자바 네트워크

소프트웨어교육원 박경태
comsi.java@gmail.com

자바 네트워크

- 스트림
  - 바이트와 문자 입출력 스트림
- TCP/IP 서버/클라이언트 소켓
  - Echo 서버, 서버 시간 전송
- UDP 프로토콜과 URLConnection
- 쓰레드와 멀티태스킹
- 서버와 클라이언트 통신 프로그램
  - 1:1채팅, 1:N 채팅과 귓속말

스트림

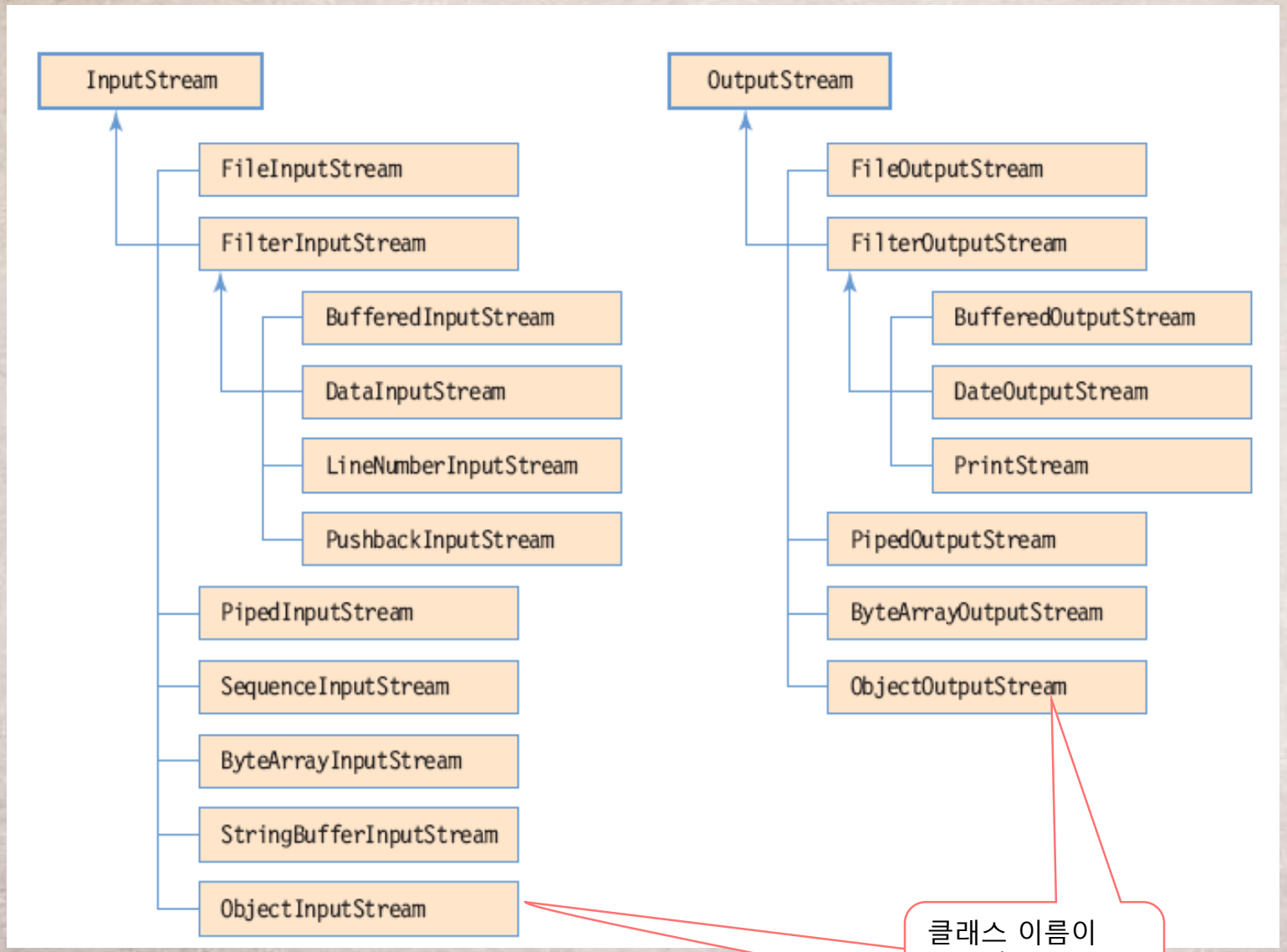
스트림(Stream)

- 자바의 스트림
  - 자바의 스트림은 입출력 장치와 프로그램을 연결하며, 이들 사이의 데이타 흐름을 처리하는 소프트웨어 모듈
  - 입력 스트림(데이터 read)
    - 입력 장치로부터 자바 프로그램으로 전달되는 데이터의 흐름 혹은 데이터 전송 소프트웨어 모듈
  - 출력 스트림(데이터 write)
    - 자바 프로그램에서 출력 장치로 보내는 데이터의 흐름 혹은 데이터 전송 소프트웨어 모듈
- 입출력 스트림 기본 단위 : 바이트
- 자바 입출력 스트림 특징

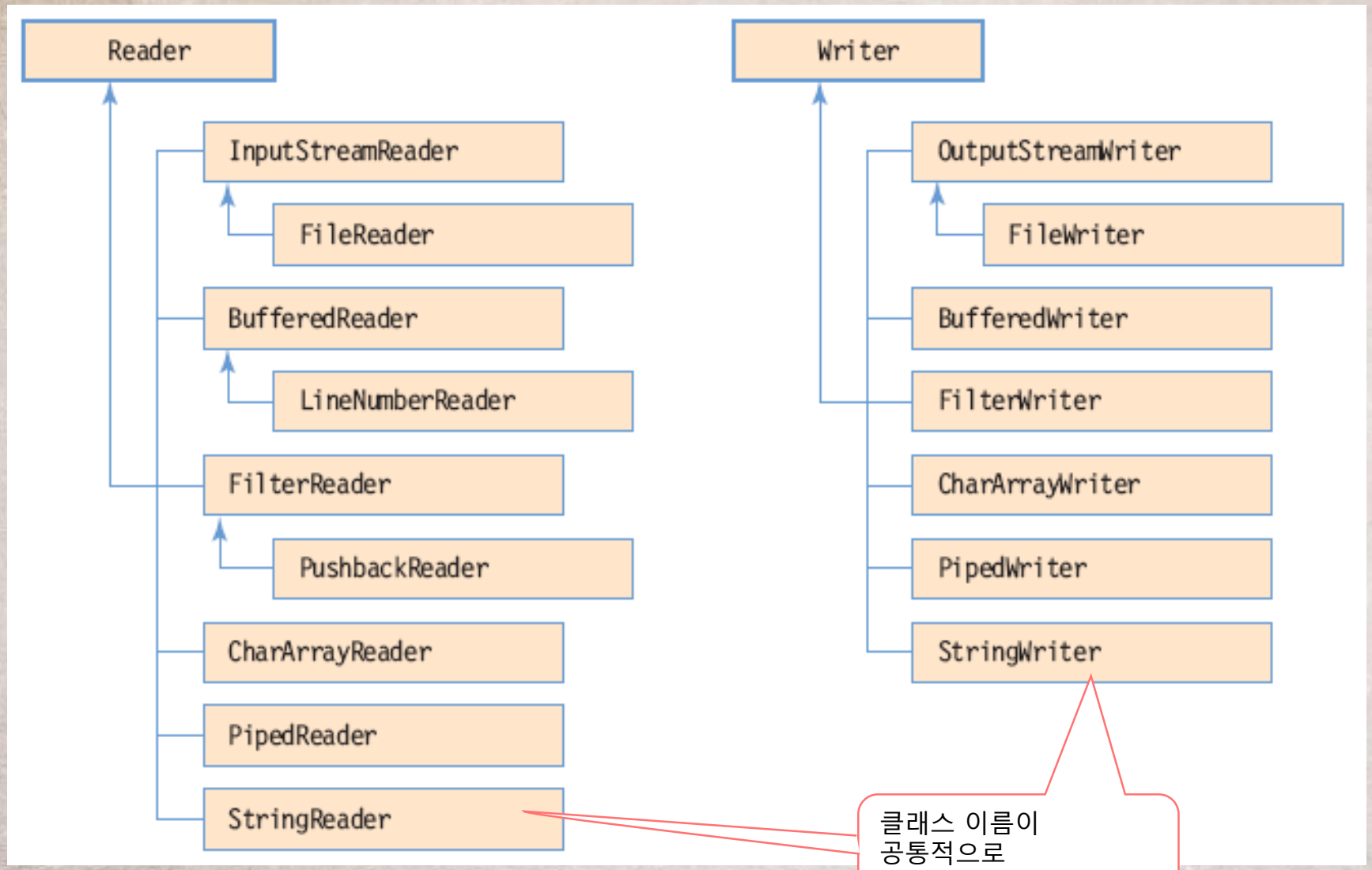
자바의 입출력 스트림 종류

- 처리하는 데이터에 따라 바이트 입출력 스트림과 문자 입출력 스트림
  - 바이트 입출력 스트림**(이진수 데이터)**
    - 단순 바이트의 흐름으로만 처리
    - 예) 바이너리 파일을 읽는 입력 스트림
  - 문자 입출력 스트림**(문자)**
    - 문자의 흐름으로 처리.
    - 문자가 아닌 바이너리 데이터는 스트림에서 처리하지 못함
    - 예) 텍스트 파일을 읽는 입력 스트림
- JDK는 입출력 스트림을 구현한 다양한 클래스 제공

JDK의 바이트 스트림 클래스 계층 구조



JDK의 문자 스트림 클래스 계층 구조



바이트 입출력 스트림

OutputStream과 InputStream 클래스

OutputStream	
void close() throws IOException	OutputStream을 닫는다.
void flush() throws IOException	버퍼에 남은 스트림을 출력한다.
abstract void write(int i) throws IOException	정수 i의 하위 8bit를 출력한다.
void write(byte buf[]) throws IOException	buf의 내용을 출력한다.
void write(byte buf[], int index, int size) throws IOException	buf의 index위치부터 size만큼의 바이트를 출력한다.

- 특정한 매체에 데이터를 출력 (OutputStream 메소드 이용)
  - FileOutputStream → 파일
  - TelnetOutputStream → 네트워크
  - ByteOutputArrayStream → 확장 가능한 바이트 배열

write(int b)

- 0 에서 255까지 정수를 메소드 인자로 받아 바이트를 출력 스트림에 써넣음
- abstract 메소드
- 정수형이지만 실제로 부호 없는 바이트를 데이터 타입으로 함
- 255 이상의 숫자를 입력하면 1 바이트만 인식하고 3바이트는 무시
- 데이터 전송의 효율성을 위하여 write(byte[] data), write(byte[] data, int offset, int length) 메소드 이용

OutputStream 예제) ASCII 코드 32~126에 해당하는 문자 출력

DisplayCharacter.java

```
1
2  public class DisplayCharacter {
3
4      public static void main(String[] args) {
5          for (int i=32; i < 127; i++) {
6              System.out.write(i); // 32부터 127까지 정수값을 출력
7              if (i % 8 == 7) {
8                  System.out.write('\n'); // 줄 넘김
9              }else {
10                 System.out.write('\t'); //
11             }
12         }
13         System.out.write('\n'); // 줄 넘김
14     }
15
16 }
17
```

Problems | Javadoc | Console | Progress

<terminated> DisplayCharacter [Java Application] C:₩Program Files₩Java₩jre1.8.0_171₩bin₩javaw.exe (2018. 11. 17. 오후 10:05:01)

```
        !       "       #       $       %       &       '
(       )       *       +       ,       -       .       /
0       1       2       3       4       5       6       7
8       9       :       ;       <       =       >       ?
@       A       B       C       D       E       F       G
H       I       J       K       L       M       N       O
P       Q       R       S       T       U       V       W
X       Y       Z       [       \       ]       ^       _
`       a       b       c       d       e       f       g
h       i       j       k       l       m       n       o
p       q       r       s       t       u       v       w
x       y       z       {       |       }       ~
```

OutputStream 예제) 문자데이터를 바이트 데이터로 변환시키고 출력

DisplayCharacter.java | DisplayString.java

```
1  import java.io.IOException;
2
3  public class DisplayString {
4
5      public static void main(String[] args) throws IOException {
6          byte[] buffer;
7          for (int i=0; i<args.length;i++) {
8              buffer = args[i].getBytes();
9              System.out.write(buffer);
10             System.out.write('\n');
11         }
12     }
13 }
14
```

Main | Arguments | JRE | Classpath | Source | Environment

Program arguments:

Hello Java! & Java network

Variables...

Problems | Javadoc | Console | Progress

<terminated> DisplayString [Java Application] C:₩Program Files₩Java₩jre1.8.0_171₩bin₩javaw.exe (2018. 11. 17. 오후 10:14:30)

```
Hello
Java!
&
Java
network
```

OutputStream과 InputStream 클래스

InputStream	
int available() throws IOException	현재 읽을 수 있는 바이트 수를 얻는다.
void close() throws IOException	InputStream을 닫는다
void mark(int readlimit)	InputStream에서 현재 읽고 있는 위치를 표시한다.
boolean markSupported()	해당 InputStream에서 mark()로 지정된 지점이 있는지 여부를 체크한다.
abstract int read() throws IOException	InputStream에서 한 바이트를 읽어서 int 값으로 반환한다.
int read(byte[] b) throws IOException	byte[] b 만큼의 데이터를 읽어서 b에 저장하고 읽은 바이트 수를 반환한다
int read(byte[] b, int off, int len) thrwos IOException	len 만큼을 읽어 byte[] b의 off위치에 저장하고 읽은 바이트 수를 반환한다
void reset()	모든 mark() 를 초기화한다.
long skip(long n) throws IOException	InputStream에서 n 바이트 만큼 데이터를 스킵하고 바이트 수를 반환한다

- 특정한 매체의 데이터 읽기 (OutputStream 메소드 이용)
  - FileInputStream → 파일
  - TelnetInputStream → 네트워크
  - ByteArrayInputStream → 확장 가능한 바이트 배열

InputStream 예제) 바이트 데이터 읽어서 출력

DisplayCharacter.java | DisplayString.java | ReadCharacter.java

```java
import java.io.IOException;

public class ReadCharacter {

    public static void main(String[] args) throws IOException {
        int data;
        while( (data = System.in.read()) >= 0 ) // 종료(Ctrl+Z) = -1
            System.out.write(data);

    }

}
```

System.in 은 표준 키보드 입력을 의미

Problems | Javadoc | Console | Progress

<terminated> ReadCharacter [Java Application] C:₩Program Files₩Java₩jre1.8.0_171₩bin₩javaw.exe (2018. 11. 17. 오후 10:21:49)

```
1
1
abcdef
abcdef
rrrr
rrrr
Hello world
Hello world
```

InputStream 예제) 바이트 데이터 읽어서 출력(배열)

DisplayCharacter.java | DisplayString.java | ReadCharacter.java | ReadCharacterOffset.java

```
1  import java.io.IOException;
2
3  public class ReadCharacterOffset {
4
5      public static void main(String[] args) {
6          try {
7              int bufferSize = 80;
8              int size=0;
9              int dataRead;
10             byte buffer[] = new byte[bufferSize];
11
12             while((dataRead = System.in.read(buffer, size, bufferSize-size))  >= 0) {
13                 size += dataRead;
14             }
15             System.out.write(buffer,0,size);
16         }catch(IOException e) {
17             System.out.println("스트림으로부터 데이터를 읽을 수 없습니다.");
18         }
19
20     }
21 }
22
```

Problems | Javadoc | Console | Progress

<terminated> ReadCharacterOffset [Java Application] C:₩Program Files₩Java₩jre1.8.0_171₩bin₩javaw.exe (2018. 11. 17. 오후 10:30:58)

```
Hello World
Read Character Offset test
Hello World
Read Character Offset test
```

바이트 스트림 클래스

- 바이트 스트림
  - 바이트 단위의 바이너리 값을 읽고 쓰는 스트림
- 바이트 스트림 클래스
  - java.io 패키지에 포함
  - InputStream/OutputStream
    - 추상 클래스
    - **바이트 입출력 스트림을 다루는 모든 클래스의 수퍼 클래스**
  - FileInputStream/FileOutputStream
    - **파일**로부터 바이트 단위로 읽거나 저장하는 클래스
    - 바이너리 파일의 입출력
  - DataInputStream/DataOutputStream
    - 자바의 기본 데이터 타입의 값(변수)을 바이너리 값 그대로 입출력
    - 문자열도 바이너리 형태로 입출력

FileInputStream을 이용한 파일 읽기

- 파일 전체를 읽어 화면에 출력하는 코드 샘플

```
FileInputStream fin = new FileInputStream("c:₩₩test.txt");

int c;

while((c = fin.read()) != -1) {

   System.out.print((char)c);

}

fin.close();
```

입력 바이트 스트림객체를 생성하고
C:₩test.txt 파일 오픈

파일 끝까지 반복하며 한 바이트씩 c에 읽어 들임.
파일의 끝을 만나면 read()는 -1 리턴

바이트 c를 문자로 변환하여 화면에 출력

스트림을 닫음. 파일도 닫힘.
더 이상 스트림으로부터 읽을 수 없음

예제) 윈도우에 있는 system.ini 파일을 읽어 화면에 출력하기

FileInputStreamEx.java | *FileOutputStreamEx.java

```java
import java.io.FileInputStream;
import java.io.IOException;


public class FileInputStreamEx {

    public static void main(String[] args) {
        FileInputStream fin = null;

        try{
            fin = new FileInputStream("c:\\windows\\system.ini");

            int  c;

            while((c=fin.read()) != -1){
                System.out.print((char)c);
            }
            fin.close();
        }catch(IOException e){
            System.out.println("입출력 오류");
        }
    }
}
```

<terminated> FileInputStreamEx [Java Application] C:₩Pro

```
; for 16-bit app support
[386Enh]
woafont=dosapp.fon
EGA80WOA.FON=EGA80WOA.FON
EGA40WOA.FON=EGA40WOA.FON
CGA80WOA.FON=CGA80WOA.FON
CGA40WOA.FON=CGA40WOA.FON

[drivers]
wave=mmdrv.dll
timer=timer.drv

[mci]
```

FileOutputStream을 이용한 파일 쓰기

- 바이너리 값을 파일에 저장하는 바이트 스트림 코드

```
FileOutputStream fout = new FileOutputStream("c:₩₩test.out");

int num[]={1,4,-1,88,50};
byte b[]={7,51,3,4,1,24};

for(int i=0; i<num.length; i++)
  fout.write(num[i]);

fout.write(b);

fout.close();
```

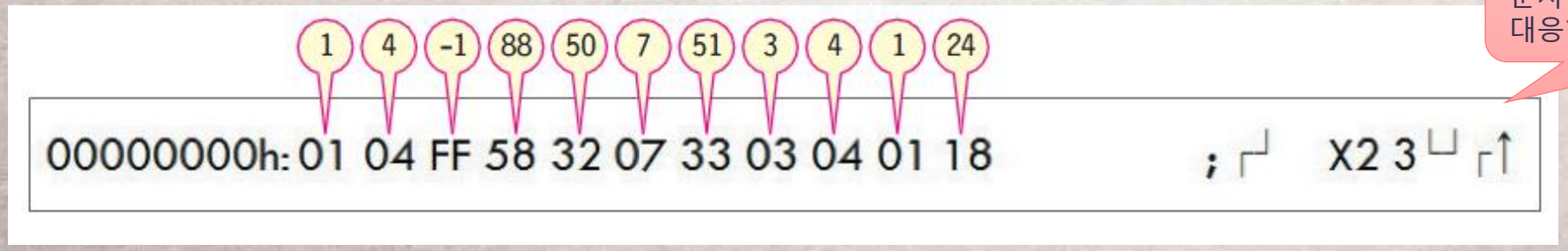
출력 바이트 스트림 객체를 생성하고 C:₩test.out 파일 오픈

파일에 정수 값(바이너리)을 그대로 기록

파일에 바이트 배열(바이너리) 값을 그대로 기록

스트림을 닫음. 파일도 닫힘. 더 이상 스트림으로부터 읽을 수 없음

파일에 있는 각 바이너리 값들은 문자 정보가 아님. 바이너리 값에 대응하는 그래픽 심볼들

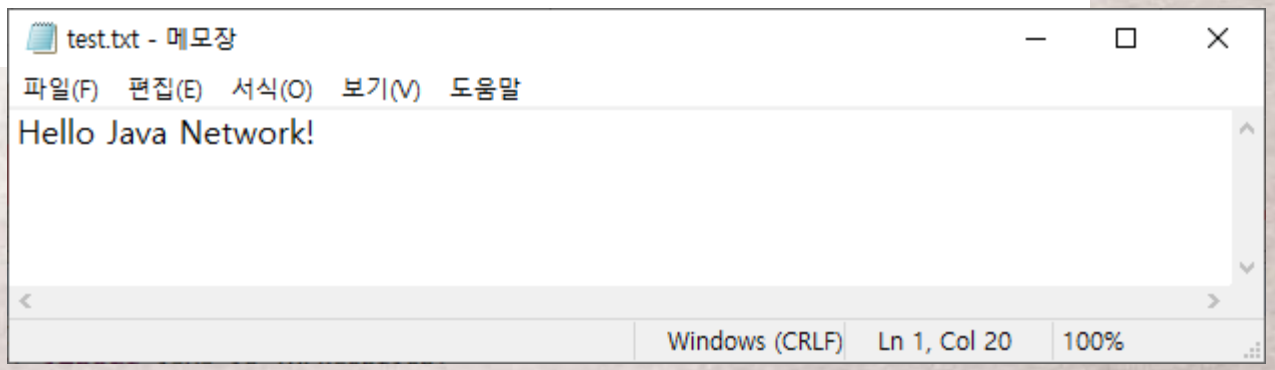


test.out 파일의 내부

예제) 문자열을 text.txt 파일로 출력하기

DisplayCharacter.java | DisplayString.java | ReadCharacter.java | FileOutputStreamEx.java

```java
import java.io.FileOutputStream;
import java.io.IOException;

public class FileOutputStreamEx {

    public static void main(String[] args) {
        String text = "Hello Java Network!";

        try {
            FileOutputStream fout = new FileOutputStream("d:\\test.txt");

            fout.write(text.getBytes());
            fout.close();
        }catch(IOException e) {
            System.out.println(e.toString());
        }
    }

}
```



버퍼화한 스트림-BufferedOutputSream/InputStream

- BufferedOutputStream
  - 데이터를 버퍼화 하여 저장하는 buf라는 필드를 포함하고 있다.
  - buf는 protected로 선언된 바이트 배열
  - 버퍼가 가득 차거나 스트림이 비워지기 까지 작업은 계속됨
- BufferedInputStream
  - protected로 선언된 buf라는 바이트 배열이 버퍼기능 지원
  - 버퍼에 데이터가 없을 경우에만 소스로 부터 데이터를 읽어 드림
  - 퍼포먼스 향상
- 생성자
  - public BufferedInputStream(InputStream in)
  - public BufferedInputStream(InputStream in, int bufferSize)
  - public BufferedOutputStream(OutputStream out)
  - public BufferedOutputStream(OutputStream out, int bufferSize)

예제) 버퍼스트림 테스트(입력 후 종료-Ctrl+z)

BufferedStreamCopier.java

```java
import java.io.*;

public class BufferedStreamCopier {

    public static void main(String[] args) {
        try {
            copy(System.in, System.out);
        }catch(IOException e) {
            System.err.println(e);
        }

    }

    public static void copy(InputStream in, OutputStream out) throws IOException {
        synchronized(in){
            synchronized(out){
                BufferedInputStream bin = new BufferedInputStream(in);
                BufferedOutputStream bout = new BufferedOutputStream(out);
                    while(true) {
                        int data = bin.read();
                        if (data == -1) break;
                        bout.write(data);
                    }
                    bout.flush();
            }
        }
    }
}

```

Problems @ Javadoc Console Progress

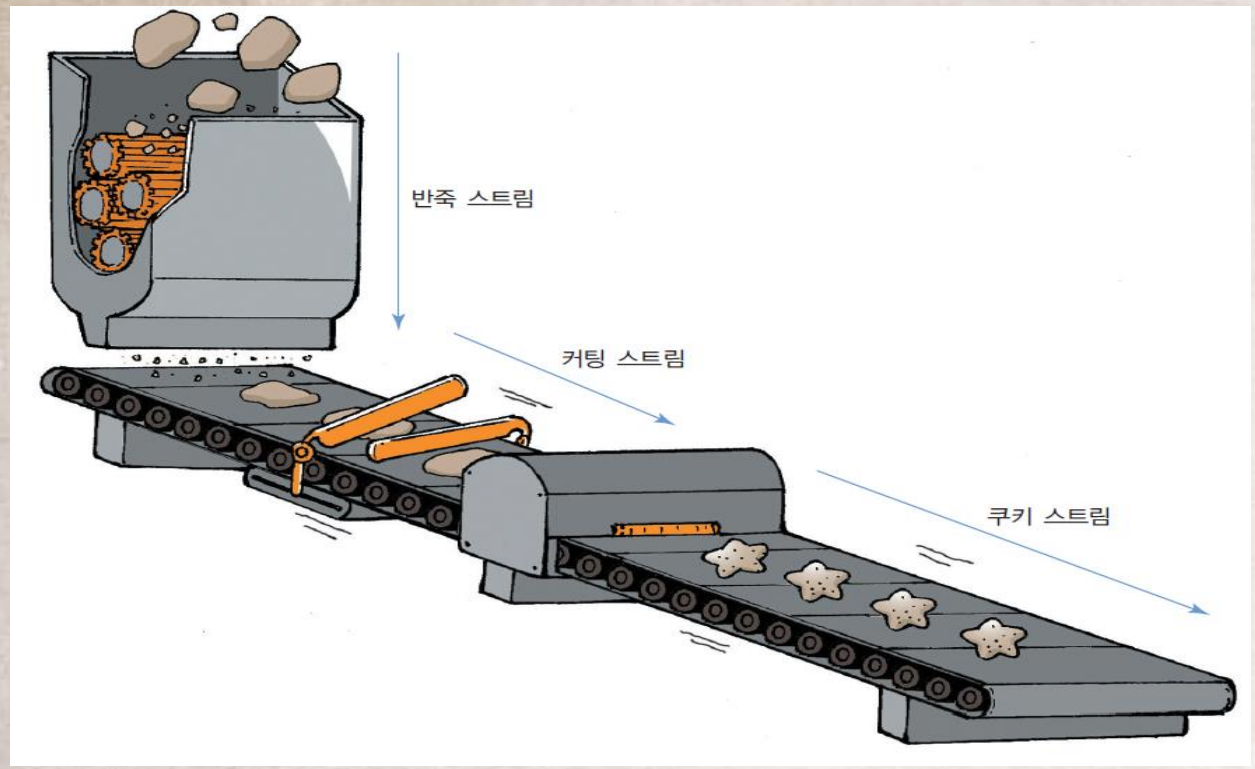
<terminated> BufferedStreamCopier [Java Application] C:₩Program Files₩Java₩jre1.8.0_171₩bin₩javaw.exe (2018. 11. 18. 오후 12:55:42)

```
Hello Word
Java network programming
Hello Word
Java network programming
```

필터스트림

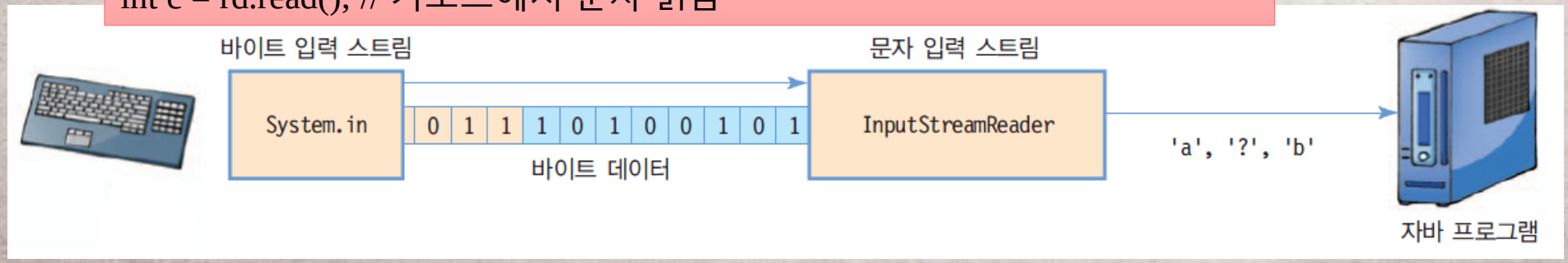
- 입력스트림이나 출력스트림 어느쪽에도 연결 가능
- 데이터의 read와 write과정에서 데이터 수정에 사용됨
- 데이터를 다른 포맷으로 변경하기 위해서 읽고 쓰는 데 필요한 부가적인 메소드를 필터가 제공
  - java.io.DataOutputStream은 정수형 숫자를 read하여 4 바이트로 변형하여 출력스트림에 해당 바이트를 써 넣을 수 있음(즉, 정수값을 출력할 수 있음)
    - public final void writeInt(int i) throws IOException
  - java.io.DataInputStream
    - public final int readInt() throws IOException

스트림은 연결될 수 있다(Filter 를 통과하는 Data)



* 표준 입력 스트림 System.in에 InputStreamReader 스트림을 연결하는 사례

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



문자 입출력 스트림

바이트와 문자 입출력 스트림 클래스 비교

바이트 스트림	대응 문자스트림	기능
InputStream	Reader	기본 입력 스트림 클래스
FileInputStream	FileReader	파일 입력 스트림 클래스
FilterInputStream	FilterReader	입력필터 스트림 클래스들의 최상위 클래스
BufferedInputStream	BufferedReader	기본 스트림에 버퍼를 추가한다.
DataInputStream	없음	기본 자료형 데이터를 읽는다.
OutputStream	Writer	기본 출력 스트림 클래스
FileOutputStream	FileWriter	파일 출력 스트림 클래스
FilterOutputStream	FilterWriter	출력필터 스트림 클래스들의 최상위 클래스
BufferedOutputStream	BufferedWriter	기본 스트림에 버퍼를 추가한다.
DataOutputStream	없음	기본 자료형에 버퍼를 추가한다.
PrintStream	PrintWriter	표준 시스템(화면)에 출력한다.

Writer/Reader 클래스

Writer	
void close() throws IOException	연결된 채널을 닫는다.
void flush() throws IOException	버퍼에 남은 모든 데이터를 전송한다.
abstract void write(int c) throws IOException	정수 c의 하위 16bit를 전송한다(자바에서 문자는 디폴트로 16비트 유니코드).
void write(char text[]) throws IOException	text의 내용을 전송한다.
void write(String s) throws IOException	주어진 문자열 s를 전송한다.
void write(String s, int index, int size) throws IOException	주어진 문자열 s의 index위치부터 size만큼의 바이트를 출력한다.

Reader	
int read() throws IOException	스트림으로부터 읽은 바이트를 2바이트씩 묶어서 하나의 문자를 하위 16비트로 반환한다.
int read(char buff[]) throws IOException	최대 buff 배열 길이만큼의 문자를 읽어서 buff문자 배열에 저장하고 읽은 수를 반환한다.
int read(char buff[], int index, int length) throws IOException	최대 length만큼 문자를 읽어서 buff 배열의 index위치부터 저장하고 읽은 수를 반환한다.
long skip(long n) throws IOException	n만큼 문자들을 건너뛰고 성공적으로 뛰어넘은 문자의 개수를 반환한다.
boolean ready() throws IOException	문자 입력 스트림이 준비되었는지 리턴한다.
void close() throws IOException	문자 입력 스트림을 해제한다.

FileWriter 및 FileReader 클래스

- FileOutputStream/FileInputStream과 대응되는 문자를 입출력 클래스
  - 기본 인코딩과 버퍼의 크기는 시스템에서 사용하는 디폴트(유니코드) 사용

BufferedStreamCopier.java | *WriteCharacter.java

```
1  import java.io.FileWriter;
2  import java.io.IOException;
3
4  public class WriteCharacter {
5
6      public static void main(String[] args) throws IOException{
7          String text = "한글 문서 파일입니다.";
8
9          FileWriter fw = new FileWriter("example.txt");
10         fw.write(text, 0, text.length());
11         fw.close();
12     }
13
14 }
15
```

example.txt *

```
1  한글 문서 파일입니다.
2
```

FileReader 클래스

BufferedStreamCopier.java | WriteCharacter.java | ReadCharacter.java

```
1  import java.io.FileReader;
2  import java.io.IOException;
3
4  public class ReadCharacter {
5
6      public static void main(String[] args) throws IOException {
7          int numberRead;
8          char[] buffer = new char[80];
9
10         FileReader fr = new FileReader("example.txt");
11         while((numberRead = fr.read(buffer)) > -1) {
12             System.out.println(buffer);
13         }
14         fr.close();
15     }
16
17 }
18
```

Problems | Javadoc | Console | Progress

<terminated> ReadCharacter [Java Application] C:₩Program Files₩Java₩jre1.8.0_171₩bin₩javaw.exe (2018. 11. 18. 오후 3:01:40)

한글 문서 파일입니다.

OutputStreamWriter 및 InputStreamReader

- 문자데이터를 바이트 기반의 입출력 스트림으로 전송
  - OutputStreamWriter: 문자 입력 스트림을 지정된 인코딩 바이트 스트림으로 변환시켜서 전송
  - InputStreamReader: 바이트 스트림으로 읽은 바이트 데이터를 지정된 인코딩 방식을 사용하여 문자로 변환

BufferedStreamCopier.java | WriteCharacter.java | ReadCharacter.java | WriteCharacter5601.java

```java
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.OutputStreamWriter;

public class WriteCharacter5601 {

    public static void main(String[] args) throws IOException {
        String text = "한글 문서 파일입니다.";

        FileOutputStream fos = new FileOutputStream("example5601.txt");
        OutputStreamWriter osw = new OutputStreamWriter(fos,"KSC5601");
        osw.write(text,0,text.length());
        osw.flush();
        osw.close();
    }
}
```

FileWriter/FileReader는 유니코드(16비트) 사용

example.txt *

```
1 한글 문서 파일입니다.
2
```

BufferedStreamCopier.java | WriteCharacter.java | ReadCharacter.java | WriteCharacter5601.java | ReadCharacter5601.java

```java
import java.io.*;

public class ReadCharacter5601 {

    public static void main(String[] args) throws IOException {
        int byteRead;
        char[] buffer = new char[128];

        FileInputStream fis = new FileInputStream("example5601.txt");
        InputStreamReader isr = new InputStreamReader(fis, "KSC5601");
        while((byteRead = isr.read(buffer, 0, buffer.length)) != -1) {
            System.out.println(buffer);
        }
    }

}
```

Problems | Javadoc | Console | Progress

<terminated> ReadCharacter5601 [Java Application] C:₩Program Files₩Java₩jre1.8.0_171₩bin₩javaw.exe (2018. 11. 18. 오후 3:56:56)

```
한글 문서 파일입니다.
```

TCP/IP

TCP/IP 소개

- TCP 프로토콜
  - TCP는 Transmission Control Protocol
  - 두 시스템 간에 신뢰성 있는 데이터의 전송을 관장하는 프로토콜
    - 헤더 체크섬은 TCP Header에 있는 손상을 감지
    - TCP는 각 패킷이 송신된 순서대로 수신되도록 책임을 짐
  - TCP에서 동작하는 응용프로그램 사례
    - e-mail, FTP, 웹(HTTP) 등

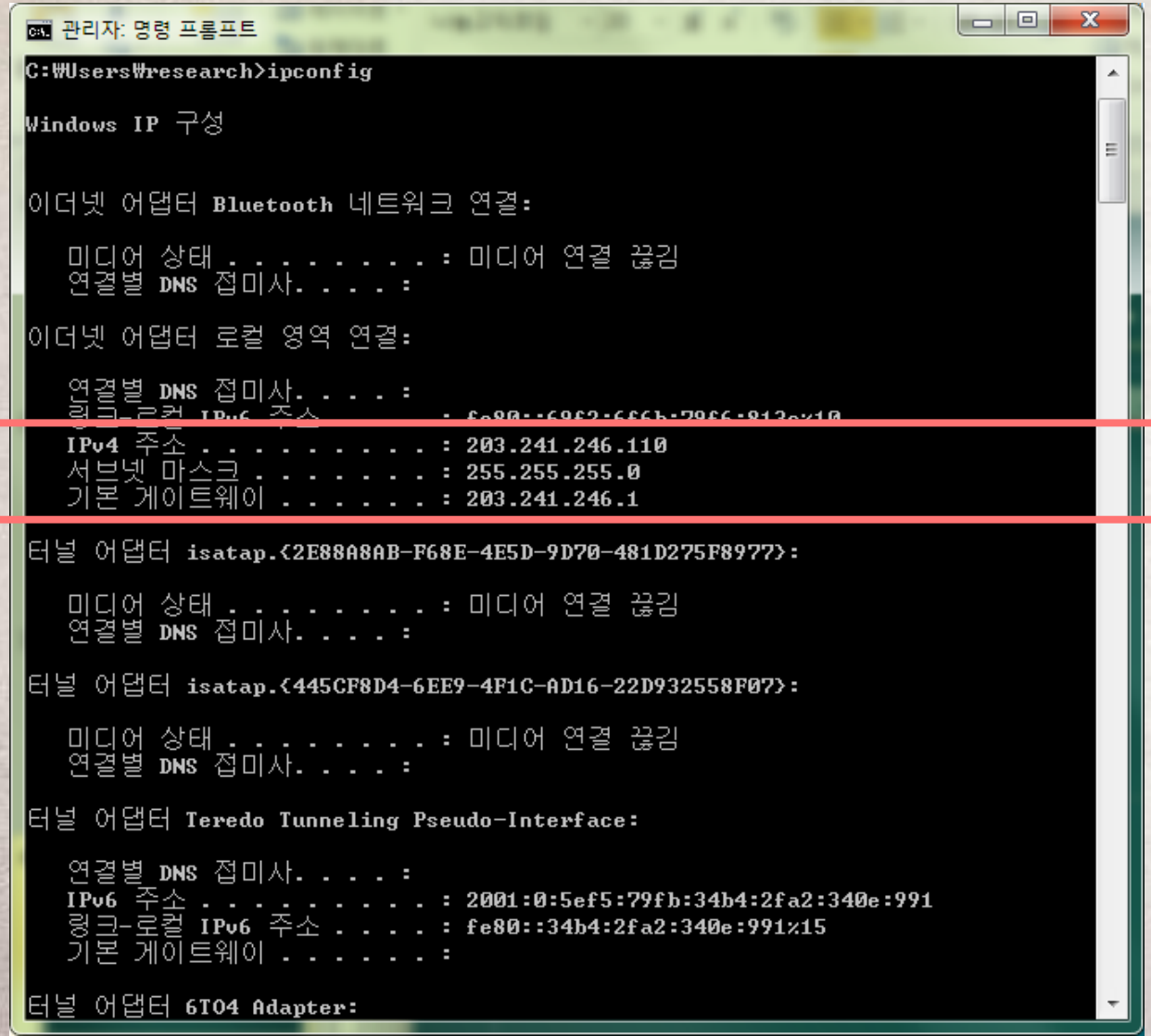
- IP
  - Internet Protocol
  - 통신 노드 간의 IP패킷을 전송하는 기능과 라우팅 기능을 담당하는 프로토콜
  - TCP보다 하위 레벨 프로토콜

IP 주소

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

내 컴퓨터의 IP 주소 확인하기

- 내 컴퓨터의 윈도우에서 명령창을 열어 ipconfig 명령 수행

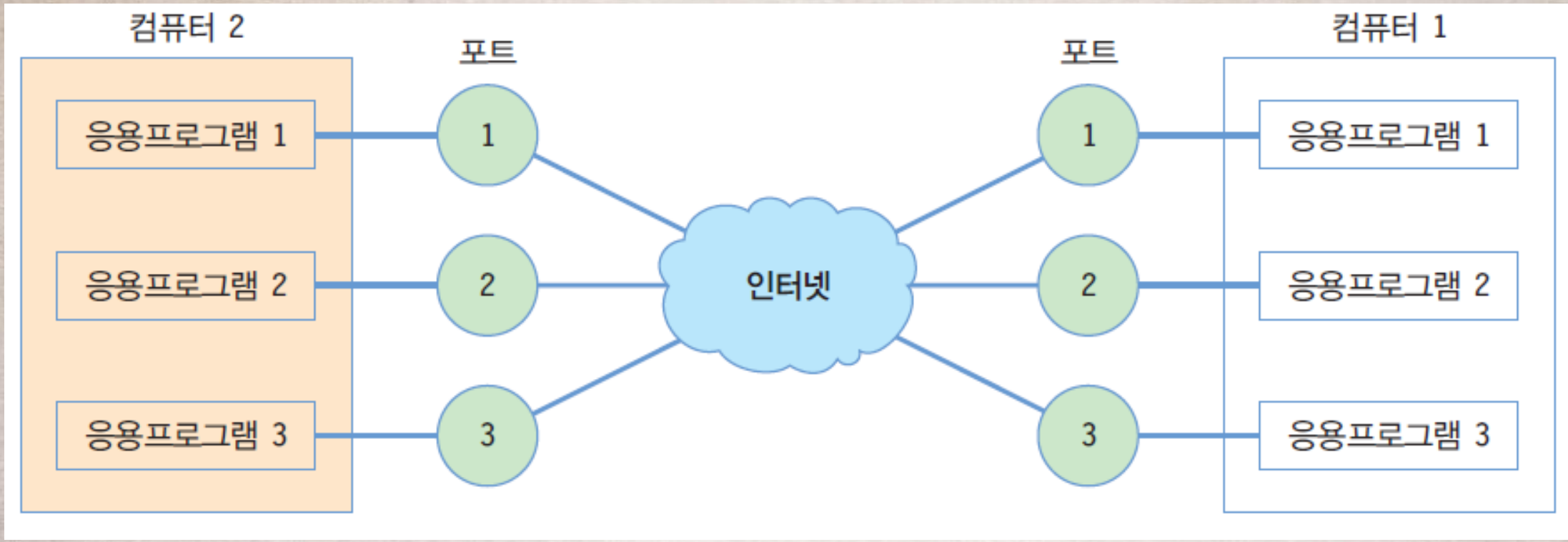


포트

- 포트
  - 통신하는 프로그램 간에 가상의 연결단
    - IP 주소는 네트워크 상의 컴퓨터 또는 시스템을 식별하는 주소
    - 포트 번호를 이용하여 통신할 응용프로그램 식별
  - 모든 응용프로그램은 하나 이상의 포트 생성 기
    - 포트를 이용하여 상대방 응용프로그램과 데이터
  - 잘 알려진 포트(well-know ports)
    - 시스템이 사용하는 포트 번호
    - 잘 알려진 응용프로그램에서 사용하는 포트 번
      - 0부터 1023 사이의 포트 번호

      ex) 텔넷 23, HTTP 80, FTP 21
    - 잘 알려진 포트 번호는 개발자가 사용하지 않는 좋음
      - 충돌 가능성 있음

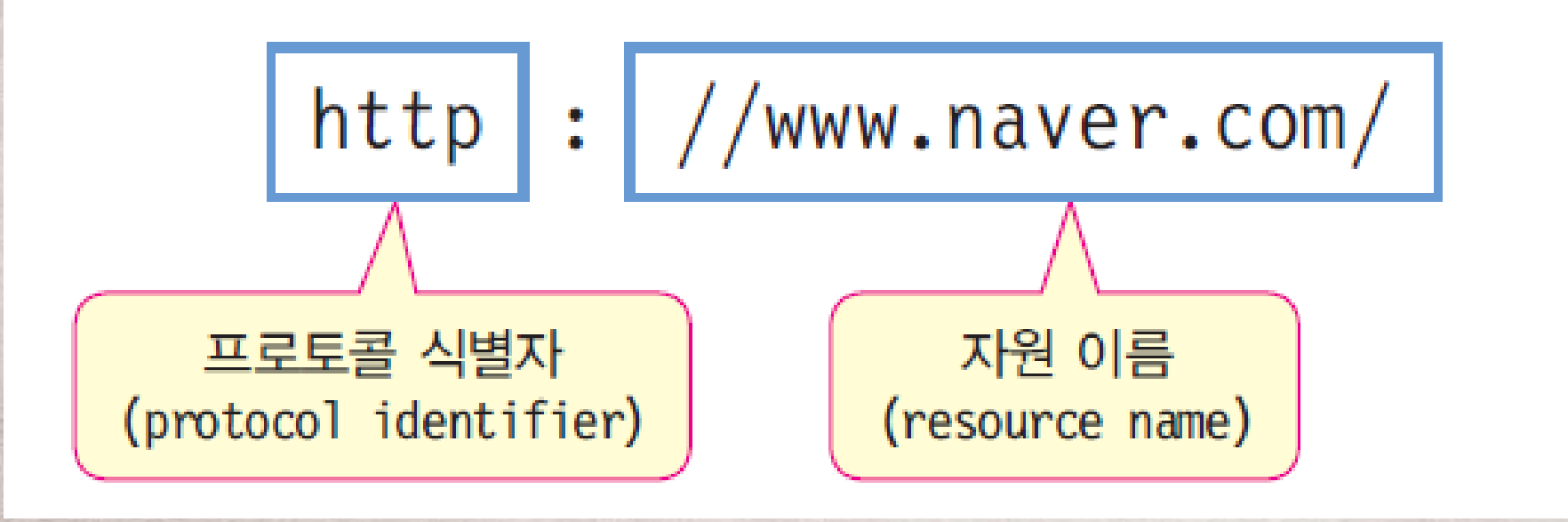
포트를 이용한 통신


컴퓨터 2
포트
포트
컴퓨터 1
응용프로그램 1
응용프로그램 2
응용프로그램 3
1
2
3
인터넷
1
2
3
응용프로그램 1
응용프로그램 2
응용프로그램 3

URL을 이용한 웹 프로그래밍

- URL이란?
  - URL은 Uniform Resource Locator
  - 인터넷 상의 리소스에 대한 주소(HTML 문서, 이미지, ...)

- URL 구조

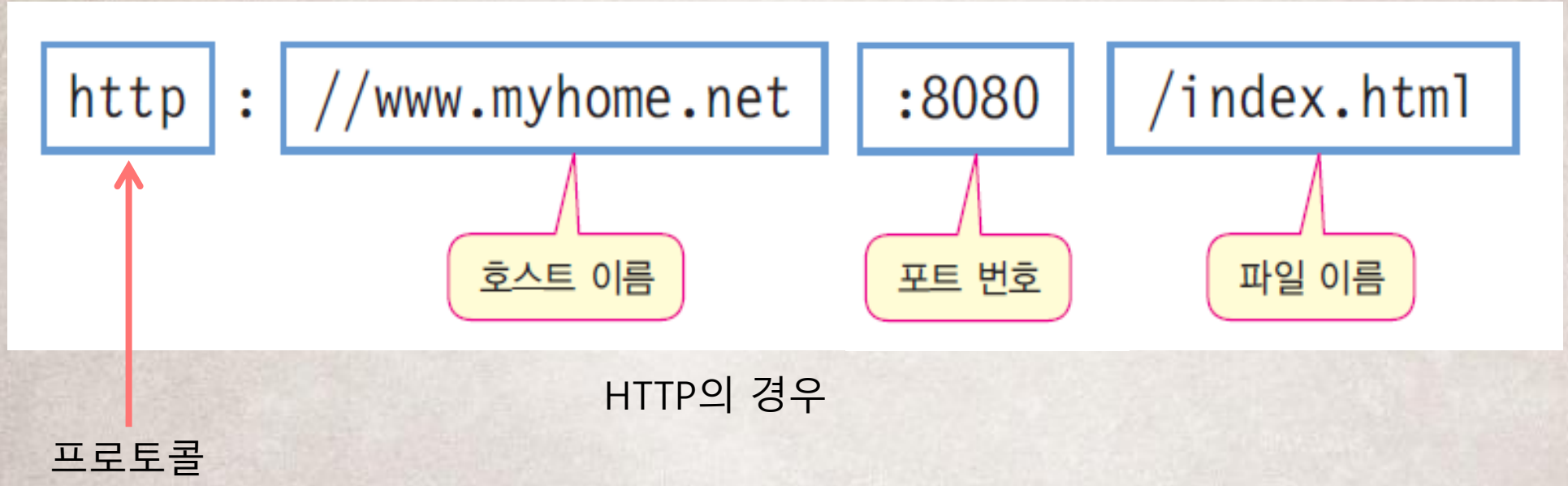


프로토콜 식별자

- 프로토콜 식별자
  - 인터넷상의 자원을 가져
  - 종류
    - HTTP, FTP, TELNET
  - 대부분의 브라우저들은 HTTP

자원 이름

- 자원 이름
  - 자원 이름은 사용되는 프로토콜에 따라서 그 구성이 달라짐



HTTP의 경우

자바의 URL 클래스

- URL 클래스(URL상의 리소스에 대한 객체)
  - java.net 패키지에 포함
  - 웹 상의 자원을 지정하는 URL을 나타냄

생성자	설 명
URL(String spec)	문자열이 지정하는 자원에 대한 URL 객체를 생성
URL(String protocol, String host, int port, String file)	프로토콜 식별자, 호스트 주소, 포트 번호, 파일 이름이 지정하는 자원에 대한 URL 객체 생성
URL(String protocol, String host, String file)	프로토콜 식별자, 호스트 주소, 파일 이름이 지정하는 자원에 대한 URL 객체 생성
URL(URL context, String spec)	URL context객체 내에 spec에서 지정하는 자원에 대한 URL 객체 생성

자바의 URL 클래스의 주요 메소드

- 주요 메소드

메소드	설 명
Object getContent()	URL의 컨텐트를 반환
String getFile()	URL 주소의 파일 이름 반환
String getHost()	URL 주소의 호스트 이름 반환
String getPath()	URL 주소의 경로 부분 반환
int getPort()	URL 주소의 포트 번호 반환
int getLocalPort()	소켓이 연결된 로컬 포트 번호 반환
int getPort()	소켓이 연결한 서버의 포트 번호 반환
InputStream openStream()	URL에 대해 연결을 설정하고 이 연결로부터 입력을 받을 수 있는 InputStream 객체 반환
URLConnection openConnection()	URL 주소의 원격 객체에 접속한 뒤 통신할 수 있는 URLConnection 객체 리턴

예제 : URL 파싱하기

- URL 클래스를 이용하여 URL을 구성하는 프로토콜 이름, 호스트 주소, 포

BufferedIOEx.java | FileClassExample.java | ParseURL.java

```java
import java.net.MalformedURLException;
import java.net.URL;


public class ParseURL {

    public static void main(String[] args) {
        URL opinion = null;
        URL homepage = null;

        try{
            homepage = new URL("http://news.hankooki.com:80");
            opinion = new URL(homepage, "opionion/editorial.html");
        }catch(MalformedURLException e){
            System.out.println("잘못된 URL입니다.");
        }

        System.out.println("protocol = "+opinion.getProtocol());
        System.out.println("host = "+opinion.getHost());
        System.out.println("port = "+opinion.getPort());
        System.out.println("path = "+opinion.getPath());
        System.out.println("filename = "+opinion.getFile());

    }

}
```

```
protocol = http
host = news.hankooki.com
port = 80
path = /opionion/editorial.html
filename = /opionion/editorial.html
```

URLConnection 클래스

- URLConnection 클래스
  - **주어진 원격지의 주소 URL에 네트워크 접속 후 데이터를 보내거나 받을 수 있도록 하는 기능**
  - URL 객체 생성 방법
    - URL.openConnection() 이용

```
URL aURL = new URL("http://www.naver.com");
URLConnection uc = aURL.openConnection(); // 원격지와 연결한다.
```

    - URLConnection 생성자 이용

```
URL aURL = new URL("http://www.naver.com");
URLConnection uc = new URLConnection(aURL);
uc.connect(); // 원격지와 연결한다.
```

      - 연결하기 전에 여러 가지 인자들과 요청과 관련된 속성들을 설정 가능

URLConnection 클래스 주요 메소드

메소드	설명
abstract void connect()	URL에 의해 참조되는 외부 리소스와 통신 연결 설정
Object getContent()	URL 연결에서 컨텐트를 가져옴
String getContentEncoding()	컨텐트 인코딩 필드를 반환
int getContentLength()	컨텐트 길이 필드 반환
String getContentType()	컨텐트 타입 필드 반환
boolean getDoInput()	URLConnection 객체의 doInput 필드 값 반환
boolean getDoOutput()	URLConnection 객체의 doOutput 필드 값 반환
InputStream getInputStream()	설정된 연결에서 데이터를 읽을 입력 스트림 반환
OutputStream getOutputStream()	설정된 연결로 데이터를 출력할 출력 스트림 반환
URL getURL()	URLConnection 객체의 URL 필드 값 반환
void setDoInput(boolean doInput)	URLConnection 객체의 doInput 필드 값 설정
void setDoOutput(boolean doOutput)	URLConnection 객체의 doOutput 필드 값 설정

- doInput 필드가 true로 설정되면 URLConnection 객체로 표현되는 URL 연결이 입력을 위해 사용됨을 의미.
- doOutput 필드가 true로 설정되면 출력을 위해 사용됨을 의미

URLConnection 객체를 이용하여 원격지 데이터 받기

- URLConnection 객체에서 데이터 읽기
  - URLConnection 객체에서 getInputStream() 메소드를 이용하여 입력 스트림을 얻은 후에 스트림 입력을 수행

예제 7-2 : URLConnection으로 원격지에서 데이터 읽기

- URLConnection 객체를 이용하여 www.daum.net에 연결하여 데이터를 읽고 화면에 출력하는 프로그램을 작성하라.

*URLConnectionReader.java

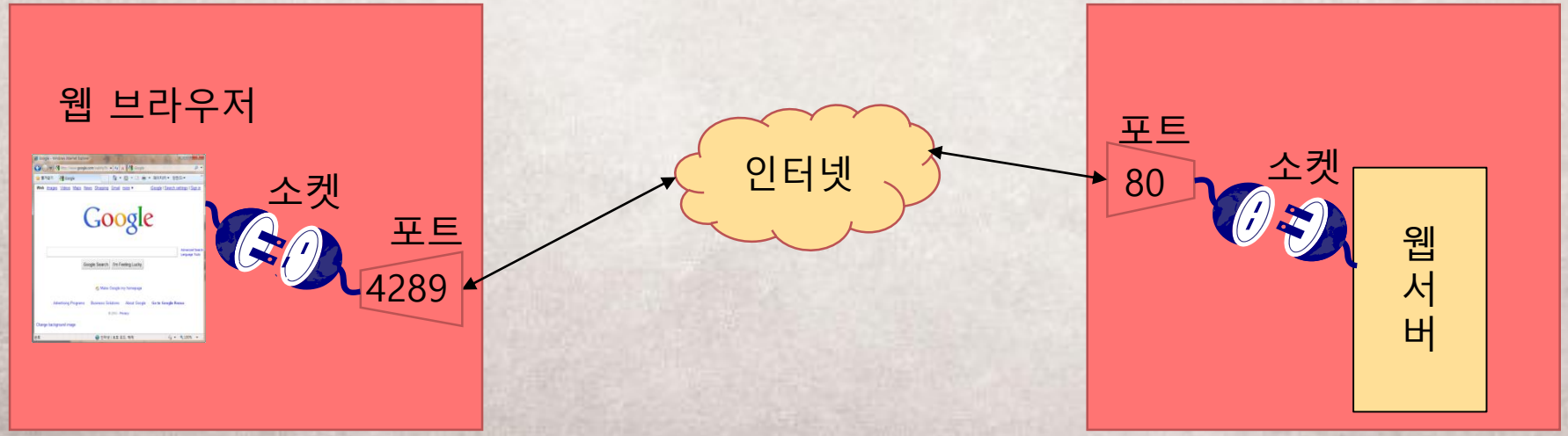
```
1  import java.io.BufferedReader;
2  import java.io.IOException;
3  import java.io.InputS
4  import java.net.URL;
5  import java.net.URLCo
6
7
8  public class URLConne
9
10     public static voi
11         try{
12             URL url =
13
14             URLConnec
15             BufferedR
16             String in
17
18             while((in
19                 Syste
20
21             }
22             in.close(
23         }catch(IOExce
24             System.ou
25         }
26
27     }
28
29 }
30
```

<terminated> URLConnectionReader [Java Application] C:₩Program Files₩Java₩jre1.8.0_31₩bin₩javaw.exe (2015. 4. 14. 오후 1:58:42)

```
.vertical_song .link_more {position:absolute;top:18px;right:11px;padding-right:7px;font-size:11px;line-height:14px;bacl
.tiles_story .vertical_reader {display:inline;float:right;padding:0 1px;background-color:#e1e1e1}
.vertical_reader .wrap_bestreader {overflow:hidden;position:relative;width:188px;height:339px;border-top:1px solid #e1e
.vertical_reader .tit_bestreader {display:block;overflow:hidden;width:174px;height:29px;padding:18px 0 0 14px;line-heig
.vertical_reader .list_thumb {overflow:hidden;width:188px;padding-top:15px;border-top:1px solid #ececec}
.vertical_reader .list_thumb li {float:none;margin:0;padding:0 15px 20px;font-size:0}
.vertical_reader .list_thumb .link_cont {width:158px;height:50px;cursor:pointer}
.vertical_reader .list_thumb .info_thumb {float:left;margin-right:10px}
.vertical_reader .list_thumb .info_cont {float:none}
.vertical_reader .info_thumb .img_daum {overflow:hidden;position:absolute;top:0;left:0;z-index:1;width:15px;height:15px
.vertical_reader .info_thumb .num1 {background-position:-290px -55px}
.vertical_reader .info_thumb .num2 {background-position:-310px -55px}
.vertical_reader .info_thumb .num3 {background-position:-330px -55px}
.vertical_reader .info_thumb .num4 {background-position:-350px -55px}
.vertical_reader .frame_g {width:58px;height:48px}
.vertical_reader .tit_story {display:block;overflow:hidden;float:left;width:88px;height:34px;margin-top:-1px;font-size
.vertical_reader .readers_info {display:block;font-size:11px;line-height:14px;color:#878787}
.vertical_reader .readers_info .ico_daum {float:left;width:15px;height:13px;margin:4px 2px 0 0;text-indent:-9999px}
.vertical_reader .readers_info .ico_reader {background-position:-80px -250px}
.vertical_reader .readers_info .num_reader {overflow:hidden;float:left;width:71px;margin-top:4px;white-space:nowrap;te
.vertical_reader .link_more {position:absolute;top:18px;right:11px;padding-right:7px;font-size:11px;line-height:14px;ba
.layout_headline_vertical .vertical_img {float:left}
.layout_vertical .default_img .box_tit {height:43px;padding-top:12px}
.layout_promotion .default_img .box_tit {height:43px;padding-top:12px}
.wrap_cafe .layout_promotion .default_img .box_tit {height:45px;padding-top:10px}
.wrap_storyball .ico_newarr {background-position:0 -35px}
.wrap_storyball .ico_recom {background-position:-55px -35px}
```

소켓 프로그래밍

- 소켓 (socket)
  - 소켓은 네트워크 상에서 수행되는 두 프로그램 간의 양방향 통신 링크의 한 쪽 끝 단을 의미
  - 소켓은 특정 포트 번호와 연결되어 있음
    - TCP에서 데이터를 보낼 응용프로그램을 식별할 수 있음.
  - 자바에서의 데이터 통신 시 소켓 사용
  - 소켓 종류
    - 서버 소켓과 클라이언트 소켓



소켓 주소

- IP는 호스트와의 통신을 확인하기 위해 32비트 이진 주소 사용한다.
  - 자바에서 주소는 숫자형 주소(169.1.1.1)이나 이름(comsi.inje.ac.kr)같은 스트링을 사용하여 표현
- 클라이언트는 반드시 서버 프로그램이 통신을 시작할 때 작동되는 호스트의 IP 주소를 알아야 한다.
- InetAddress 클래스를 이용해 IP주소 확인
  - getByNames()나 getAllByNames()는 이름이나 IP주소를 가지고 이에 상응하는 InetAddress 반환
  - getLocalHost()는 지역 호스트 주소를 포함하는 InetAddress인스턴스 반환

예제7-3.소켓 주소 알아보기

InetAddressEx.java

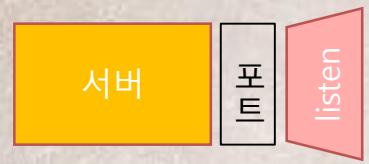
```java
import java.net.*;

public class InetAddressEx {

    public static void main(String[] args) {
        try{
            InetAddress address = InetAddress.getLocalHost();
            System.out.println("Local Host:");
            System.out.println("\t"+address.getHostName());
            System.out.println("\t"+address.getHostAddress());
        }catch(UnknownHostException e){
            System.out.println("Unable to determine this host's address");
        }

        for(int i = 0; i < args.length; i++){
            try{
                InetAddress[] addressList = InetAddress.getAllByName(args[i]);
                System.out.println(args[i] + ":");
                System.out.println("\t"+addressList[0].getHostName());
                for(int j=0; j<addressList.length; j++){
                    System.out.println("\t"+addressList[j].getHostAddress());
                }
            }catch(UnknownHostException e){
                System.out.println("Unable to find address for "+args[i]);
            }
        }
    }
}
```

```
<terminated> PointVectorEx [Java Application] C:₩Program Files₩Java₩j
Local Host:
        tae-pc
        192.168.0.8
comsi.inje.ac.kr:
        comsi.inje.ac.kr
        211.220.195.180
```

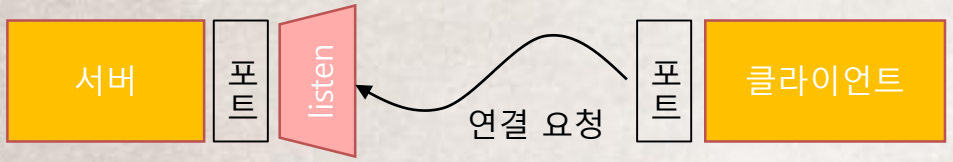
소켓을 이용한
서버 클라이언트 통신 프로그램

클라이언트와 서버 연결 순서

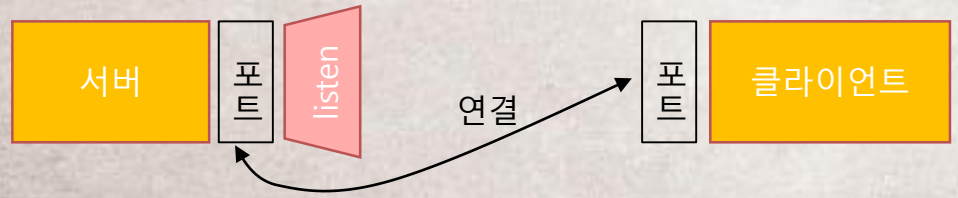
- 클라이언트와 서버 연결
  - 서버는 서버 소켓으로 들어오는 연결 요청을 기다림



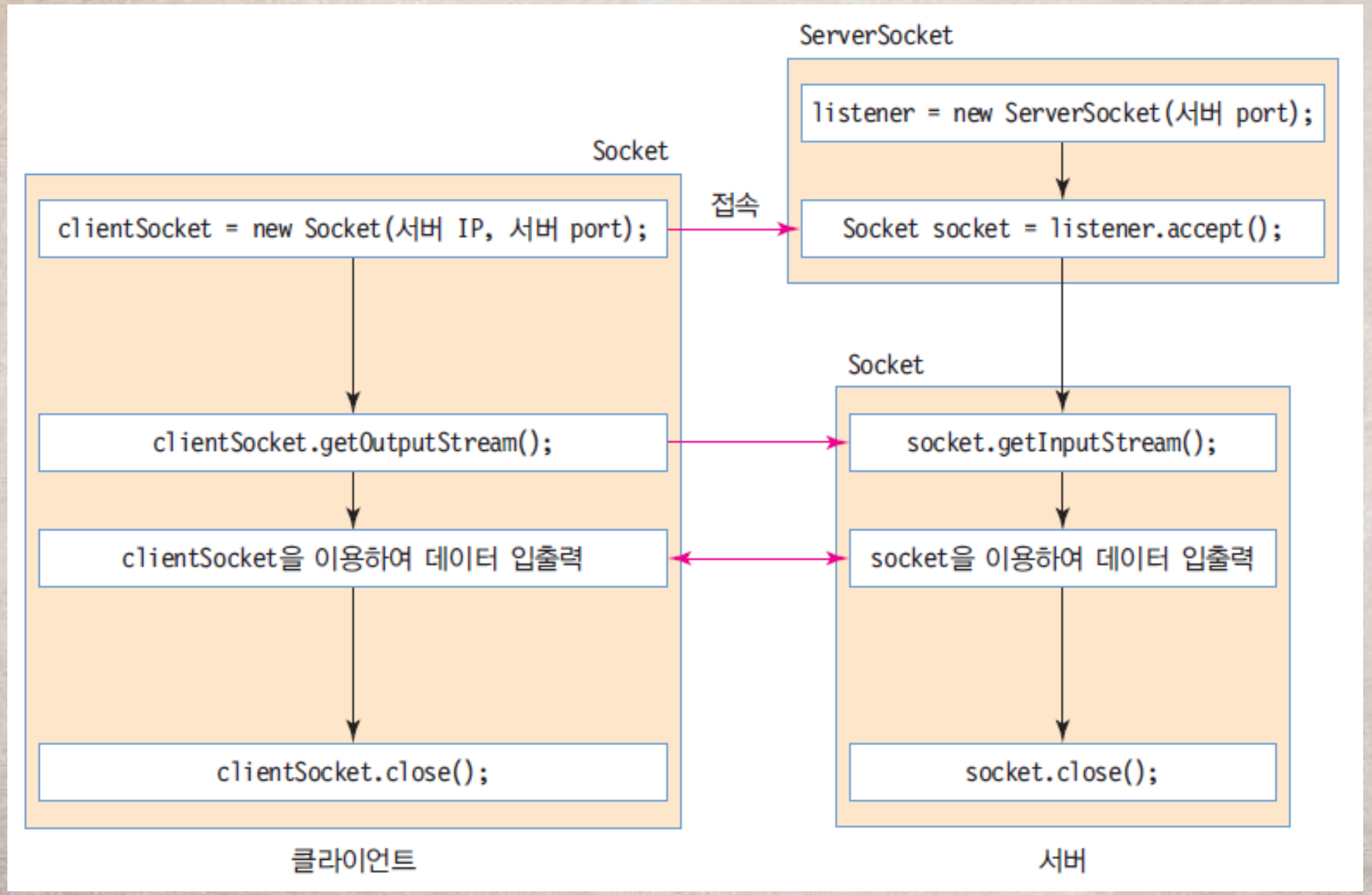
  - 클라이언트가 서버에게 연결 요청



  - 서버가 연결 요청 수락하고 새로운 소켓을 만들어 클라이언트와 연결 생성



소켓을 이용한 서버 클라이언트 통신 프로그램의 구조



Socket 클래스, 클라이언트 소켓

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

생성자	설명
Socket(InetAddress address, int port)	소켓을 생성하여 지정된 IP 주소와 포트 번호에 연결한다.
Socket(String host, int port)	소켓을 생성하여 지정된 호스트와 포트 번호에 연결한다. 호스트 이름이 null인 경우는 루프백(loopback) 주소로 가정한다.

루프백주소

- 컴퓨터의 네트워크 입출력 기능을 시험하기 위하여 가상으로 할당한 인터넷 주소(127.0.0.1)
- 웹 서버나 인터넷 소프트웨어의 네트워크 동작 기능을 시험하는 데 사용.

클라이언트 소켓 생성(서버 접속, 입출력 스트림 생성)

- 클라이언트 소켓 생성 및 서버에 접속
  - Socket 객체의 생성되면 곧 바로 128.12.1.1의 5550번 포트로 서버 접속 시도

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

- 네트워크 입출력 스트림 생성

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

  - 일반 스트림을 입출력 하는 방식과 동일
- 서버로 데이터 전송
  - Write()는 버퍼에 데이터를 저장
  - flush()를 호출하면 스트림 속(버퍼)에 데이터를 남기지 않고 모두 전송

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

- 서버로부터 데이터 수신

```
int x = in.read();          // 서버로부터 한 개의 문자 수신
String line = in.readline(); //서버로부터 한 행의 문자열 수신
```

- 네트워크 접속 종료

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

ServerSocket 클래스, 서버 소켓

- ServerSocket 클래스
  - 서버 소켓에 사용되는 클래스
  - java.net 패키지에 포함
  - 주요 생성자

생성자	설명
ServerSocket(int port)	소켓을 생성하여 지정된 포트 번호에 연결한다.

메소드	설명
Socket accept()	연결 요청을 기다리다 연결 요청이 들어오면 수락하고 새 Socket 객체를 반환
void close()	서버 소켓을 닫는다.
InetAddress getInetAddress()	서버 소켓에 연결된 로컬 주소 반환
int getLocalPort()	서버 소켓이 연결 요청을 모니터링하는 클라이언트 포트 번호 반환
boolean isBound()	서버 소켓이 로컬 주소에 연결되어있으면 true 반환
boolean isClosed()	서버 소켓이 닫혀있으면 true 반환
void setSoTimeout(int timeout)	accept()에 대한 타임 아웃 시간 지정. 0이면 타임아웃이 해제.

서버 소켓 생성(클라이언트 접속, 입출력 스트림 생성)

- 서버 소켓 생성

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

  - 이미 사용 중인 포트 번호를 지정하면 오류가 발생
- 클라이언트로부터 접속 기다림

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

  - accept() 메소드는 연결 요청이 오면 새로운 Socket 객체 반환
  - 서버에서 클라이언트와의 데이터 통신은 새로 만들어진 Socket 객체를 통해서 이루어짐
  - ServerSocket 클래스는 Socket 클래스와 달리 주어진 연결에 대해 입출력 스트림을 만들어 주는 메소드가 없음
- 네트워크 입출력 스트림 생성

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

  - accept() 메소드에서 얻은 Socket 객체의 getInputStream()과 getOutputStream() 메소드를 이용하여 데이터 스트림 생성
  - 일반 스트림을 입출력하는 방식과 동일하게 네트워크 데이터 입출력

클라이언트로 데이터 송수신

- 클라이언트로부터 데이터 수신

```
int x = in.read();              // 클라이언트로부터 한 개의 문자 수신
String line = in.readline(); //클라이언트로부터 한 행의 문자열 수신
```

- 클라이언트로 데이터 전송
  - Write()는 버퍼에 저장
  - flush()를 호출하면 스트림 버퍼에 있는 모든 데이터를 전송

```
out.write("Hi!, Client"+"\n");
out.flush();
```

- 네트워크 접속 종료
- 서버 응용프로그램 종료

```
socket.close();
```

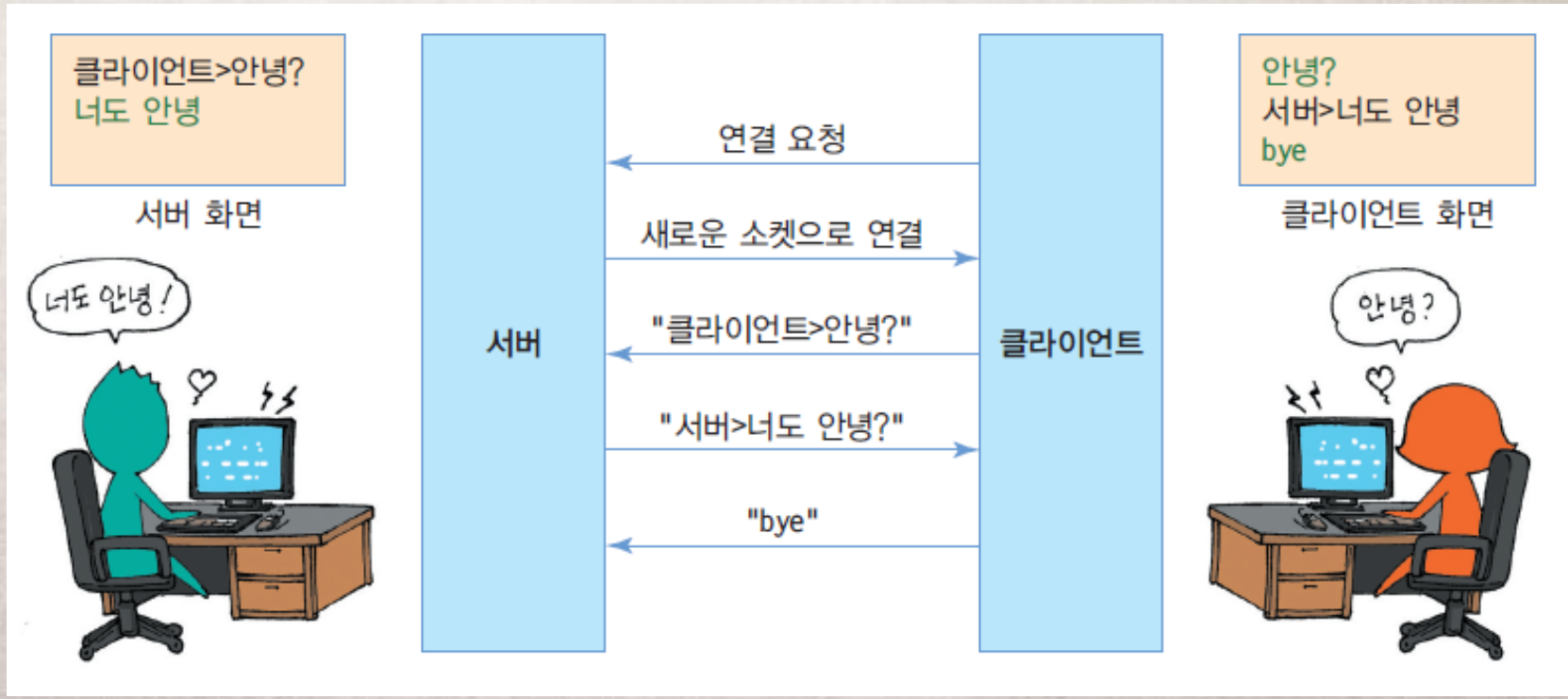
```
serverSocket.close();
```

  - 더 이상 클라이언트의 접속을 받지 않고 서버 응용 프로그램을 종료하고자 하는 경우 ServerSocket 종료

Practice – Chatting program

소켓을 이용한 클라이언트/서버 채팅 예제

- 간단한 채팅 프로그램 예제
  - 서버와 클라이언트가 1:1로 채팅 하는 간단한 예제
  - 서버와 클라이언트 간의 메시지 구분을 위해 서버는 메시지 앞에 "서버>"을 접두어로 붙여 메시지를 전송하며 클라이언트는 "클라이언트>"를 접두어로 붙여 메시지 전송
  - 서버와 클라이언트가 번갈아 가면서 메시지 전송 및 수신
  - 클라이언트가 bye를 보내면 프로그램 종료



서버 프로그램

- 서버 소켓 생성

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

  - 시스템에서 사용되지 않은 포트 번호로 서버 소켓 생성
- 클라이언트 요청 대기

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

  - 클라이언트가 연결 요청이 올 때까지 소켓 기다림
  - 해당 포트 번호로 연결 요청이 오면
    - 수락과 함께 새로운 소켓을 생성
    - 새 소켓으로 클라이언트와 통신
  - 새로운 소켓의 포트 번호는 자동으로 할당

서버 프로그램

- 클라이언트와 통신을 위한 입출력 스트림 생성

```
InputStream in = socket.getInputStream();
OutputStream = socket.getOutputStream();
```

  - 스트림을 생성하여 클라이언트와 데이터 송수신
    - 데이터의 종류에 따라 바이트 스트림 또는 문자 스트림을 생성
    - 채팅과 같이 문자열을 송수신하는 경우는 문자 스트림 사용
    - 효율적 입출력을 위하여 버퍼 스트림(Buffered Stream) 사용
- 클라이언트로부터 데이터 수신 및 송신

```
int ch;
wihle((ch = in.read()) != -1){
    System.out.print((char)ch);
   outStream.write(ch);
};
```

  - 스트림 생성 이후는 데이터 입력 받는 방법과 동일
  - 클라이언트에서 한 행의 문자열(-1)을 보내올 때까지 기다림
  - flush() 메소드로 스트림의 모든 데이터를 클라이언트로 송신가능

서버 프로그램

- 연결 종료

```
socket.close();
servSocket.close();
```

  - 데이터의 송수신이 끝나면 소켓을 닫아야 함
  - 소켓을 닫으면 소켓의 입출력 스트림도 같이 닫힘
  - 서버 소켓을 닫으면 클라이언트 연결 요청을 받을 수 없음

클라이언트 프로그램

- 연결 요청

```
socket = new Socket("localhost", 9999);
```

  - 소켓 생성
    - 서버의 호스트 주소
    - 서버가 연결 요청을 모니터링하는 포트 번호로 소켓 생성
  - 예제는 호스트 이름을 “localhost”로 지정
    - 동일한 시스템에서 서버와 클라이언트가 동작하기 때문
- 클라이언트와 통신을 위한 입출력 스트림 생성

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

  - 스트림을 생성하여 서버와 데이터 송수신
    - 데이터의 종류에 따라 바이트 스트림 또는 문자 스트림 사용
    - 채팅과 같이 문자열을 송수신하는 경우는 문자 스트림 사용
    - 효율적 입출력을 위하여 버퍼 스트림(Buffered Stream) 사용

클라이언트 프로그램

- 서버에 데이터 송신

```
outStream.write(“Hello World!”.getBytes());
//out.flush();
```

  - 스트림 생성 이후는 데이터 출력 방법과 동일
  - 콘솔에서 입력 받은 문자열을 서버로 송신
  - flush() 메소드로 스트림의 모든 데이터를 서버로 송신
- 클라이언트의 데이터 수신

```
while(receivedTotalByte < sendByteBuffer.length){
    if((bytesRcvd = inStream.read(sendByteBuffer, receivedTotalByte,
        sendByteBuffer.length - receivedTotalByte)) == -1) throw new
            SocketException("Connection close prematurely");
   receivedTotalByte += bytesRcvd;
}
```

  - 스트림 생성 이후는 데이터 입력 방법과 동일
  - 서버로 보낸 바이트 수만큼 받을 때까지 반복해서 바이트를 받는다.
  - read()함수의 파라메터-받을 버퍼, 처음으로 받은 바이트가 있어야 오프셋, 버퍼에 담을 수 있는 바이트의 최대값

클라이언트 프로그램

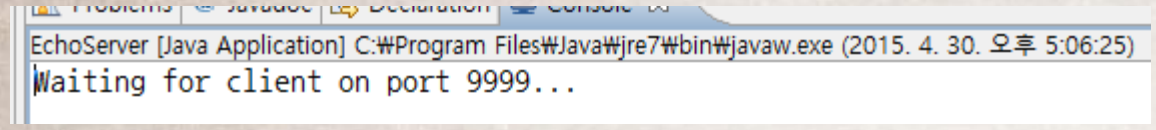
- 연결 종료

```
socket.close();
```

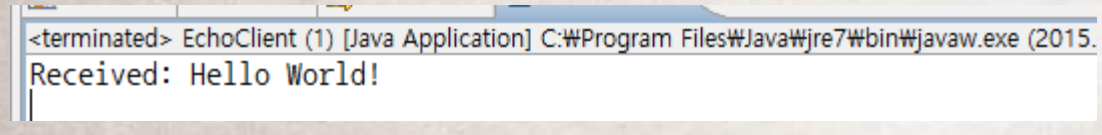
  - 데이터의 송수신이 끝나면 소켓을 닫아야 함
  - 소켓을 닫으면 소켓의 입출력 스트림도 같이 닫힘

실행 보기

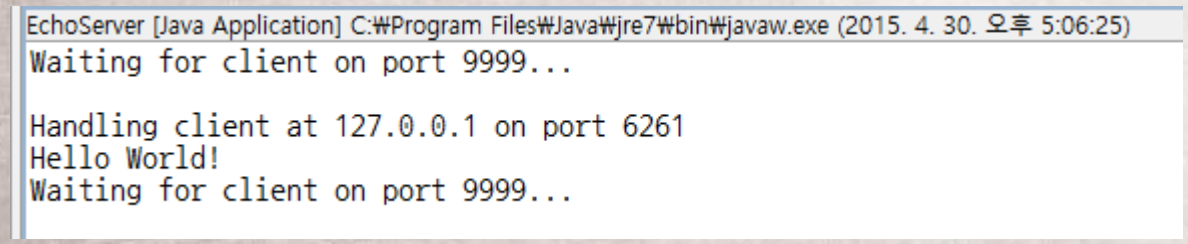
- 서버 시작



- 클라이언트가 서버로 전송 후 서버의 응답을 받음



- 클라이언트의 접속 후 서버의 상태



예제7-3. 서버 프로그램

EchoServer.java | EchoClient.java

```
1  import java.io.IOException;
2  import java.io.InputStream;
3  import java.io.OutputStream;
4  import java.net.ServerSocket;
5  import java.net.Socket;
6
7
8  public class EchoServer {
9
10     // 클라이언트의 요청을 받기 위한 서버소켓 포트
11     private static final int SERVER_SOCKET_PORT = 9999;
12     private static final int BUFFER_SIZE = 32;
13
14     public static void main(String[] args) {
15         ServerSocket  servSocket = null;
16         byte[] byteBuffer = new byte[BUFFER_SIZE];
17         try {
18             // 클라이언트의 컨넥션 요청을 받기 위한 서버소켓 생성
19             servSocket = new ServerSocket(SERVER_SOCKET_PORT);
20
21             for(;;){
22                 System.out.println("Waiting for client on port " + servSocket.getLocalPort() + "...\n");
23
24                 // 클라이언트의 소켓을 얻는다.
25                 Socket clientSocket = servSocket.accept();
26
27                 System.out.println("Handling client at " +
28                             clientSocket.getInetAddress().getHostAddress() + " on port " +
29                             clientSocket.getPort());
30                 InputStream inStream = clientSocket.getInputStream();
31                 OutputStream outStream = clientSocket.getOutputStream();
32
```

예제7-3. 서버 프로그램

```
32
33                  int ch;
34
35                  while((ch = inStream.read()) != -1){
36                      System.out.print((char)ch);
37                      outStream.write(ch);
38                  }
39
40                  System.out.println(new String(byteBuffer));
41
42                  clientSocket.close();
43              }
44
45          } catch (IOException e) {
46              System.out.println("스트림입출력 중에 오류가 발생했습니다.");
47              System.out.println(e.getMessage());
48
49          }
50      }
51
52  }
53
```

예제7-3. 클라이언트 프로그램

EchoServer.java | EchoClient.java

```
1  import java.io.IOException;
2  import java.io.InputStream;
3  import java.io.OutputStream;
4  import java.net.Socket;
5  import java.net.SocketException;
6  import java.net.UnknownHostException;
7
8
9  public class EchoClient {
10
11     // 클라이언트에 요청을 보내기 위한 서버소켓 포트
12     private static final int SERVER_SOCKET_PORT = 9999;
13     private static final String SERVER_ADDRESS = "localhost";
14
15
16     public static void main(String[] args) {
17
18         byte[] sendByteBuffer = "Hello World!".getBytes();
19         Socket socket = null;
20
21         try {
22             // 해당 서버(SERVER_ADDRESS)의 포트(SERVER_SOCKET_PORT)에 연결한다.
23             socket = new Socket(SERVER_ADDRESS, SERVER_SOCKET_PORT);
24
25             InputStream inStream = socket.getInputStream();
26             OutputStream outStream = socket.getOutputStream();
27
28             // 서버에 문자열을 보낸다.
29             outStream.write(sendByteBuffer);
30
```

예제7-3. 클라이언트 프로그램

```
31              // 서버로부터 돌려 받기 위한 처리
32              int receivedTotalByte = 0; // 전체 받은 바이트 수
33              int bytesRcvd; // 마지막에 read에서 받은 바이트 수
34              while(receivedTotalByte < sendByteBuffer.length){
35                  if((bytesRcvd = inStream.read(sendByteBuffer, receivedTotalByte,
36                          sendByteBuffer.length - receivedTotalByte)) == -1)
37                      throw new SocketException("Connection close prematurely");
38                  receivedTotalByte += bytesRcvd;
39              }
40
41              System.out.println("Received: " + new String(sendByteBuffer));
42
43              socket.close();
44          } catch (UnknownHostException e) {
45              System.out.println("정상적인 Host가 아닙니다.");
46              System.out.println(e.getMessage());
47          } catch (IOException e) {
48              System.out.println("스트림입출력 중에 오류가 발생했습니다.");
49              System.out.println(e.getMessage());
50
51          }
52
53      }
54
55  }
56
```