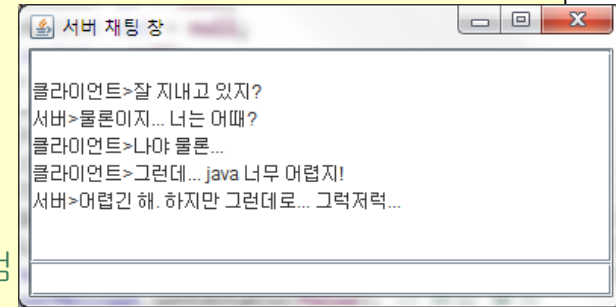
```
    MsgViewer msgView; // 메시지를 읽고, 보여주기 위한 컴  
    JTextField msgInput;
```

```
    public ChatServer()  
    {
```

```
        this.setTitle("서버 채팅 창"); // 프레임 타이틀  
        this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);  
        this.setLayout(new BorderLayout());
```

```
        // 클라이언트에서 받은 메시지를 읽고 출력할 컴포넌트  
        msgView = new MsgViewer();  
        msgView.setEditable(false); // 편집 불가  
        JScrollPane sp = new JScrollPane(msgView);  
        // 클라이언트에서 메시지 수신을 위한 스레드 생성  
        Thread th = new Thread(msgView);
```

```
        msgInput = new JTextField();  
        msgInput.addActionListener(this);
```

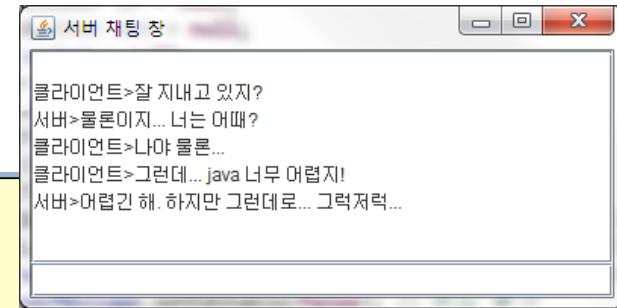


채팅 Server 제작-2

30

```
this.add(sp, BorderLayout.CENTER);
this.add(msgInput, BorderLayout.SOUTH);
this.setSize(400, 200);
this.setVisible(true); // 프레임이 화면에 나타나도록 설정
msgInput.requestFocus();

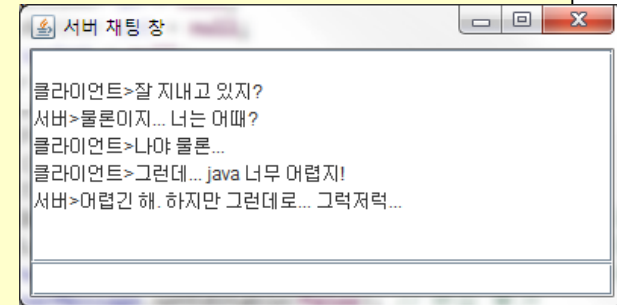
try
{
    server = new ServerSocket(9999); // 서버 소켓 생성
    socket = server.accept(); // 클라이언트로부터 연결 요청 대기
    System.out.println("연결됨");
    in = new BufferedReader(new InputStreamReader
        (socket.getInputStream())); // 클라이언트로부터의 입력 스트림
    out = new BufferedWriter(new OutputStreamWriter
        (socket.getOutputStream())); // 클라이언트로의 출력 스트림
}
catch (IOException e)
{
    System.out.println(e.getMessage());
    System.exit(1);
}
th.start();
}
```



채팅 Server 제작-3

31

```
private class MsgViewer extends JTextArea implements Runnable
{
    public void run()
    {
        String msg=null;
        while (true)
        {
            try
            {
                msg = in.readLine(); // 클라이언트에서 한 행의 문자열 읽음
            }
            catch (IOException e)
            {
                System.out.println(e.getMessage());
                System.exit(1);
            }
            msgView.append("\n" + msg);
            int pos = msgView.getText().length();
            msgView.setCaretPosition(pos); // caret 포지션을 끝으로 이동
        }
    }
}
```

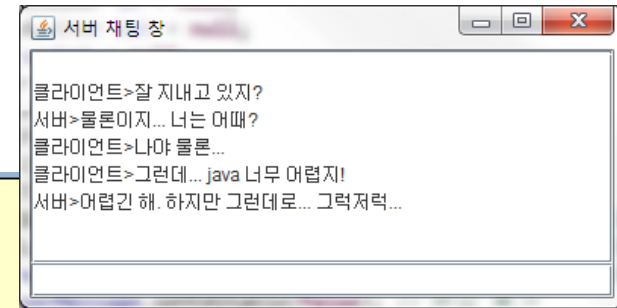


채팅 Server 제작-4

32

```
public void actionPerformed(ActionEvent e)
{
    if (e.getSource() == msgInput)
    {
        String msg = msgInput.getText(); // 텍스트 필드에서 문자열 얻어옴
        try
        {
            out.write("서버>" + msg+"\n"); // 클라이언트로 문자열 전송
            out.flush();
            msgView.append("\n서버>" + msg);
            int pos = msgView.getText().length();
            msgView.setCaretPosition(pos); // caret 위치를 끝으로 이동
            msgInput.setText(null); // 입력창의 문자열 지움
        }
        catch (IOException e1)
        {
            System.out.println(e1.getMessage());
            System.exit(1);
        }
    }
}

public static void main(String[] args)
{
    new ChatServer();
}
```



채팅 Client 전체구조

33

```
public class ChatClient extends JFrame implements ActionListener
{
    BufferedReader in = null;    // 서버로부터의 입력 스트림
    BufferedWriter out = null;   // 서버로의 출력 스트림
    Socket socket = null;        // 소켓
    MsgViewer msgView;           // 메시지를 읽고, 보여주기 위한 컴포넌트
    JTextField msgInput;         // 메시지 입력을 위한 컴포넌트

    public ChatClient()
    {
        // 서버에서 메시지 수신을 위한 스레드 생성
        // 클라이언트 소켓 생성
        // 서버로부터의 입력 스트림 생성
        // 서버로의 출력 스트림 생성
        // 서버에서 메시지 수신을 위한 스레드 시작
    }

    private class MsgViewer extends JTextArea implements Runnable
    {
        public void run() {
            while (true) {
                // 서버에서 한 행의 문자열 읽음
                // msgView에 문자열 붙여넣기
            }
        }
    }

    public void actionPerformed(ActionEvent e)
    {
        // 텍스트 필드에서 문자열 얻어와서 서버로 문자열 전송
    }

    public static void main(String[] args)
    {
        new ChatClient();
    }
}
```

채팅 Client 제작-1

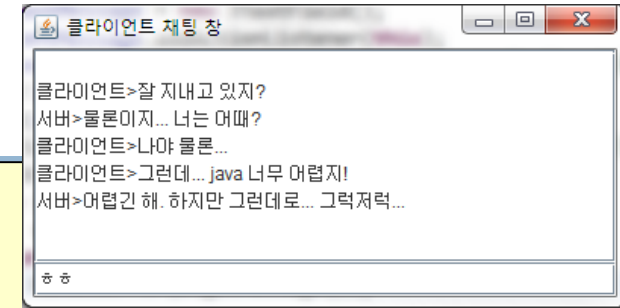
34

```
import java.awt.*;
import java.awt.event.*;
import java.io.*;
import java.net.*;
import javax.swing.*;

public class ChatClient extends JFrame implements ActionListener
{
    BufferedReader in = null;
    BufferedWriter out = null;
    Socket socket = null;

    MsgViewer msgView; // 메시지를 읽고, 보여주기 위한 컴포넌트
    JTextField msgInput;

    public ChatClient()
    {
        this.setTitle("클라이언트 채팅 창"); // 프레임 타이틀
        this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        this.setLayout(new BorderLayout()); //BorderLayout 배치관리자의 사용
        msgView = new MsgViewer(); // 서버에서 받은 메시지를 읽고 출력할 컴포넌트
        msgView.setEditable(false); // 편집 불가
        JScrollPane sp = new JScrollPane(msgView);
        Thread th = new Thread(msgView); // 서버에서 메시지 수신을 위한 스레드
```



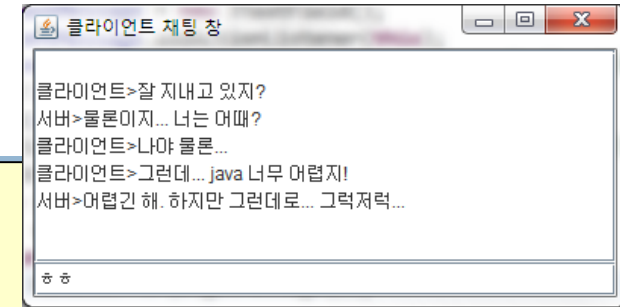
채팅 Client 제작-2

35

```
msgInput = new JTextField();
msgInput.addActionListener(this);

this.add(sp, BorderLayout.CENTER);
this.add(msgInput, BorderLayout.SOUTH);
this.setSize(400, 200);
this.setVisible(true); // 프레임이 화면에 나타나도록 설정
msgInput.requestFocus();

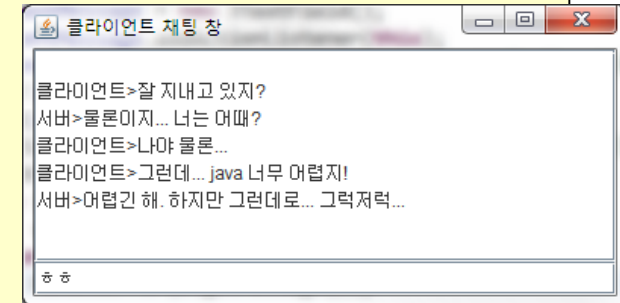
try
{
    socket = new Socket("localhost", 9999); // 클라이언트 소켓 생성
    System.out.println("연결됨");
    in = new BufferedReader(new InputStreamReader
        (socket.getInputStream())); // 서버로부터의 입력 스트림
    out = new BufferedWriter(new OutputStreamWriter
        (socket.getOutputStream())); // 서버로의 출력 스트림
}
catch (IOException e)
{
    System.out.println(e.getMessage());
    System.exit(1);
}
th.start();
}
```



채팅 Client 제작-3

36

```
private class MsgViewer extends JTextArea implements Runnable
{
    public void run()
    {
        String msg=null;
        while (true)
        {
            try
            {
                msg = in.readLine(); // 서버에서 한 행의 문자열 읽음
            }
            catch (IOException e)
            {
                System.out.println(e.getMessage());
                System.exit(1);
            }
            msgView.append("\n" + msg);
            int pos = msgView.getText().length();
            msgView.setCaretPosition(pos); // caret 포지션을 끝으로 이동
        }
    }
}
```

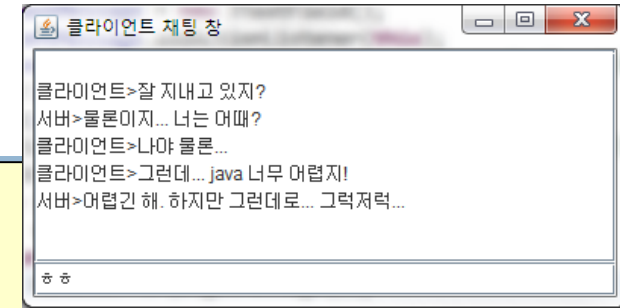


채팅 Client 제작-4

37

```
public void actionPerformed(ActionEvent e)
{
    if (e.getSource() == msgInput)
    {
        String msg = msgInput.getText(); // 텍스트 필드에서 문자열 얻어옴
        try
        {
            out.write("클라이언트>" + msg+"\n"); // 서버로 문자열 전송
            out.flush();
            msgView.append("\n클라이언트>" + msg);
            int pos = msgView.getText().length();
            msgView.setCaretPosition(pos); // caret 포지션을 끝으로 이동
            msgInput.setText(null); // 입력창의 문자열 지움
        }
        catch (IOException e1)
        {
            System.out.println(e1.getMessage());
            System.exit(1);
        }
    }
}

public static void main(String[] args)
{
    new ChatClient();
}
```



소켓을 이용한 다중채팅의 구현

38

□ 서버

- ▣ 클라이언트가 서버에 접속할 때 마다, 클라이언트를 각각 스레드로 만들어 ArrayList 또는 Vector에 저장
- ▣ 스레드(클라이언트에 대한 입출력 수행)
 - 입출력 스트림 생성
 - 입출력 수행- read, write
- ❖ 서버는 단순히, 클라이언트들을 중계하는 역할만 수행

□ 클라이언트

- ▣ 1:1 채팅과 동일

❖ <http://javaking75.blog.me/140188484493>

참고: Reflection

39

□ 리플렉션이란?

- ▣ 객체를 통해 클래스의 정보를 분석해 내는 프로그램 기법
- ▣ 사전적 의미 : 투영, 반사

□ 필요성

- ▣ 클래스의 타입을 모르면 메소드의 실행이 어렵다?
- ▣ (예)
 - `Object car = new Car(); // up-casting`
`car.drive(); // error`
 - 왜? 부모클래스로 up-casting된 객체는 부모클래스의 멤버들만 접근할 수 있다.
- ▣ 클래스의 구체적인 타입을 몰라도 접근할 수 있도록 도와주는 Java API가 필요

□ 리플렉션의 용도

- ▣ 실행시간에 다른 클래스 객체를 동적으로 로딩하여 접근(예: Spring 프레임워크 어노테이션 등)
 - 필드, 메소드, 클래스의 패키지 정보, 접근지정자, 부모클래스, 어노테이션(annotation) 등

```
Class v = Class.forName("java.util.Vector");  
Method[] methods = v.getDeclaredMethods();  
for (Method method : methods)  
    System.out.println(method.toString());
```

□ 주의사항

- ▣ Field.setAccessible(true)를 통해 private 멤버도 접근과 조작이 가능

Class 객체 생성 방법

41

- Reflection을 위해서는 `java.lang.Class` 객체를 생성한 이후에, 필드, 메소드 등에 접근
 - ▣ `import java.lang.reflect.*;`
 - `Field`, `Method`, `Constructor` 클래스를 사용하기 위해...
- Class 객체 생성 방법
 - ▣ `Class obj = "instance_name".getClass();`
 - instance가 있는 경우에 사용
 - ▣ `Class obj = "class_name".class;`
 - instance가 없는 경우에도 가능
 - ▣ `Class obj = Class.forName("class_name");`
 - instance가 없는 경우에도 가능

메소드 접근 관련

42

□ Class 클래스

- ▣ `public Method getDeclaredMethod(String name)`
- ▣ `public Method getMethod(String name)`
- ▣ `public Method[] getDeclaredMethods()`
- ▣ `public Method[] getMethods()`

□ Method 클래스

- ▣ `public String getName()`
- ▣ `public Class<?>[] getParameterTypes()`
- ▣ `public Class<?> getReturnType()`
- ▣ `public Annotation[] getDeclaredAnnotations()`
- ▣ `public Object invoke(Object obj, ...)`

예제

43

```
import java.lang.reflect.*;

class Example
{
    private void print(int p, int q) { System.out.println(p+q); }
}

public class Test
{
    public static void main(String[] args) throws ClassNotFoundException
    {
        Class a = Class.forName("Example");
        Method[] methods = a.getDeclaredMethods();
        for(Method method : methods)
        {
            System.out.println(method.getName());
            Class[] params = method.getParameterTypes();
            for(int i=0; i<params.length; i++)
                System.out.println(params[i]);
            System.out.println(method.getReturnType());
        }
    }
}
```

```
import java.lang.reflect.*;

class Example
{
    public Example(int x) { }
    public void print(int p, int q) { System.out.println(p+q); }
}

public class Test
{
    public static void main(String[] args)
    {
        try {
            Class a = Class.forName("Example");
            Class[] paramTypes = new Class[2];
            paramTypes[0] = Integer.TYPE;
            paramTypes[1] = Integer.TYPE;
            Method method = a.getMethod("print", paramTypes);

            Example obj = new Example(0);
            Object[] argList = new Object[2];
            argList[0] = new Integer(3);
            argList[1] = new Integer(5);
            Object rtn = method.invoke(obj, argList);
        }
        catch(Exception e) {}
    }
}
```

생성자 접근 관련

45

□ Class 클래스

- ▣ `public Constructor<?>[]`
`getDeclaredConstructors()`
- ▣ `public Constructor<?>[]` `getConstructors()`

□ Constructor 클래스

- ▣ `public String getName()`
- ▣ `public Class<?>[] getParameterTypes()`
- ▣ `public Annotation[] getDeclaredAnnotations()`
- ▣ `public T newInstance(Object... initargs)`

예제

46

```
import java.lang.reflect.*;

class Example
{
    public Example(int x) { }
}

public class Test
{
    public static void main(String[] args) throws ClassNotFoundException
    {
        Class a = Class.forName("Example");
        Constructor[] cons = a.getDeclaredConstructors();
        for(Constructor con : cons)
        {
            System.out.println(con.getName());
            Class[] params = con.getParameterTypes();
            for(int i=0; i<params.length; i++)
                System.out.println(params[i]);
        }
    }
}
```

```
import java.lang.reflect.*;

class Example
{
    public Example(int x, int y) {
        System.out.println("Constructor!!");
    }
}

public class Test
{
    public static void main(String[] args)
    {
        try {
            Class a = Class.forName("Example");
            Class[] paramTypes = new Class[2];
            paramTypes[0] = Integer.TYPE;
            paramTypes[1] = Integer.TYPE;
            Constructor cs = a.getConstructor(paramTypes);

            Object[] argList = new Object[2];
            argList[0] = new Integer(3);
            argList[1] = new Integer(5);
            Object rtn = cs.newInstance(argList);
        }
        catch(Exception e) {}
    }
}
```

필드 접근 관련

48

□ Class 클래스

- ▣ `public Field getDeclaredField(String name)`
- ▣ `public Field getField(String name)`
- ▣ `public Field[] getDeclaredFields()`
- ▣ `public Field[] getFields()`

□ Field 클래스

- ▣ `public String getName()`
- ▣ `public Class<?> getType()`
- ▣ `public Annotation[] getDeclaredAnnotations()`
- ▣ `public Object get(Object obj)`
- ▣ `public void setInt(Object obj, int i)`
- ▣ `public void setDouble(Object obj, double d)`


```
import java.lang.reflect.*;

class Example
{
    private double data = 3.14;
    public Example(int x) { }
    private void print(int p, int q) { System.out.println(p+q); }
}

public class Test
{
    public static void main(String[] args) throws ClassNotFoundException,
    IllegalArgumentException, IllegalAccessException
    {
        Class a = Class.forName("Example");
        Field[] fields = a.getDeclaredFields();
        for(Field field : fields)
        {
            System.out.println(field.getName());
            System.out.println(field.getType());
            field.setAccessible(true);
            Example obj = new Example(0);
            System.out.println(field.get(obj));
            field.setDouble(obj, 2.75);
            System.out.println(field.get(obj));
        }
    }
}
```