

Chapter 08.TCP 프로그래밍

Originally made by Prof. Hanku Lee

Modified by Mingyu Lim

**Collaborative Computing Systems Lab.
School of Internet & Multimedia Engineering
Konkuk University, Seoul, Korea**

0. skip 내용 확인

□ 7장 네트워크 프로그래밍 기초

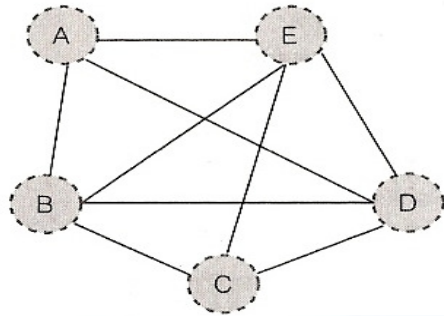
1. 소켓이란 무엇인가 ?
2. 소켓의 일반적인 세가지 형식 ?
3. IP와 PORT 의 개념은 무엇인가 ?
4. InetAddress 클래스의 메소드에 대해서 설명하시요 ?
5. InetAddress 클래스를 이용한 nslookup 명령구현 순서에 대해서 설명하시요 ?

CHROMATIC COLOR

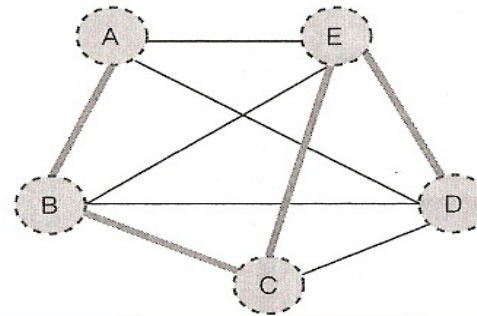
ACHROMATIC COLOR

1. TCP 프로그래밍 기본

□ TCP 기본 개요



[그림 8-1] 연결되기 전 각 노드와 네트워크 상태

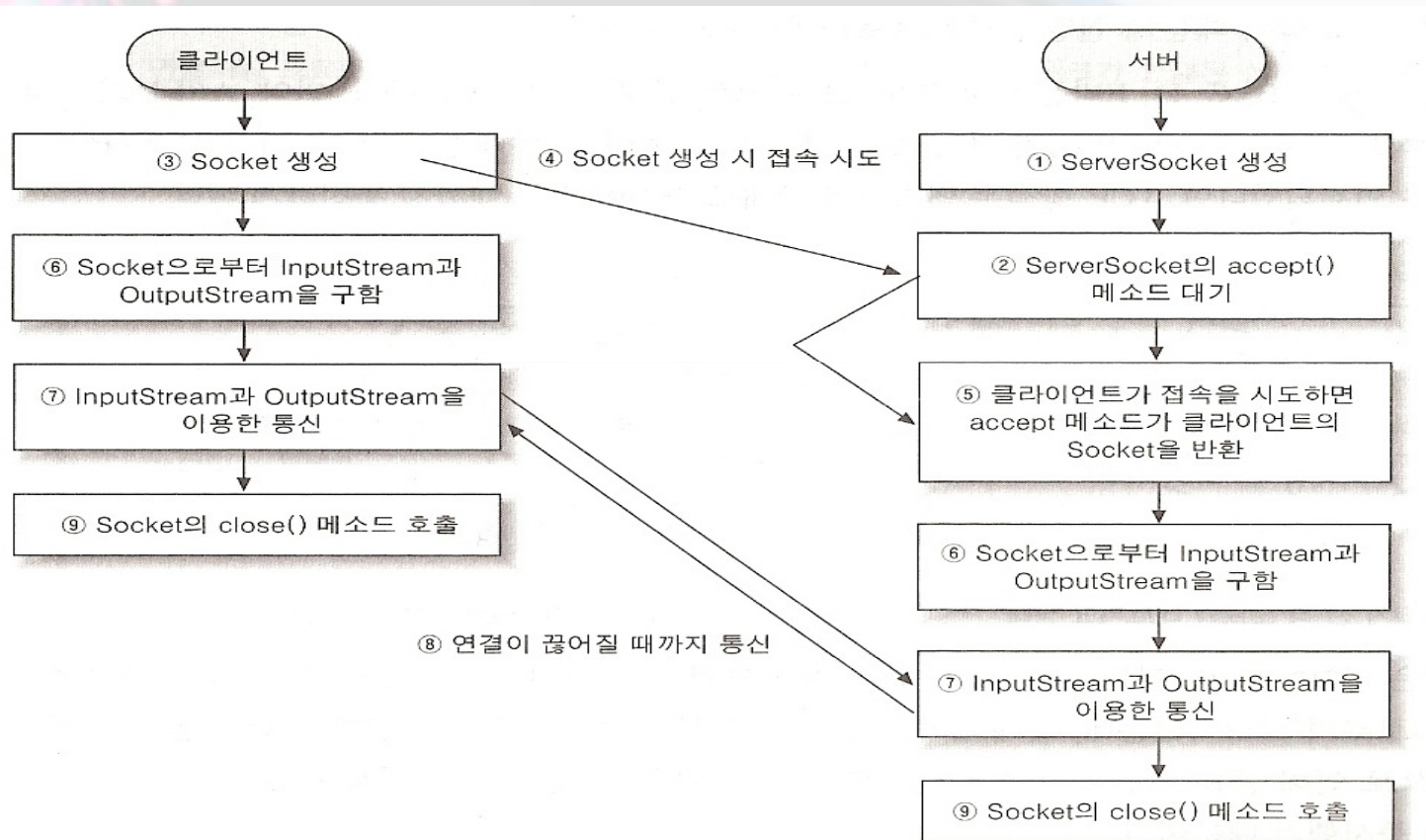


[그림 8-2] A와 D가 통신하기 위해서 연결된 모습

- 소켓프로그래밍 : TCP와 UDP 두 가지 방법 제공 (자바 제공)
- 스트림 통신 프로토콜 (연결지향 프로토콜)
- 신뢰성 있는 데이터 전송 프로그램
- 연결지향 (연결이 끊어질때까지 송신한 데이터가 차례대로 목적지 소켓에 전달되는 신뢰성 있는 통신 가능)

1. TCP 프로그래밍 기본(계속)

□ TCP 기본 개요



1. TCP 프로그래밍 기본(계속)

□ 클라이언트의 접속 대기

```
ServerSocket server = new ServerSocket (10001);  
System.out.println("접속을 기다립니다.");  
Socket sock = server.accept();
```

- ServerSocket 객체를 생성한후 ServerSocket에 있는 accept() 메소드를 실행해서 대기
- 대기 하기 위해서 멈춰 있는 메소드를 블록킹 메소드

ACHROMATIC COLOR

1. TCP 프로그래밍 기본(계속)

□ 클라이언트 접속

```
Socket sock = new Socket ("127.0.0.1",10001);
```

- 서버에서 accept() 로 대기 하고 있다면 클라이언트는 서버로 접속 가능
- Socket 만 생성하면 내부적으로 알아서 접속
- 클라이언트에서 Socket 객체가 성공적으로 생성 되면 서버의 accept() 메소드는 클라이언트에 대한 Socket 객체를 반환
- 클라이언트에서 위문장이 실행되면 accept() 대기하고 있다가 다음 방법으로 접속한 클라이언트의 Socket 객체를 반환

```
Socket sock = server.accept();
```

1. TCP 프로그래밍 기본(계속)

□ Socket 으로부터 InputStream과 OutputStream 구하기

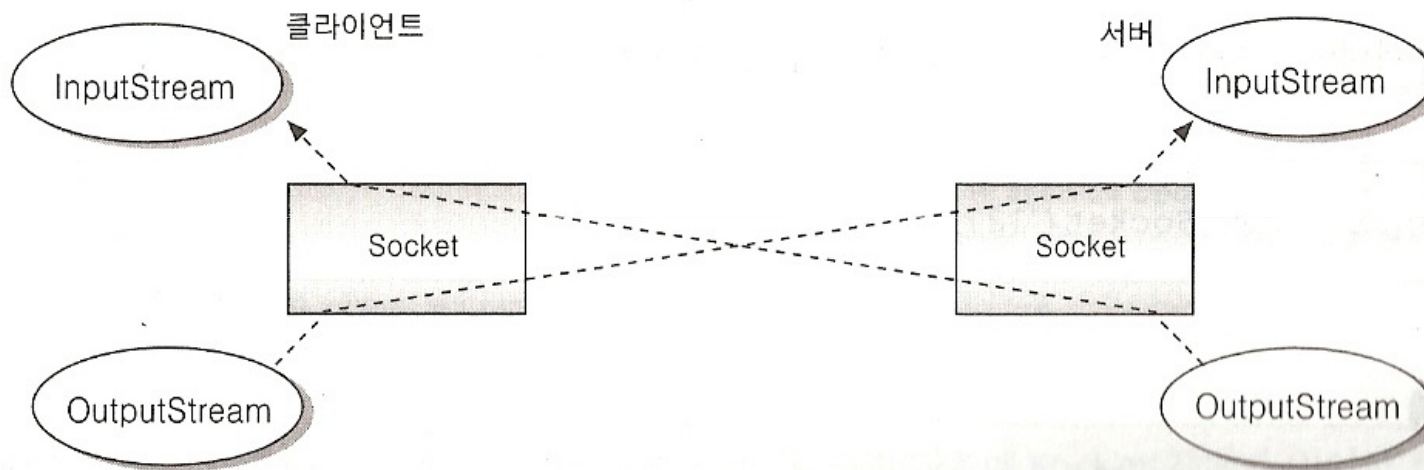
```
OutputStream out = sock.getOutputStream();  
InputStream in = sock.getInputStream();
```

- 소켓이 있다는 것은 소켓이 연결되어 있는 곳에 읽고 쓸수있는 InputStream과 OutputStream을 구할수 있다는 것을 의미

클라이언트에서 OutputStream 을 썼다면, 서버에서는 InputStream 이용해서 읽어 들여야 한다. 읽기만 할경우 서버와 클라이언트가 멈출수 있
다. 그렇게 때문에 TCP 방식의 프로그래밍은 프로토콜에 맞춰 조심스럽
게 작성하여야 한다.

1. TCP 프로그래밍 기본(계속)

□ Socket 으로부터 InputStream과 OutputStream 구하기



[그림 8-4] Socket의 InputStream과 OutputStream을 이용한 통신

1. TCP 프로그래밍 기본(계속)

□ Socket 으로부터 InputStream과 OutputStream 구하기

```
PrintWriter pw = new PrintWriter(new OutputStreamWriter(out));  
BufferedReader br = new BufferedReader(new InputStreamReader(in));
```

- 소켓에서 구한 InputStream 과 OutputStream 을 BufferedReader 와 PrintWriter 형태로 변환

□ 접속 끊기

```
sock.close()
```

- 소켓을 통해서 얻은 IO 객체가 있을 경우에는 반드시 close() 메소드 이용해서 종료

2. 간단한 에코 클라이언트 / 서버 프로그래밍

□ 에코 서버

1. 1001번 포트에서 동작하는 ServerSocket 생성
2. ServerSocket의 accept() 메소드를 실행해서 클라이언트의 접속을 대기
3. 클라이언트가 접속할 경우 accept() 메소드는 Socket 객체 반환
4. 반환 받은 Socket 으로부터 InputStream, OutputStream 을 구한다.
5. InputStream은 BufferedReader 형식으로 변환 OutputStream 은 PrintWriter 형식으로 변환한다.
6. readLine() 메소드를 이용해서 클라이언트가 보내는 문자열 한줄을 읽음
7. 6에서 읽어 들인 문자열을 PrintWriter 에 있는 println() 메소드를 이용해서 다시 클라이언트로 전송
8. 6,7 의 작업은 클라이언트가 접속을 종료할때까지 반복
9. IO 객체와 소켓의 close() 메소드를 호출

2. 간단한 에코 클라이언트 / 서버 프로그래밍

□ 에코 클라이언트

1. Socket 생성자에 서버의 IP주소와 서버 동작 포트 값(10001)을 인자로 생성
2. 생성된 Socket 으로부터 InputStream과 OutputStream 을 구한다.
3. InputStream은 BufferedReader 형식으로 변환 OutputStream 은 PrintWriter 형식으로 변환한다.
4. 키보드로부터 한줄씩 입력 받는 BufferedReader 객체를 생성
5. 키보드로부터 한줄을 입력 받아 PrintWriter에 있는 println()를 이용해서 서버 전송
6. 서버가 다시 반환하는 문자열을 BufferedReader에 있는 readLine() 메소드를 이용해서 읽어 들이고 화면에 출력한다.
7. 4,5,6을 키보드로부터 quit 문자열을 입력받을 때까지 반복
8. 키보드로부터 quit 문자열이 입력되면 IO 객체와 소켓의 close() 메소드 호출

2. 간단한 에코 클라이언트 / 서버 프로그래밍

□ 에코 서버 프로그래밍(예제)

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

public class EchoServer {

    public static void main(String[] args) {
        try{
            ServerSocket server = new ServerSocket(10001);
            System.out.println("접속을 기다립니다.");
            Socket sock = server.accept();
            InetAddress inetaddr = sock.getInetAddress();
            System.out.println(inetaddr.getHostAddress() + " 로 부터 접속하였습니다.");
            OutputStream out = sock.getOutputStream();
            InputStream in = sock.getInputStream();
            PrintWriter pw = new PrintWriter(new OutputStreamWriter(out));
            BufferedReader br = new BufferedReader(new InputStreamReader(in));
            String line = null;
            while ((line = br.readLine()) != null){
                System.out.println("클라이언트로 부터 전송받은 문자열 : " + line);
                pw.println(line);
                pw.flush();
            }
            pw.close();
            br.close();
            sock.close();
        } catch (Exception e) {
            System.out.println(e);
        }
    } // main
}
```

n/w 프로그래밍의 경우 출력한 후에는 반드시 `flush()` 메소드 호출. 메소드를 호출하지 않을 경우에 실제로 전송이 안되는 경우도 발생된다.

flush()

출력 스트림을 통하여 쓰기를 할 때 일반적으로 버퍼에 가득차게 되면 한꺼번에 보내게 되는데, 이 메서드를 사용하게 되면 버퍼에 가득 차 있
지 않더라도 버퍼의 내용을 바로 보내게 된다.

2. 간단한 에코 클라이언트 / 서버 프로그래밍

□ 에코 클라이언트 프로그래밍(예제)

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

public class EchoClient{

    public static void main(String[] args) {
        try{
            Socket sock = new Socket("127.0.0.1", 10001);
            BufferedReader keyboard = new BufferedReader(new InputStreamReader(System.in));
            OutputStream out = sock.getOutputStream();
            InputStream in = sock.getInputStream();
            PrintWriter pw = new PrintWriter(new OutputStreamWriter(out));
            BufferedReader br = new BufferedReader(new InputStreamReader(in));
            String line = null;
            while((line = keyboard.readLine()) != null){
                if(line.equals("quit")) break;
                pw.println(line);
                pw.flush();
                String echo = br.readLine();
                System.out.println("서버로부터 전달받은 문자열 : " + echo);
            }
            pw.close();
            br.close();
            sock.close();
        } catch (Exception e) {
            System.out.println(e);
        }
    } // main
}
```

2. 간단한 에코 클라이언트 / 서버 프로그래밍

□ 에코 서버 프로그래밍(실행)

```
C:\>명령 프롬프트 - java EchoServer
C:\>C:\java\weclipse-SDK-3.3-win32\weclipse\workspace\자바 io nio\8장 TCP\bin>dir/w
C 드라이브의 볼륨에는 이름이 없습니다.
볼륨 일련 번호: 747A-FF42

C:\>C:\java\weclipse-SDK-3.3-win32\weclipse\workspace\자바 io nio\8장 TCP\bin 디렉터리

[.]
[.]
EchoClient.class  EchoServer.class
2개 파일        4,369 바이트
2개 디렉터리    84,340,396,032 바이트 남음

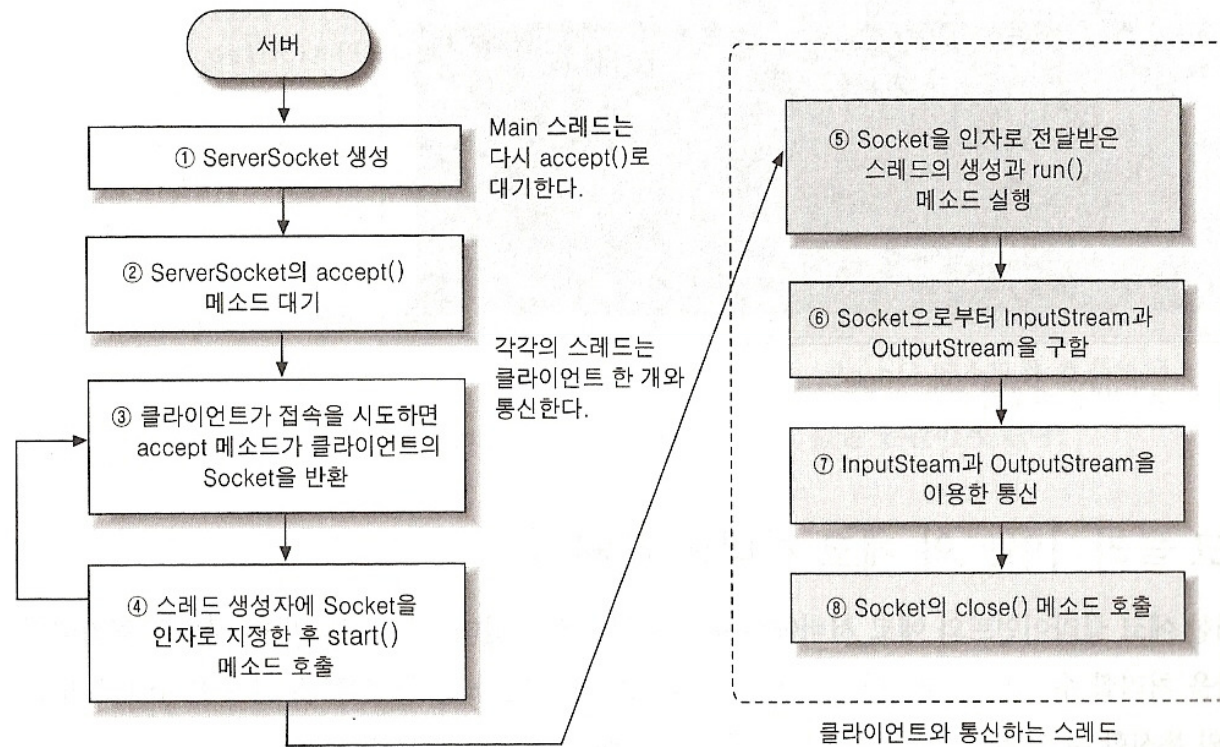
C:\>C:\java\weclipse-SDK-3.3-win32\weclipse\workspace\자바 io nio\8장 TCP\bin>javac *.
java
javac: file not found: *.java
Usage: javac <options> <source files>
use -help for a list of possible options

C:\>C:\java\weclipse-SDK-3.3-win32\weclipse\workspace\자바 io nio\8장 TCP\bin>java Ech
oServer
접속을 기다립니다.
127.0.0.1 로 부터 접속하였습니다.
클라이언트로부터 전송받은 문자열 : 안녕하세요
클라이언트로부터 전송받은 문자열 : 반갑습니다.
클라이언트로부터 전송받은 문자열 : n/w 프로그래밍 참재미 있네요
* 서버쪽 실행창

C:\>명령 프롬프트 - java EchoClient
C:\>C:\java\weclipse-SDK-3.3-win32\weclipse\workspace\자바 io nio\8장 TCP\bin>java Ech
oClient
안녕하세요
서버로부터 전달받은 문자열 :안녕하세요
반갑습니다.
서버로부터 전달받은 문자열 :반갑습니다.
n/w 프로그래밍 참재미 있네요
서버로부터 전달받은 문자열 :n/w 프로그래밍 참재미 있네요
* 클라이언트 실행창
```

3. 멀티스레드를 이용한 에코 서버

□ 멀티스레드를 이용한 에코 서버 동작 순서



[그림 8-7] 멀티스레드 에코 서버의 동작 순서

3. 멀티스레드를 이용한 에코 서버(계속)

□ 멀티스레드 에코 서버 프로그래밍(예제)

```
EchoServer.java  EchoClient.java  EchoThreadServer.java X
import java.net.*;
import java.io.*;

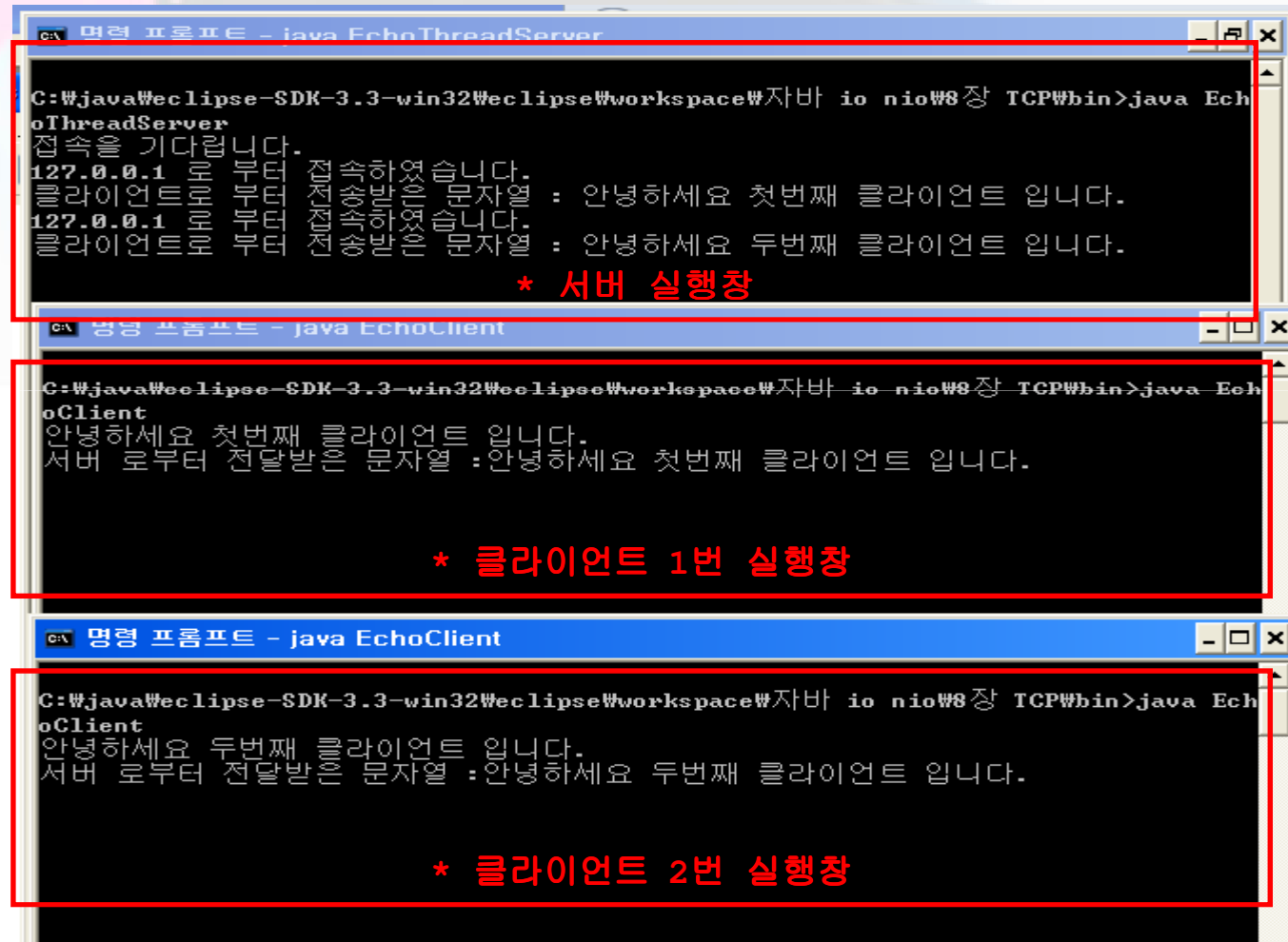
public class EchoThreadServer {

    public static void main(String[] args) {
        try{
            ServerSocket server = new ServerSocket(10001);
            System.out.println("접속을 기다립니다.");
            while(true){
                Socket sock = server.accept();
                EchoThread echothread = new EchoThread(sock);
                echothread.start();
            } // while
        } catch (Exception e) {
            System.out.println(e);
        }
    } // main
}

class EchoThread extends Thread{
    private Socket sock;
    public EchoThread(Socket sock){
        this.sock = sock;
    } // 생성자
    public void run(){
        try{
            InetAddress inetaddr = sock.getInetAddress();
            System.out.println(inetaddr.getHostAddress() + " 로 부터 접속하였습니다.");
            OutputStream out = sock.getOutputStream();
            InputStream in = sock.getInputStream();
            PrintWriter pw = new PrintWriter(new OutputStreamWriter(out));
            BufferedReader br = new BufferedReader(new InputStreamReader(in));
            String line = null;
            while((line = br.readLine()) != null){
                System.out.println("클라이언트로 부터 전송받은 문자열 : " + line);
                pw.println(line);
                pw.flush();
            }
            pw.close();
            br.close();
            sock.close();
        } catch (Exception ex) {
            System.out.println(ex);
        }
    } // run
}
```


3. 멀티스레드를 이용한 에코 서버(계속)

□ 멀티스레드 에코 서버 프로그래밍(실행결과)



The image displays three sequential command prompt windows from a Windows operating system, each with a title bar indicating the current directory is 'C:\java\weclipse-SDK-3.3-win32\weclipse\workspace\자바 io.nio.8장 TCP\bin'.

The first window, titled '명령 프롬프트 - java EchoThreadServer', shows the execution of 'java EchoThreadServer'. The output indicates the server is listening on port 127.0.0.1 and has received two connection requests from the same IP address, each with a greeting message.

The second window, titled '명령 프롬프트 - java EchoClient', shows the execution of 'java EchoClient'. The output shows the client sending a greeting and receiving a response from the server.

The third window, also titled '명령 프롬프트 - java EchoClient', shows the execution of 'java EchoClient' again. The output shows the client sending a greeting and receiving a response from the server.

*** 서버 실행창**

*** 클라이언트 1번 실행창**

*** 클라이언트 2번 실행창**

3. 간단한 채팅 클라이언트 / 서버 프로그래밍

□ 채팅 클라이언트 동작 방법

1. 채팅 클라이언트를 실행할때 사용자의 아이디와 접속할 서버의 IP 주소를 전달한다.
2. 다른 클라이언트가 접속하면, “xxx님이 접속했습니다.” 란 메시지를 출력
3. 다른 사람의 대화 내용이 클라이언트에게 키보드로 입력하는 중에도 전달 되어 화면에 출력된다.
4. 클라이언트에서 키보드로 문장을 입력한 후 엔터 키를 입력하면, 접속된 모든 클라이언트에 입력된 문자열이 전송된다.
5. 클라이언트를 종료하면, “xxx 님이 접속 종료했습니다.” 란 메시지를 출력

- * 클라이언트는 다음과 같은 내용을 서버에 전송할수 있어야 한다.
- 클라이언트의 ID 정보를 서버에 전송한다.
- 클라이언트에서 키보드로 입력된 문자열을 서버에 전송한다.
- 클라이언트의 접속이 종료될 경우, 접속이 종료 되었음을 서버에 알린다.

3. 간단한 채팅 클라이언트 / 서버 프로그래밍

□ 채팅 서버의 동작방법

1. 클라이언트 여러 개가 서버에 접속할수 있어야 한다.
2. 클라이언트가 접속할 경우, 서버는 이미 접속되어 있는 클라이언트에게 “xxx 님이 접속했습니다.” 라는 문자열을 전송해야한다.
3. 클라이언트가 문자열을 전송할 경우, 서버는 접속되어 있는 모든 클라이언트에게 전달받은 문자열을 전송해야 한다.
4. 클라이언트가 접속을 종료했을 경우, 서버는 접속되어 있는 클라이언트에게 “xxx님이 접속 종료했습니다.” 라는 문자열을 전송해야 한다.

CHROMATIC COLOR

- 클라이언트의 경우에는 입력과 출력을 동시에 할수 있어야한다.
- 서버는 클라이언트 여러 개로부터 입출력을 동시에 해야한다.
- 클라이언트 / 서버 동시에 각종일을 수행해야 하기 때문에 스레드 사용

ACHROMATIC COLOR

3. 간단한 채팅 클라이언트 / 서버 프로그래밍

□ 채팅 서버 프로그래밍 : ChatServer

- 채팅 서버는 클라이언트마다 스레드 하나를 생성해서 동작하게 한다.
- 클라이언트가 보낸 문자열을 접속한 모든 클라이언트에게 전송하기 위해서 스레드간에 접속한 클라이언트의 OutputStream을 공유하는 방법 필요
(스레드 간에 정보를 공유하기 위해서 해시맵(HashMap) 자료구조 이용)

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

public class ChatServer {

    public static void main(String[] args) {
        try{
            ServerSocket server = new ServerSocket(10001);
            System.out.println("접속을 기다립니다.");
            HashMap hm = new HashMap();
            while(true){
                Socket sock = server.accept();
                ChatThread chatthread = new ChatThread(sock, hm);
                chatthread.start();
            } // while
        } catch (Exception e) {
            System.out.println(e);
        }
    } // main
}

class ChatThread extends Thread{
    private Socket sock;
    private String id;
    private BufferedReader br;
    private HashMap hm;
    private boolean initFlag = false;
```

3. 간단한 채팅 클라이언트 / 서버 프로그래밍

```
public ChatThread(Socket sock, HashMap hm) {
    this.sock = sock;
    this.hm = hm;
    try {
        PrintWriter pw = new PrintWriter(new OutputStreamWriter(sock.getOutputStream()));
        br = new BufferedReader(new InputStreamReader(sock.getInputStream()));
        id = br.readLine();
        broadcast(id + "님이 접속하였습니다.");
        System.out.println("접속한 사용자의 아이디는 " + id + "입니다.");
        synchronized(hm) {
            hm.put(this.id, pw);
        }
        initFlag = true;
    } catch (Exception ex) {
        System.out.println(ex);
    }
} // 생성자

public void run() {
    try {
        String line = null;
        while ((line = br.readLine()) != null) {
            if (line.equals("/quit"))
                break;
            if (line.indexOf("/to ") == 0) {
                sendmsg(line);
            } else
                broadcast(id + " : " + line);
        }
    } catch (Exception ex) {
        System.out.println(ex);
    } finally {
        synchronized(hm) {
            hm.remove(id);
        }
        broadcast(id + "님이 접속 종료하였습니다.");
        try {
            if (sock != null)
                sock.close();
        } catch (Exception ex) {}
    }
} // run
```

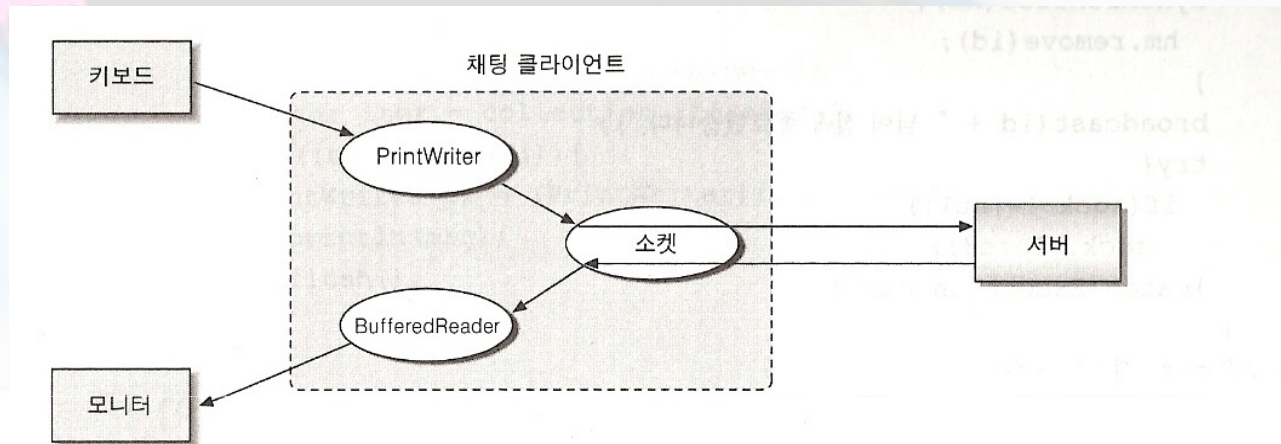
3. 간단한 채팅 클라이언트 / 서버 프로그래밍

```
public void sendmsg(String msg){
    int start = msg.indexOf(" ") + 1;
    int end = msg.indexOf(" ", start);
    if(end != -1){
        String to = msg.substring(start, end);
        String msg2 = msg.substring(end+1);
        Object obj = hm.get(to);
        if(obj != null){
            PrintWriter pw = (PrintWriter)obj;
            pw.println(id + " 님이 다음의 귓속말을 보내셨습니다. :" + msg2);
            pw.flush();
        } // if
    }
} // sendmsg

public void broadcast(String msg){
    synchronized(hm){
        Collection collection = hm.values();
        Iterator iter = collection.iterator();
        while(iter.hasNext()){
            PrintWriter pw = (PrintWriter)iter.next();
            pw.println(msg);
            pw.flush();
        }
    }
} // broadcast
}
```

3. 간단한 채팅 클라이언트 / 서버 프로그래밍

□ 채팅 클라이언트 프로그래밍 : ChatClient

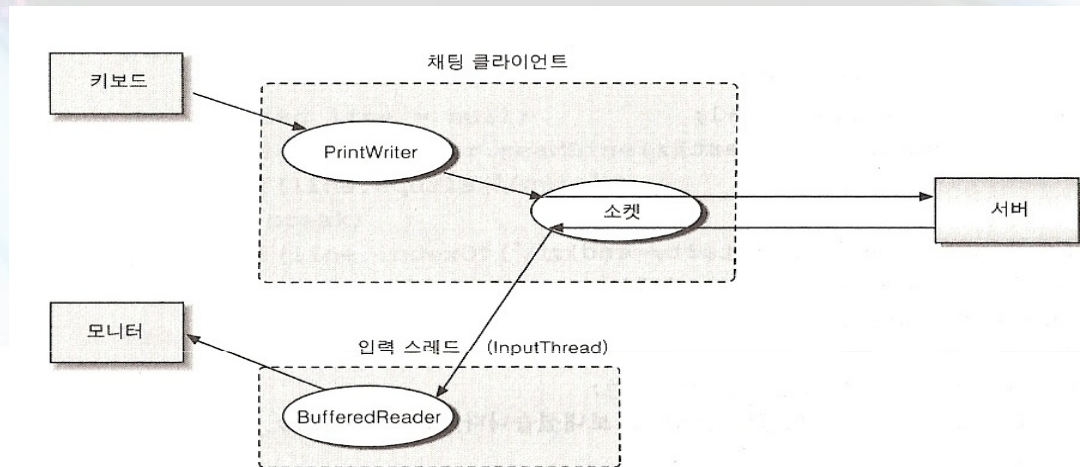


[그림 8-19] 채팅 클라이언트의 작동 모습

- 채팅 클라이언트는 윈도우용 프로그램이 아니라 명령창에서 동작하는 프로그램
- 키보드로부터 입력 받은 문자열을, 소켓을 통해서 구한 PrintWriter를 이용해서 출력한다.(서버에 문자열이 전송된다.)
- **문제점** : 키보드로 사용자가 글을 입력하고 있는 중간에도 서버에서 다른 클라이언트에서 전송한 문자열을 소켓을 통해서 전달 받을 수있다.

3. 간단한 채팅 클라이언트 / 서버 프로그래밍

□ 채팅 클라이언트 프로그래밍 : ChatClient



[그림 8-20] 서버가 전달하는 문자열을 처리하기 위한 스레드가 있는 채팅 클라이언트

- 메인스레드가 키보드로부터 입력을 받을 때에는 다른 일 할수 없다.
- 문제점 해결 : 입출력을 따로 하는 동작하는 클라이언트 작성
- 키보드로 부터 문자열을 입력 받아 서버로 전달 : 메인스레드
- 서버로 부터 전송받은 문자열은 스레드를 작동시켜서 처리해야한다.

3. 간단한 채팅 클라이언트 / 서버 프로그래밍

□ 채팅 클라이언트 프로그래밍 : ChatClient (예제)

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

public class ChatClient {

    public static void main(String[] args) {
        if (args.length != 2) {
            System.out.println("사용법 : java ChatClient id 접속할서버ip");
            System.exit(1);
        }
        Socket sock = null;
        BufferedReader br = null;
        PrintWriter pw = null;
        boolean endflag = false;
        try {
            sock = new Socket(args[1], 10001);
            pw = new PrintWriter(new OutputStreamWriter(sock.getOutputStream()));
            br = new BufferedReader(new InputStreamReader(sock.getInputStream()));
            BufferedReader keyboard = new BufferedReader(new InputStreamReader(System.in));
            // 사용자의 id를 전송한다.
            pw.println(args[0]);
            pw.flush();
            InputThread it = new InputThread(sock, br);
            it.start();
            String line = null;
            while ((line = keyboard.readLine()) != null) {
                pw.println(line);
                pw.flush();
                if (line.equals("/quit")) {
                    endflag = true;
                    break;
                }
            }
            System.out.println("클라이언트의 접속을 종료합니다.");
        } catch (Exception ex) {
            if (!endflag)
                System.out.println(ex);
        } finally {
            try {
                if (pw != null)
                    pw.close();
            } catch (Exception ex) {}
            try {
                if (br != null)
                    br.close();
            } catch (Exception ex) {}
            try {
                if (sock != null)
                    sock.close();
            } catch (Exception ex) {}
        } // finally
    } // main
} // class
```

3. 간단한 채팅 클라이언트 / 서버 프로그래밍

□ 채팅 클라이언트 프로그래밍 : ChatClient (예제)

```
class InputThread extends Thread{
    private Socket sock = null;
    private BufferedReader br = null;
    public InputThread(Socket sock, BufferedReader br){
        this.sock = sock;
        this.br = br;
    }
    public void run(){
        try{
            String line = null;
            while((line = br.readLine()) != null){
                System.out.println(line);
            }
        } catch (Exception ex){
        } finally{
            try{
                if(br != null)
                    br.close();
            } catch (Exception ex){}
            try{
                if(sock != null)
                    sock.close();
            } catch (Exception ex){}
        }
    } // InputThread
}
```

3. 간단한 채팅 클라이언트 / 서버 프로그래밍

□ 채팅 클라이언트 / 서버 프로그래밍 (실행결과)

```
C:\>명령 프롬프트 - java ChatServer
C:\>C:\java\workspace\workspace\자바 io nio\8장 TCP\bin>java ChatServer
접속을 기다립니다.
접속한 사용자의 아이디는 urstrory입니다.
접속한 사용자의 아이디는 carami입니다.

C:\>명령 프롬프트
C:\>C:\java\workspace\workspace\자바 io nio\8장 TCP\bin>cd bin
C:\>C:\java\workspace\workspace\자바 io nio\8장 TCP\bin>java ChatClient urstrory localhost
carami님이 접속하였습니다.
carami : 안녕하세요
예 안녕하세요
urstrory : 예 안녕하세요
carami : 정말로 채팅하는 것 같은 프로그램입니다.
예 정말로 채팅하는 것 같이 느껴집니다.
urstrory : 예 정말로 채팅하는 것 같이 느껴집니다.
/quit
클라이언트의 접속을 종료합니다.
C:\>C:\java\workspace\workspace\자바 io nio\8장 TCP\bin>

C:\>명령 프롬프트
C:\>C:\java\workspace\workspace\자바 io nio\8장 TCP\bin>cd bin
C:\>C:\java\workspace\workspace\자바 io nio\8장 TCP\bin>java ChatClient carami localhost
안녕하세요
carami : 안녕하세요
urstrory : 예 안녕하세요
정말로 채팅하는 것 같은 프로그램입니다.
carami : 정말로 채팅하는 것 같은 프로그램입니다.
urstrory : 예 정말로 채팅하는 것 같이 느껴집니다.
urstrory님이 접속 종료하였습니다.
/quit
클라이언트의 접속을 종료합니다.
C:\>C:\java\workspace\workspace\자바 io nio\8장 TCP\bin>
```

3. 간단한 채팅 클라이언트 / 서버 프로그래밍

□ 윈도우 채팅 클라이언트

- Frame 클래스로 구현
- ActionListener로 컴포넌트별 이벤트 처리
 - ◆ 아이디 입력 TextField와 문자열 입력 TextField에 ActionListener 추가

```
public WinChatClient(String ip){
    super("채팅 클라이언트");
    cardLayout = new CardLayout();
    setLayout(cardLayout);
    Panel loginPanel = new Panel();
    loginPanel.setLayout(new BorderLayout());
    loginPanel.add("North", new Label("아이디를 입력하여 주신후 엔터를 입력하여 주세요."));
    idTF = new TextField(20);
    idTF.addActionListener(this);
    Panel c = new Panel();
    c.add(idTF);
    loginPanel.add("Center", c);
    add("login", loginPanel);
    Panel main = new Panel();
    main.setLayout(new BorderLayout());
    input = new TextField();
    input.addActionListener(this);
    display = new TextArea();
    display.setEditable(false);
    main.add("Center", display);
    main.add("South", input);
    add("main", main);
}
```

3. 간단한 채팅 클라이언트 / 서버 프로그래밍

□ 윈도우 채팅 클라이언트

– 서버 접속

```
...
sock = new Socket(ip, 10001);
pw = new PrintWriter(new OutputStreamWriter(sock.getOutputStream()));
br = new BufferedReader(new InputStreamReader(sock.getInputStream()));
...
```

– 윈도우 Frame에 WindowListener 추가

```
Public WinChatClient(String ip){
...
addWindowListener(new WindowAdapter(){
    public void windowClosing(WindowEvent e){
        pw.println("/quit");
        pw.flush();
        try{
            sock.close();
        }catch(Exception ex){}
        System.out.println("종료합니다.");
        System.exit(0);
    }
});
setVisible(true);
}
```

3. 간단한 채팅 클라이언트 / 서버 프로그래밍

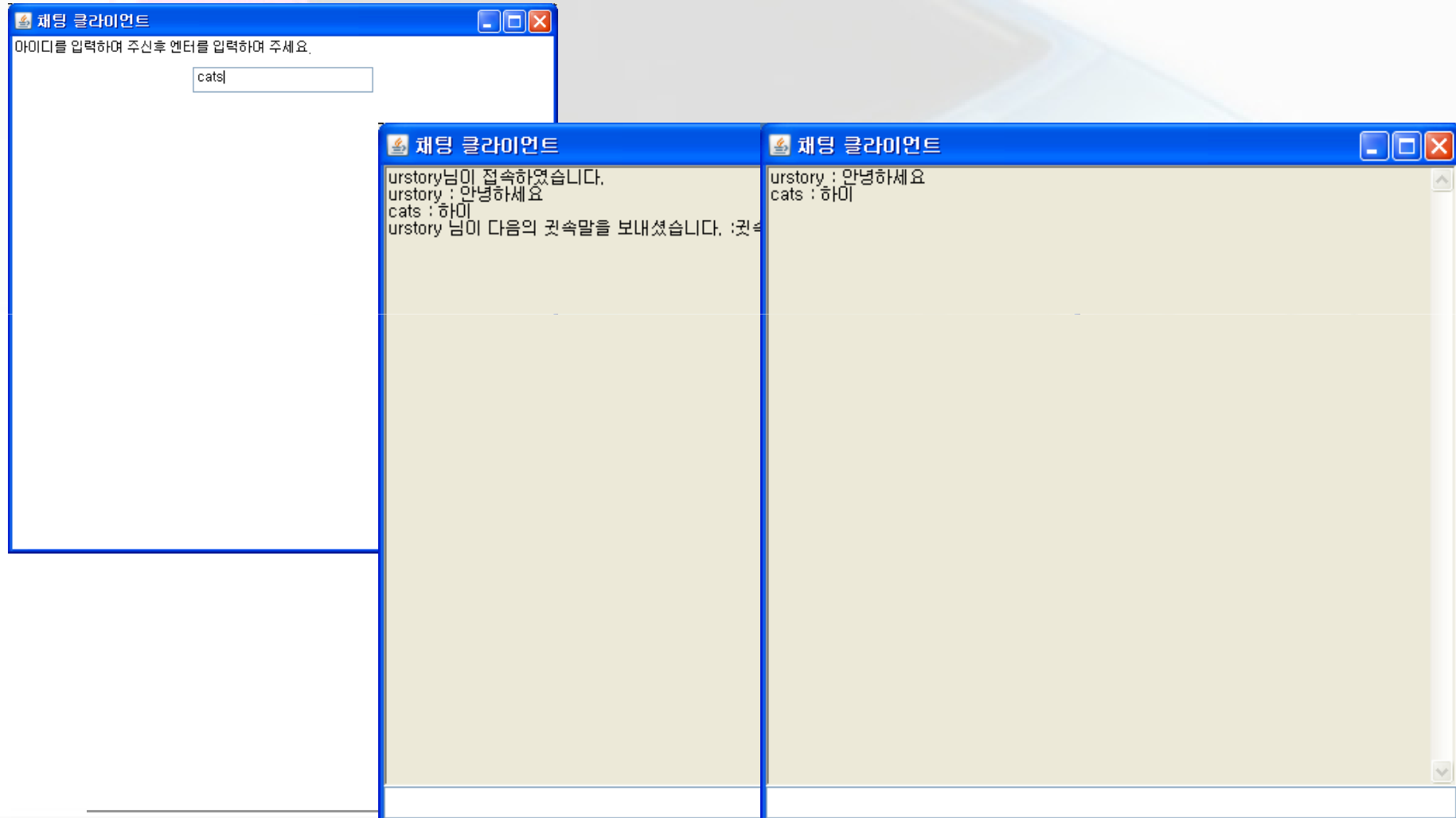
□ 윈도우 채팅 클라이언트

– ActionEvent를 처리하는 actionPerformed() 추가

```
public void actionPerformed(ActionEvent e) {
    if(e.getSource() == idTF){
        String id = idTF.getText();
        if(id == null || id.trim().equals("")){
            System.out.println("아이디를 다시 입력하여 주세요.");
            return;
        }
        pw.println(id.trim());
        pw.flush();
        WinInputThread wit = new WinInputThread(sock, br);
        wit.start();
        cardLayout.show(this, "main");
        input.requestFocus();
    }else if(e.getSource() == input){
        String msg = input.getText();
        pw.println(msg);
        pw.flush();
        if(msg.equals("/quit")){
            try{
                sock.close();
            }catch(Exception ex){}
            System.out.println("종료합니다.");
            System.exit(1);
        }
        input.setText("");
        input.requestFocus();
    }
} // actionPerformed
```

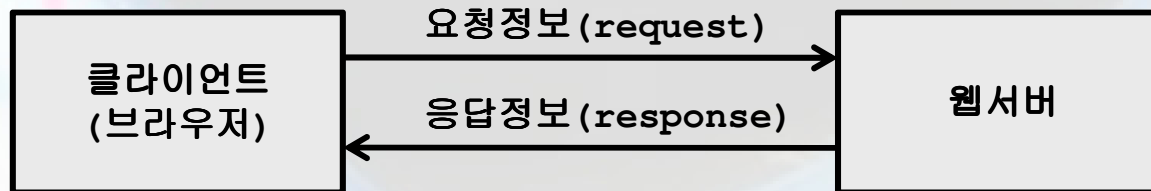
3. 간단한 채팅 클라이언트 / 서버 프로그래밍

□ 윈도우 채팅 클라이언트



4. 간단한 웹서버 프로그래밍

□ HTTP 프로토콜



□ 브라우저의 요청정보 출력

- 80번 포트에서 클라이언트 접속 대기

```
ServerSocket ss = new ServerSocket(80);  
sock = ss.accept();
```

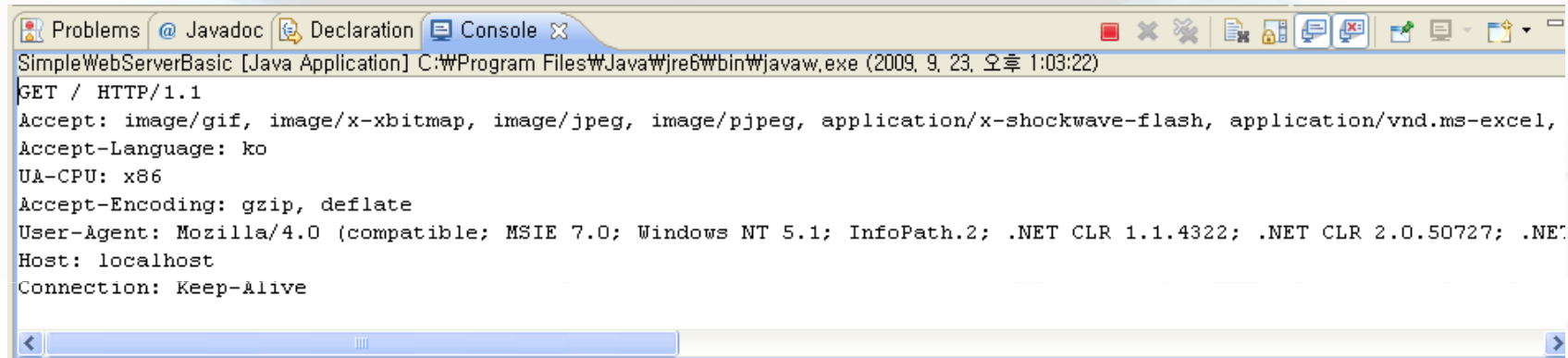
- 소켓에서 InputStream 얻은 후, BufferedReader로 변환하여 readLine() 사용

```
br = new BufferedReader(new  
InputStreamReader(sock.getInputStream()));  
String line = null;  
while((line = br.readLine()) != null)  
    System.out.println(line);
```


4. 간단한 웹서버 프로그래밍

□브라우저의 요청정보 출력

– 웹브라우저에서 <http://localhost>로 서버에 접속

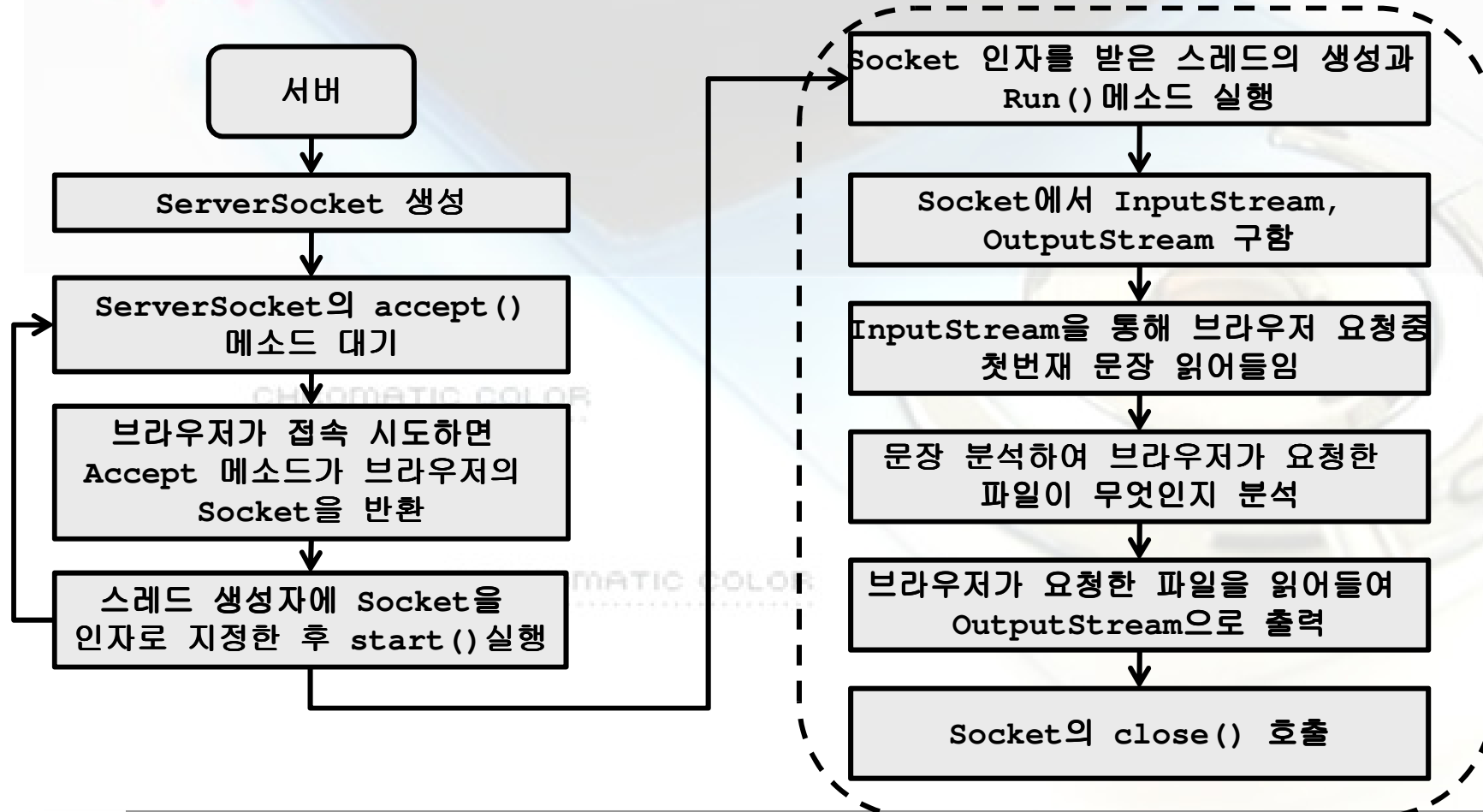


```
SimpleWebServerBasic [Java Application] C:\Program Files\Java\jre6\bin\javaw.exe (2009. 9. 23. 오후 1:03:22)
GET / HTTP/1.1
Accept: image/gif, image/x-xbitmap, image/jpeg, image/pjpeg, application/x-shockwave-flash, application/vnd.ms-excel,
Accept-Language: ko
UA-CPU: x86
Accept-Encoding: gzip, deflate
User-Agent: Mozilla/4.0 (compatible; MSIE 7.0; Windows NT 5.1; InfoPath.2; .NET CLR 1.1.4322; .NET CLR 2.0.50727; .NET
Host: localhost
Connection: Keep-Alive
```

4. 간단한 웹서버 프로그래밍

□ 간단한 웹서버 프로그래밍

– 여러 브라우저의 요청을 처리하도록 설계



4. 간단한 웹서버 프로그래밍

□ 간단한 웹서버 프로그래밍

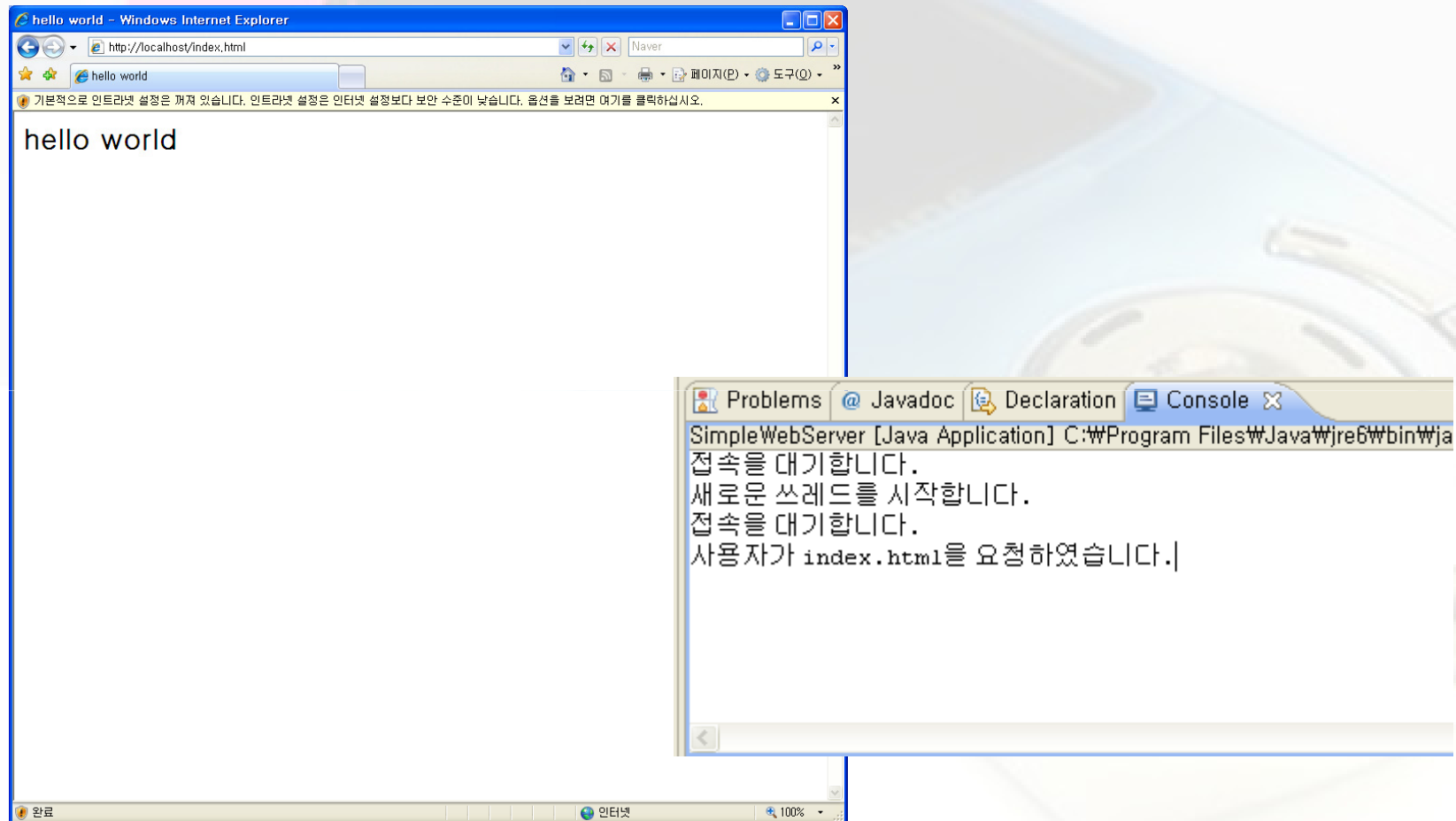
– 브라우저의 요청은 `HttpThread` 객체가 담당

```
public HttpThread(Socket sock){
    this.sock = sock;
    try{
        br = new BufferedReader(new InputStreamReader(sock.getInputStream()));
        pw = new PrintWriter(new OutputStreamWriter(sock.getOutputStream()));
    }catch(Exception ex){
        System.out.println(ex);
    }
}
```

– 요청 내용은 `HttpThread`의 `run()` 메소드에서 처리

```
public void run(){
    BufferedReader fbr = null;
    try{
        String line = br.readLine();
        int start = line.indexOf(" ") + 2;
        int end = line.lastIndexOf("HTTP") - 1;
        String filename = line.substring(start, end);
        if(filename.equals(""))
            filename = "index.html";
        System.out.println("사용자가 " + filename + "을 요청하였습니다.");
        fbr = new BufferedReader(new FileReader(filename));
        String fline = null;
        while((fline = fbr.readLine()) != null){
            pw.println(fline);
            pw.flush();
        }
    }
}
```

4. 간단한 웹서버 프로그래밍

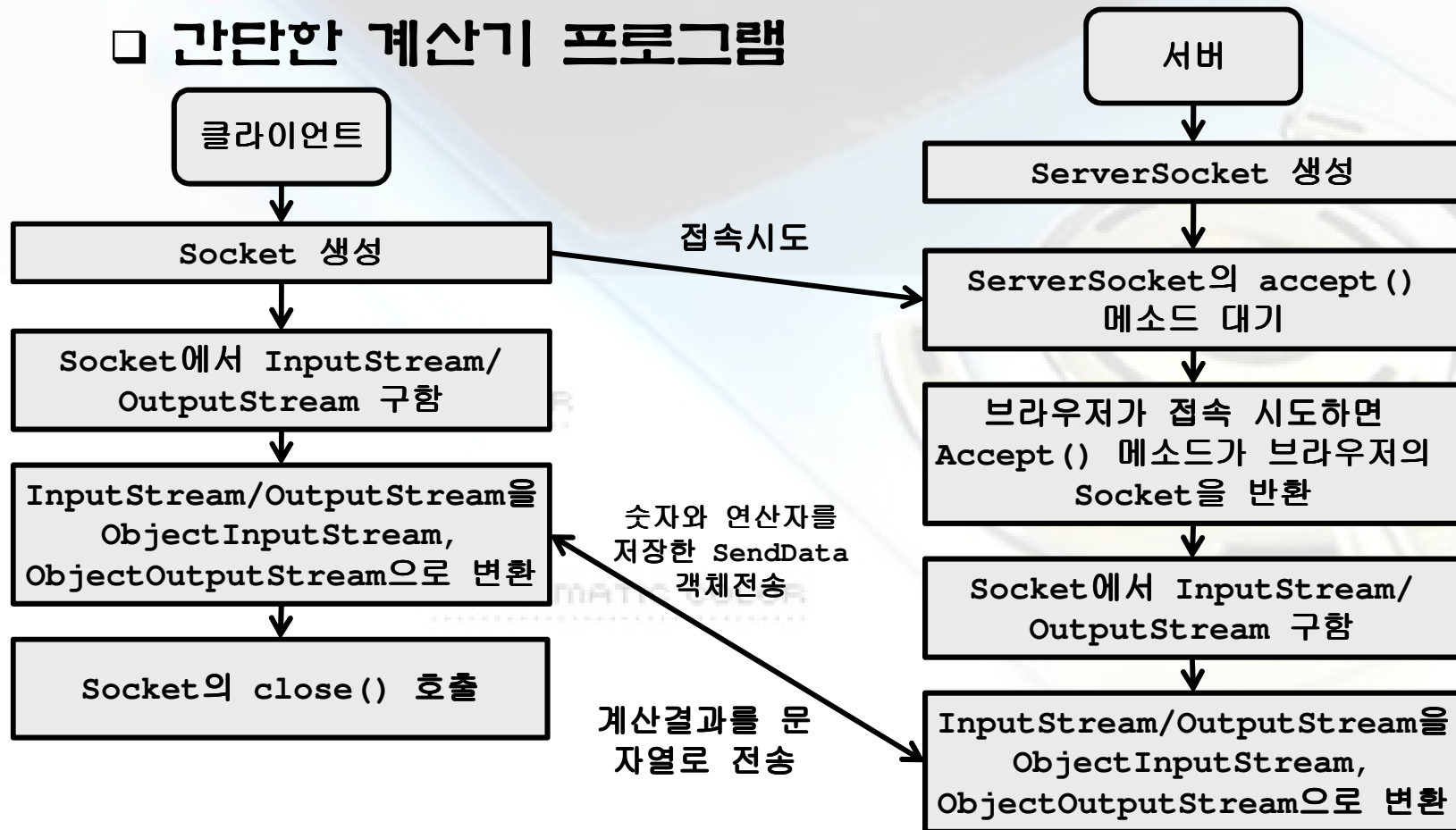


5. 객체직렬화를 이용한 네트워크 프로그래밍

□ ObjectOutputStream/ObjectInputStream

- 객체 직렬화를 이용하여 네트워크를 통해 객체 전송 가능

□ 간단한 계산기 프로그램



5. 객체직렬화를 이용한 네트워크 프로그래밍

□ 객체직렬화 객체: SendData

```
public class SendData implements Serializable {  
    private int op1;  
    private int op2;  
    private String opcode;  
    ...  
}
```

□ 계산서버

– 10005번 포트에서 동작하여 클라이언트 접속 대기

```
public static void main(String[] args) {  
    Socket sock = null;  
    ObjectOutputStream oos = null;  
    ObjectInputStream ois = null;  
    try{  
        ServerSocket ss = new ServerSocket(10005);  
        System.out.println("클라이언트의 접속을 대기합니다.");  
        sock = ss.accept();  
  
        oos = new ObjectOutputStream(sock.getOutputStream());  
        ois = new ObjectInputStream(sock.getInputStream());  
  
        ...  
    }  
}
```

5. 객체직렬화를 이용한 네트워크 프로그래밍

□계산서버

- 클라이언트로부터 SendData 객체를 읽어들이어 두개의 숫자와 연산자를 얻는다

```
Object obj = null;
while((obj = ois.readObject()) != null){
    SendData sd = (SendData)obj;
    int op1 = sd.getOp1();
    int op2 = sd.getOp2();
    String opcode = sd.getOpcode();
    if(opcode.equals("+")){
        oos.writeObject(op1 + " + " + op2 + " = " + (op1 + op2));
        oos.flush();
    }
    ...
}
```

□계산 클라이언트

- 키보드로 숫자 두개와 연산자를 입력받아 서버로 전송
- 서버로부터 String 객체를 받아 결과 출력

5. 객체직렬화를 이용한 네트워크 프로그래밍

□ 계산 클라이언트

```
sock = new Socket(args[0], 10005);  
  
oos = new ObjectOutputStream(sock.getOutputStream());  
ois = new ObjectInputStream(sock.getInputStream());  
BufferedReader keyboard = new BufferedReader(new InputStreamReader(System.in));  
...
```

```
SendData s = new SendData(op1, op2, opcode);  
oos.writeObject(s);  
oos.flush();  
String msg = (String)ois.readObject();  
System.out.println(msg);
```

□ 객체직렬화

- 네트워크를 통해서도 쉽게 객체를 주고받을 수 있다
 - ◆ RMI 기술에서 내부적으로 사용
- 내부적인 마샬링/언마샬링 과정으로 추가적인 부하가 발생

5. 객체지향학을 이용한 네트워크 프로그래밍

□ 실행결과

```
C:\WINDOWS\system32\cmd.exe - java ObjectCalculatorServer

D:\Data\Program\NIO\source\8-110bjectCalculatorServer\bin>java ObjectCalculatorSe
rver
클라이언트의 접속을 대기합니다.
결과를 전송하였습니다.
```

```
ObjectCalculatorClient [Java Application] C:\Program Files\Java\jre6\bin\javaw.exe (2009. 9. 2
첫번째 숫자를 입력하여 주세요. (잘못 입력된 숫자는 0으로 처리합니다.)
100
두번째 숫자를 입력하여 주세요. (잘못 입력된 숫자는 0으로 처리합니다.)
50
+, -, *, / 중에 하나를 입력하여 주세요. ( 잘못입력하면 + 로 처리합니다.)
*
100 * 50 = 5000
계속 계산하시겠습니까? (Y/n)
```