

Network

네트워크 프로그램

최 문 환

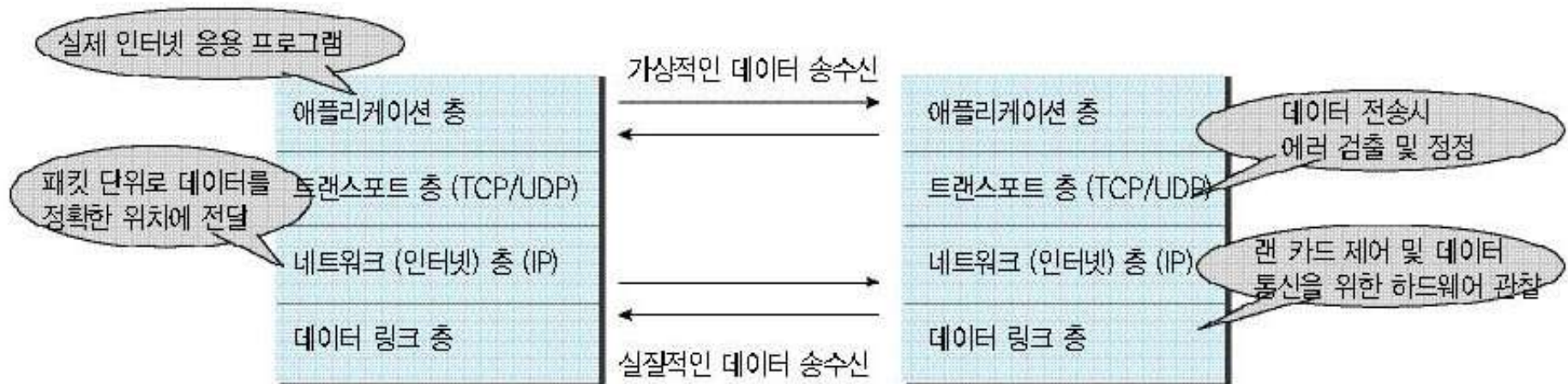


1. 네트워크에 대한 기본 개념 정리

● 통신 프로토콜

- 통신하고자 하는 두 컴퓨터 사이에 정해진 규약에 따라 접속을 하고 데이터를 주고 받도록 하기 위해서 미리 정해 놓은 규약

● TCP/IP



1. 네트워크에 대한 기본 개념 정리

- ◆ 소켓이란?

- TCP를 이용한 프로그램을 작성할 수 있도록 하기 위해서 제공되는 인터페이스

- ◆ 서버와 클라이언트

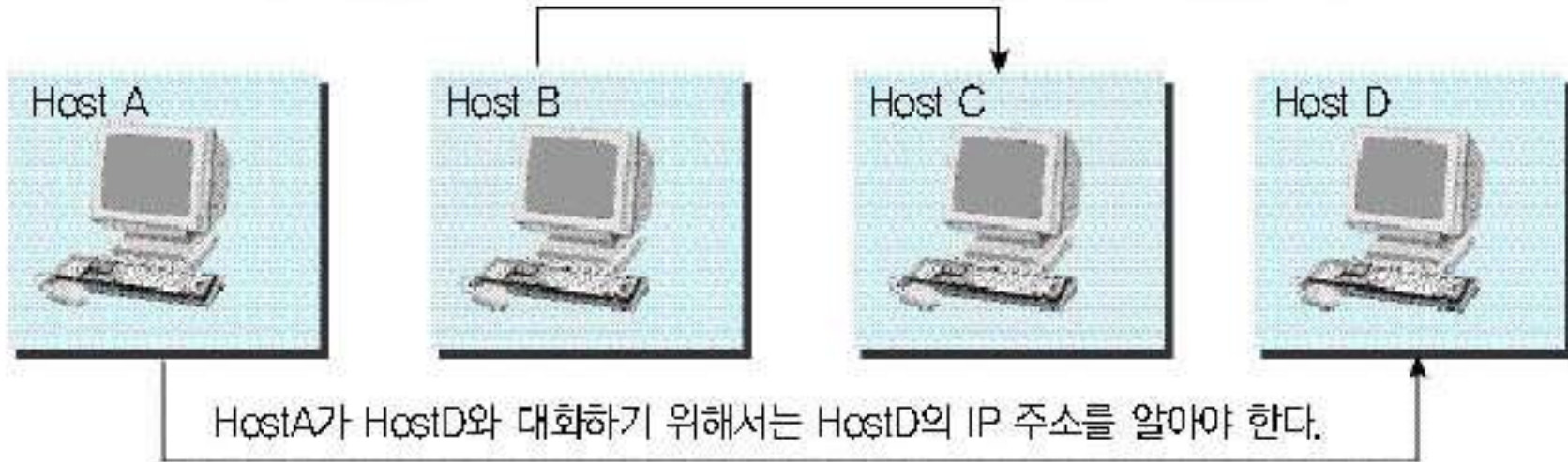
- 정보나 서비스를 제공하는 측을 서버라고 하며 제공 받는 쪽을 클라이언트

1. 네트워크에 대한 기본 개념

• IP 주소

- 인터넷상의 수많은 컴퓨터 중에서 특정 컴퓨터를 지정하기 위해서 사용
- 인터넷에 연결된 모든 컴퓨터는 고유한 IP 주소

HostB가 HostC와 대화하기 위해서는 HostC의 IP 주소를 알아야 한다.



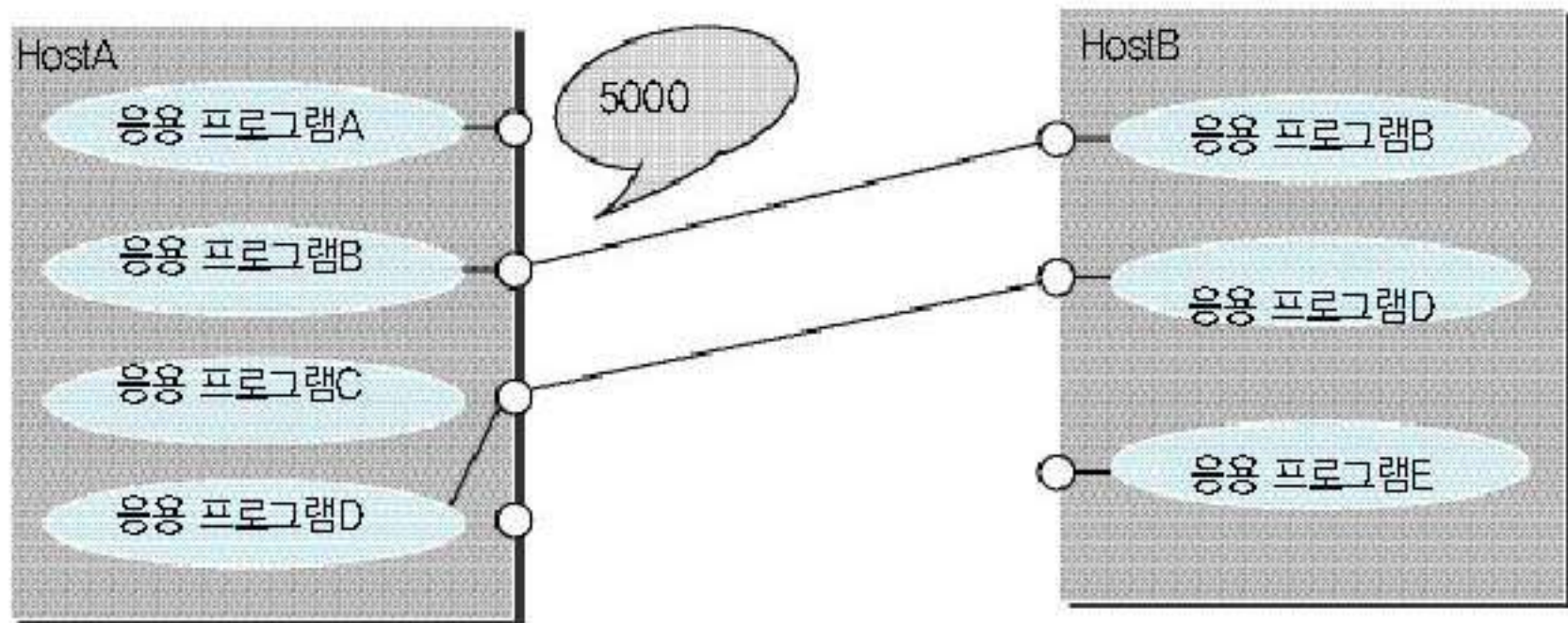
1. 네트워크에 대한 기본 개념

● 포트 번호

- 웹 브라우저, 메신저 등 다양한 프로그램 중에서 특정 프로그램으로 데이터를 보내기 위해서는 각 응용 프로그램을 구분하도록 한다.
- 1024보다 작은 값은 이미 상용화 된 프로그램에서 사용하므로 이 보다 큰 값으로 지정해야 충돌이 나지 않는다.

1. 네트워크에 대한 기본 개념

● 포트 번호



2. InetAddress 클래스

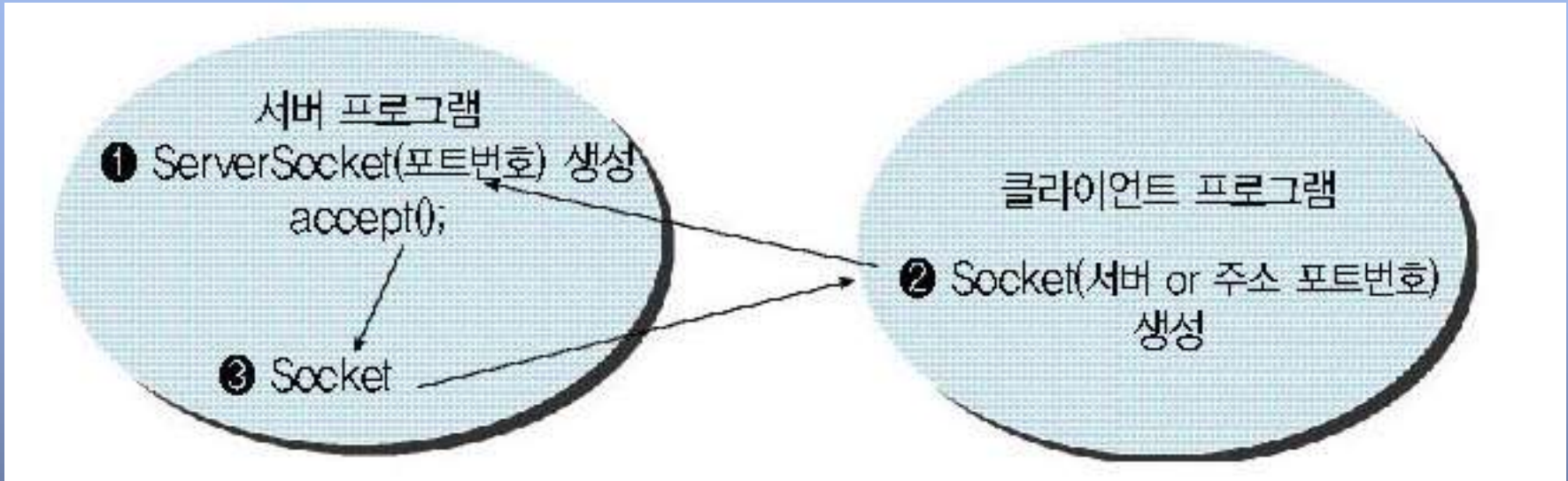
메서드	
<code>boolean equals(InetAddress other)</code>	other 객체와 같은 주소를 갖고 있으면 true, 아니면 false 반환
<code>byte[] getAddress()</code>	주소를 갖고 있는 4개 요소의 바이트 배열 반환
<code>String getHostAddress()</code>	주소 정보를 나타내는 문자열 반환
<code>String getHostName()</code>	컴퓨터 이름 반환
<code>InetAddress getLocalHost()</code>	현재 컴퓨터의 InetAddress 객체 반환
<code>InetAddress getByName(String host)</code>	host 로 지정된 컴퓨터를 나타내는 InetAddress 객체 반환
<code>InetAddress[] getAllByName(String host)</code>	host 로 지정된 모든 컴퓨터를 InetAddress 객체로 반환 (하나의 도메인 명으로 여러대의 컴퓨터가 작동할 경우)

<예제> InetAddress 클래스를 이용해서 IP 주소 알아내기

```
001:import java.net.*;
002:import java.io.*;
003:
004:class InetAddressTest02{
005:    public static void main(String [] args) throws Exception{
006:        BufferedReader reader;
007:        String url = null ;
008:        InetAddress addr = null;
009:
010:        reader = new BufferedReader(new InputStreamReader(System.in));
011:        System.out.print("웹사이트 주소를 입력하세요 -> ");
012:
013:        url = reader.readLine(); //사용자가 입력한 URL을 입력받음
014:
015:        try{
016:            //InetAddress.getByName(String url) 메서드를 이용해 InetAddress 객체를 생성
017:            addr = InetAddress.getByName(url);
018:        }catch(UnknownHostException e){
019:            e.printStackTrace();
020:        }
021:
022:        System.out.println("=====");
023:        //getHostAddress() 메서드를 이용해 IP 주소를 화면에 출력
024:        System.out.println( url +"의 IP 번호 = " + addr.getHostAddress());
025:    }
026:}
```

10.8

3. TCP를 이용한 에코 서버 클라이언트 작성



1. 서버 측의 ServerSocket이 먼저 특정 포트 번호를 열고 대기하고 있으면
2. 클라이언트 측 Socket 클래스가 생성할 때 서버 측 IP 주소와 지정한 포트 번호를 지정하면
3. 서버 측 프로그램에서는 대기 중인 ServerSocket의 accept 메서드가 접속을 요청한 클라이언트 측 소켓 객체를 반환합니다. 서버 측 프로그램에서도 이렇게 반환된 Socket 객체로 클라이언트 측의 Socket 객체와 통신을 하게 됩니다.

3. TCP를 이용한 에코 서버 클라이언트 작성

ServerSocket 클래스의 주요 메서드

생성자	
ServerSocket(int port)	클라이언트 요청을 받아들일 포트(port)번호를 갖고 ServerSocket 객체 생성
메서드	
Socket accept()	클라이언트 요청 받아 들인 다음 Socket 객체 생성해서 반환
void close()	서버 소켓 닫기

3. TCP를 이용한 에코 서버 클라이언트 작성

Socket 클래스의 주요 메서드

생성자	
Socket(String host, int port)	host : 접속할 서버의 IP 번호 port : 접속할 포트 번호 addr : 접속할 서버의 InetAddress 객체
Socket(InetAddress addr, int port)	
메서드	
InputStream getInputStream()	현재 소켓과 관련된 InputStream 객체 반환
OutputStream getOutputStream()	현재 소켓과 관련된 OutputStream 객체 반환
void close()	소켓 닫기
InetAddress getInetAddress()	소켓에 연결된 원격 컴퓨터의 InetAddress 객체 반환
InetAddress getLocalAddress()	소켓에 연결된 지역 컴퓨터의 InetAddress 객체 반환
int getPort()	소켓에 연결된 컴퓨터의 포트 번호 반환
int getLocalPort()	소켓에 연결된 지역 컴퓨터의 포트 번호 반환

3. TCP를 이용한 에코 서버 클라이언트 작성

- 서버 프로그램에서 서버 소켓이 클라이언트의 접속을 기다린다.

```
ServerSocket server = new ServerSocket(port); //포트 번호 5000  
Socket child = server.accept();//클라이언트 접속이 있기 전까지 대기 상태에 놓임
```

- 클라이언트 프로그램에서 서버 소켓에 접속을 요청한다.

```
Socket client = new Socket(ipAddress, port);
```

3. TCP를 이용한 에코 서버 클라이언트 작성

- Socket 으로부터 InputStream과 OutputStream 얻어서 입출력 스트림 생성

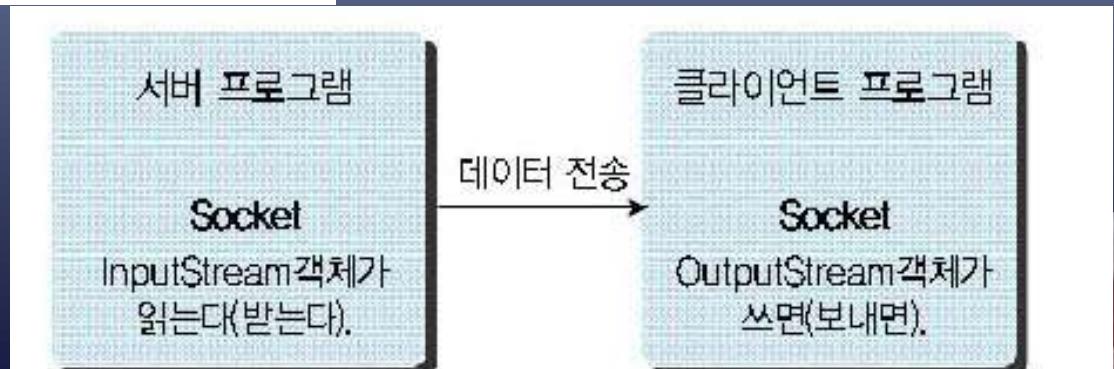
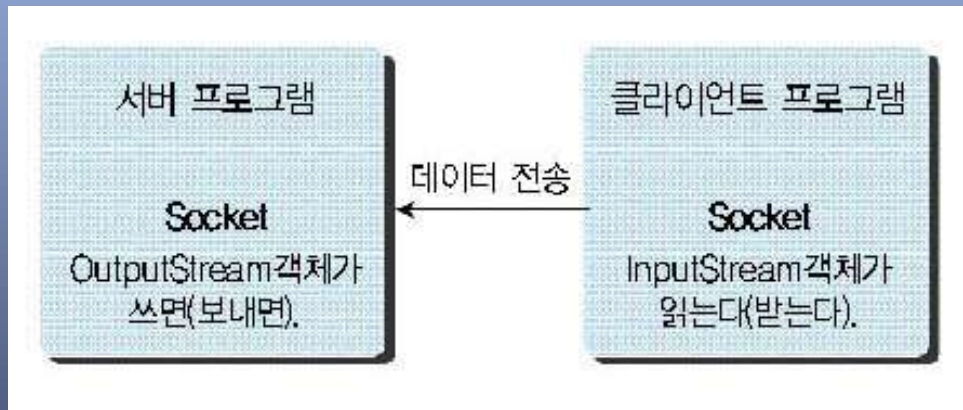
```
InputStream is = child.getInputStream();           //데이터를 받기위한 용도  
OutputStream os = child.getOutputStream();        //데이터를 보내기위한 용도
```

```
//입력 스트림을 ObjectInputStream으로 변환한다.  
ObjectInputStream ois = new ObjectInputStream(is);  
  
//출력 스트림을 ObjectOutputStream으로 변환한다.  
ObjectOutputStream oos = new ObjectOutputStream(os);
```

3. TCP를 이용한 에코 서버 클라이언트 작성

◆ 데이터 주고받기

```
String receiveData = (String)ois.readObject();  
oos.writeObject(receiveData);  
oos.flush();
```



3. TCP를 이용한 에코 서버 클라이언트 작성

- ◆ 소켓 닫기

```
ois.close();  
oos.close();  
child.close();
```

에코 서버 만들기

```
서버는 클라이언트 소켓의 접속 요청을 기다리고 있음  
/127.0.0.1로부터 연결요청 받음
```

```
Press any key to continue...
```

```
**** 클라이언트****
```

```
입력 ->안녕하십니까?
```

```
localhost/127.0.0.1로부터 받은 메시지<에코 됨>:안녕하십니까?
```

```
입력 ->오갱끼데스까?
```

```
localhost/127.0.0.1로부터 받은 메시지<에코 됨>:오갱끼데스까?
```

```
입력 ->quit
```

```
Press any key to continue...
```

에코 서버 만들기

```
001:import java.net.*;
002:import java.io.*;
003:public class EchoServerEx{
004:    ServerSocket server=null; //서버 소켓
005:    static final int port=5000; //포트 번호를 5000 번으로
006:    Socket child;             //클라이언트와 통신하기 위한 소켓
007:
008:    InputStream is;           //클라이언트와 연결된 입력 스트림 저장
009:    ObjectInputStream ois;     //클라이언트로부터 데이터를 전송받기 위한 스트림
010:
011:    OutputStream os;          //클라이언트와 연결된 출력 스트림 저장
012:    ObjectOutputStream oos;   //클라이언트에게 데이터를 전송하기 위한 스트림
013:
014:    String receiveData; //클라이언트로부터 전송받은 데이터를 저장할 변수
```

에코 서버 만들기

```
016:public EchoServerEx( ) {
017: try{
018:     //1. 에코 서버 프로그램은 포트를 지정해서 서버 소켓 생성부터 한다.
019:     server = new ServerSocket(port);
020:     System.out.println("**** 에코 서버****");
021:     System.out.println("서버는 클라이언트 소켓의 접속 요청을 기다리고 있음");
022:
023:     //1-1. 클라이언트의 접속을 항상 받아들일 수 있음
024:     //클라이언트의 요청이 없으면 대기 상태에 들어감
025:     //클라이언트가 접속하는 순간 클라이언트와 통신할 수 있는 소켓을 반환함
026:     child = server.accept();
027:
028:     //1-3. 접속이 되면 클라이언트로부터 아이피 주소를 얻어 출력함
029:     System.out.println(child.getInetAddress()+"로부터 연결요청 받음");
030:
031:     //3-1. 클라이언트가 보낸 데이터를 읽기 위해서 클라이언트로부터 입력 스트림을 얻음
032:     is = child.getInputStream();
033:     //3-2. 입력 스트림을 ObjectInputStream으로 변환한다.
034:     ois = new ObjectInputStream(is);
```

에코 서버 만들기

```
036: //3-3. 클라이언트로부터 받은 메시지를 다시 보내기 위해서 출력 스트림 생성
037: os = child.getOutputStream();
038: //3-4. 출력 스트림을 ObjectOutputStream으로 변환한다.
039: oos = new ObjectOutputStream(os);
040:
041: //4. 에코 서버는 스트림을 통해 클라이언트가 보낸 데이터를 서버가 읽어옴
042: while((receiveData = (String)ois.readObject()) != null){
043:     if(receiveData.equals("quit")) //클라이언트가 quit를 입력하면 접속 종료
044:         break;
045: //4-1. 클라이언트로부터 받은 데이터를 클라이언트에게 전송함-> 에코:메아리
046:     oos.writeObject(receiveData);
047:     oos.flush();
048: }
049:
050: is.close(); ois.close();
051: os.close(); oos.close();
052: }catch(Exception e){ //예외가 발생하면
053:     e.printStackTrace(); //에러 메시지를 출력하고
054:     System.exit(0); //프로그램을 종료한다.
055: }
056: }
057: public static void main(String[] args){
058:     new EchoServerEx( );
059: }
060: }
```

에코 클라이언트 만들기

- 클라이언트 프로그램에서는 키보드로부터 서버로 보낼 데이터를 한 줄 입력받기 위한 입력 스트림

```
BufferedReader read =  
    new BufferedReader(new InputStreamReader(System.in));
```

- BufferedReader 객체를 생성하여 이 객체로 키보드로부터 데이터를 입력 받는다.

```
String sendData = read.readLine();
```


에코 클라이언트 만들기

- 입력 받은 데이터를 서버로 보내기 위해서 출력합니다.

```
ObjectInputStream os = client.getOutputStream();  
ObjectOutputStream oos = new ObjectOutputStream(os);  
oos.writeObject(sendData);  
oos.flush();
```

에코 클라이언트 만들기

```
001:import java.net.*;
002:import java.io.*;
003:public class EchoClientEx{
004:    Socket client=null;           //클라이언트 소켓
005:
006:    String ipAddress; //접속을 요청할 서버의 아이피 주소를 저장할 변수 선언
007:    static final int port=5000; //접속 요청할 서버의 포트번호와 동일하게 지정
008:
009:    BufferedReader read;         //키보드로부터 메시지를 읽어올 입력 스트림
010:
011:    InputStream is;              //서버가 보낸 데이터를 읽기 위한 입력 스트림 저장
012:    ObjectInputStream ois;        //서버로부터 데이터를 전송받기 위한 스트림
013:
014:    OutputStream os;             //서버로 메시지를 보내기 위한 출력 스트림 저장
015:    ObjectOutputStream oos;      //서버에 데이터를 전송하기 위한 스트림
016:
017:    String sendData;             //서버로 보낼 데이터를 저장하기 위한 변수
018:    String receiveData;          //서버로부터 받은 데이터를 저장하기 위한 변수
```

에코 클라이언트 만들기

```
020: //생성자는 접속할 서버의 아이피를 전달 받음
021: public EchoClientEx(String ip) {
022:     ipAddress=ip;
023:
024:     try{
025:         System.out.println("**** 클라이언트****");
026:
027:         //2.접속할 서버의 아이피 주소와 포트를 이용해서 클라이언트 소켓 생성
028:         client = new Socket(ipAddress, port);
029:         //클라이언트 소켓이 생성되는 순간 서버의 accept 메서드가 수행된다.
030:
031:         //3-1. 키보드로부터 메시지를 읽어올 입력 스트림 생성
032:         read= new BufferedReader(new InputStreamReader(System.in));
033:
034:         //3-2. 서버로 메시지를 보내기 위해서 클라이언트로부터 출력스트림을 얻음
035:         os = client.getOutputStream();
036:         //3_3. 출력 스트림을 ObjectOutputStream으로 변환한다.
037:         oos = new ObjectOutputStream(os);
```

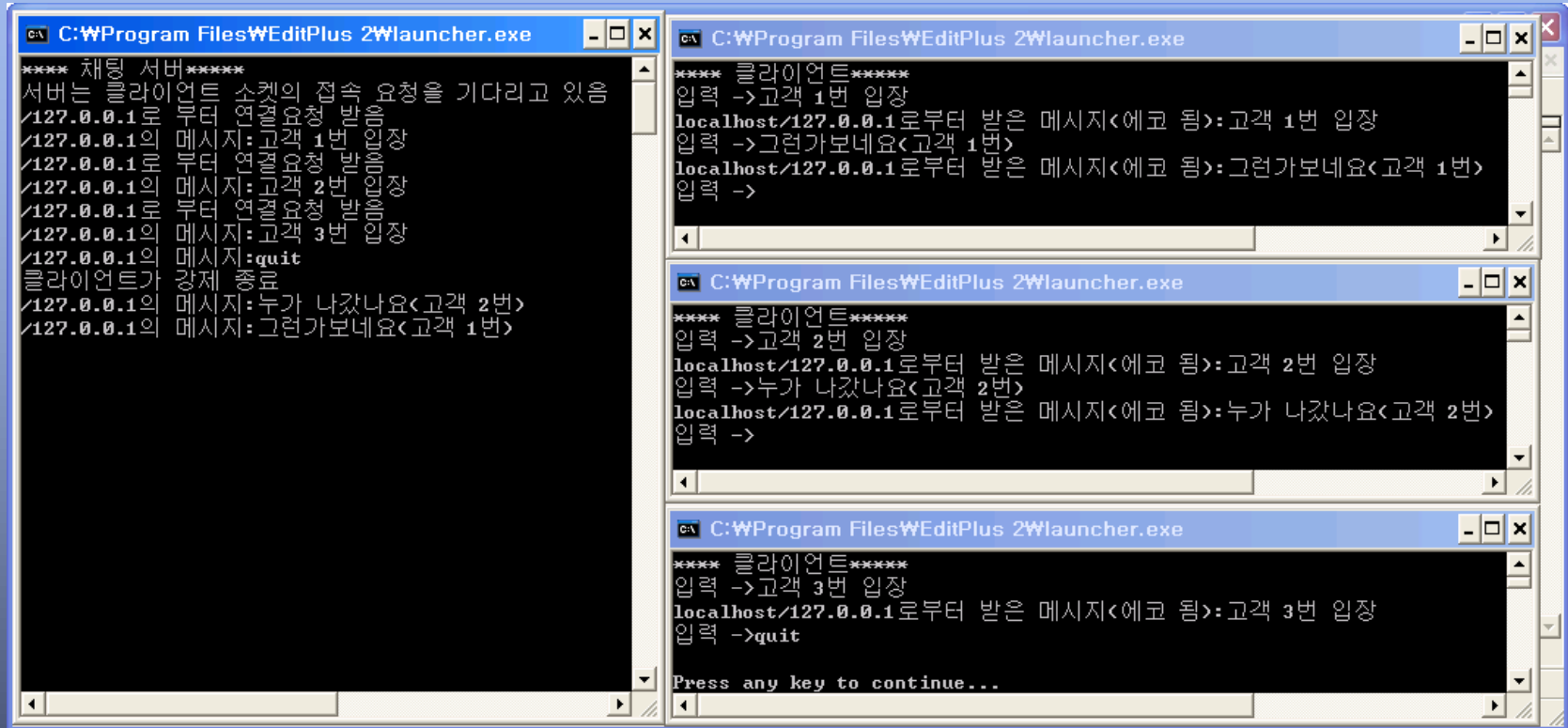
에코 클라이언트 만들기

```
039: //3-4. 서버의데이터를 수신받기 위해서 클라이언트로부터 입력 스트림을 얻어옴
040: is = client.getInputStream();
041: //3-5. 입력 스트림을 ObjectInputStream으로 변환한다.
042: ois = new ObjectInputStream(is);
043:
044: //4. 서버에게 보낼 데이터를 키보드에서 입력 받기
045: System.out.print("입력 ->");
046: //4-1. 키보드로부터 데이터를 입력 받음
047: while((sendData = read.readLine()) != null){
048:     //4-2. 서버로 데이터를 전송함
049:     oos.writeObject(sendData);
050:     oos.flush();
051:     if(sendData.equals("quit")) // "quit"란 문자열이 입력되면 종료
052:         break;
053:     //4-3. 스트림을 통해 데이터를 읽어옴
054:     receiveData = (String)ois.readObject();
055:     //4-4. 서버가 보낸 데이터를 다시 받아서 출력 (메아리처럼 )
056: System.out.println(client.getInetAddress()+"로부터 받은 메시지:"+receiveData);
057:     System.out.print("입력 ->");
058: }
```

에코 클라이언트 만들기

```
059: is.close(); ois.close();
060: os.close(); oos.close();
061: client.close();
062:     }catch(Exception e){
063:         e.printStackTrace(); //에러 메시지를 출력하고
064:         System.exit(0);      //프로그램을 종료한다.
065:     }
066: }
067:
068: public static void main(String[] args) {
069:     //원래는 다른 컴퓨터에서 서버를 돌려야 하지만
070:     //지금은 개념 파악을 위해서 하나의 컴퓨터를 서버와 클라이언트로 사용
071:     new EchoClientEx("localhost");
072:     //물론 두 대의 컴퓨터에서 돌려도 된다.
073:     //"localhost" 대신 에코 서버가 실행되는 컴퓨터의 아이피 주소를 입력한다.
074: }
075: }
```

4. 멀티 스레드를 이용한 에코 서버 클라이언트 프로그램



```
C:\Program Files\EditPlus 2\launcher.exe
**** 채팅 서버 ****
서버는 클라이언트 소켓의 접속 요청을 기다리고 있음
/127.0.0.1로 부터 연결요청 받음
/127.0.0.1의 메시지:고객 1번 입장
/127.0.0.1로 부터 연결요청 받음
/127.0.0.1의 메시지:고객 2번 입장
/127.0.0.1로 부터 연결요청 받음
/127.0.0.1의 메시지:고객 3번 입장
/127.0.0.1의 메시지:quit
클라이언트가 강제 종료
/127.0.0.1의 메시지:누가 나갔나요<고객 2번>
/127.0.0.1의 메시지:그런가보네요<고객 1번>
```

```
C:\Program Files\EditPlus 2\launcher.exe
**** 클라이언트 ****
입력 ->고객 1번 입장
localhost/127.0.0.1로부터 받은 메시지<에코 됨>:고객 1번 입장
입력 ->그런가보네요<고객 1번>
localhost/127.0.0.1로부터 받은 메시지<에코 됨>:그런가보네요<고객 1번>
입력 ->
```

```
C:\Program Files\EditPlus 2\launcher.exe
**** 클라이언트 ****
입력 ->고객 2번 입장
localhost/127.0.0.1로부터 받은 메시지<에코 됨>:고객 2번 입장
입력 ->누가 나갔나요<고객 2번>
localhost/127.0.0.1로부터 받은 메시지<에코 됨>:누가 나갔나요<고객 2번>
입력 ->
```

```
C:\Program Files\EditPlus 2\launcher.exe
**** 클라이언트 ****
입력 ->고객 3번 입장
localhost/127.0.0.1로부터 받은 메시지<에코 됨>:고객 3번 입장
입력 ->quit
Press any key to continue...
```

1. main 스레드 : 클라이언트의 접속 요청을 기다리는 스레드
2. EchoServerThread 스레드 : 클라이언트와 연결된 소켓 객체로부터 io를 얻어 데이터 송수신을 하는 스레드

main 스레드

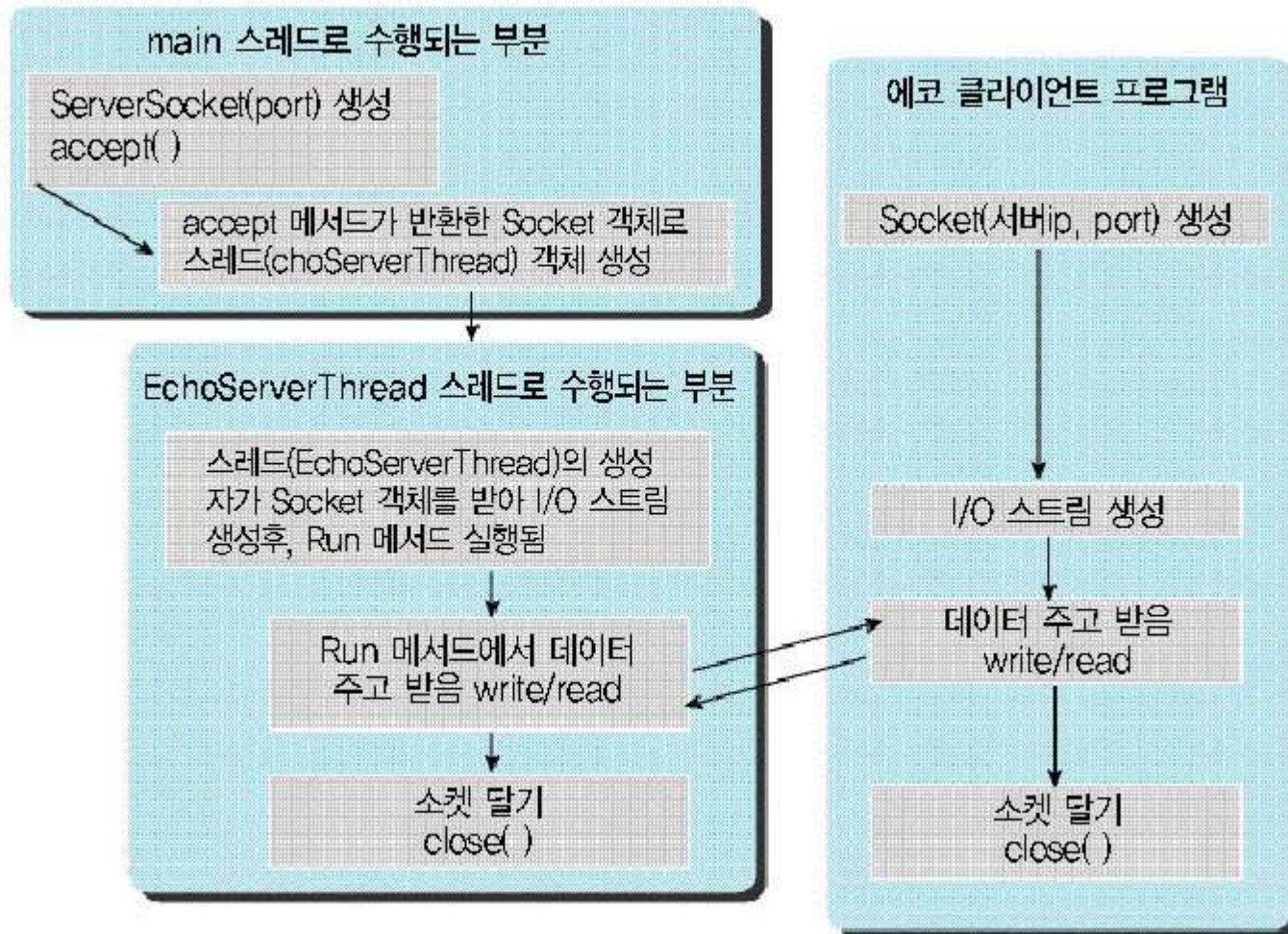
- 서버는 `accept()`로 대기하고 있다가 클라이언트가 접속하게 되면 `accept`하면서 `socket` 객체를 반환합니다.
- 반환한 소켓을 `EchoServerThread` 스레드의 생성자로 넘겨주면서 `EchoServerThread` 스레드 객체를 생성하고 에코 서버의 메인 스레드는 다시 `accept()`로 대기하고 있으면 됩니다.
- 이렇게 생성된 `EchoServerThread` 스레드가 클라이언트와 데이터 송수신을 하게 됩니다.

```
//(1) 클라이언트가 접속하는 순간 소켓을 반환함  
child = server.accept( );
```

```
//(2) 접속을 계속 유지하면서 데이터 송수신 하기 위해서 스레드 객체 생성  
EchoServerThread childThread=new EchoServerThread (child);  
Thread t = new Thread(childThread);  
t.start();
```

main 스레드

멀티 스레드 에코 서버 프로그램



EchoServerThread 클래스

◆ EchoServerThread 클래스의 생성자

```
public EchoServerThread(Socket s){ //접속 요청한 소켓 객체가 생성자로 보내지므로  
    child=s;           //클라이언트와 통신할 수 있는 소켓 정보를 child에 저장함
```

◆ EchoServerThread 클래스의 run 메서드

```
public void run( ){  
    while(true){  
        receiveData = (String)ois.readObject(); //데이터를 수신 받아서  
        oos.writeObject(receiveData); //클라이언트에게 다시 전송함->에코  
        oos.flush();  
    }//while 끝  
}//run 메서드 끝
```

<예제> 멀티 스레드를 이용한 에코 서버 만들기

```
001:import java.net.*;
002:import java.io.*;
003:public class MultiEchoServerEx{
004:    ServerSocket server;        //서버 소켓
005:    static final int port=5000; //포트 번호
006:    Socket child;               //클라이언트와 통신하기 위한 소켓
007:
008:    public MultiEchoServerEx( ) { //생성자
009:        try{
010:            //1. 에코 서버 프로그램은 포트를 지정해서 서버 소켓 생성부터 한다.
011:            server = new ServerSocket(port);
012:        }catch(Exception e){ //서버 소켓 생성에 실패하면
013:            e.printStackTrace(); //에러 메시지를 출력하고
014:            System.exit(0);      //프로그램을 종료한다.
015:        }
016:
017:        //2. 클라이언트의 접속을 항상 받아들일 수 있도록 무한루프를 돌림
018:        System.out.println("**** 채팅 서버****");
019:        System.out.println("서버는 클라이언트 소켓의 접속 요청을 기다리고 있음
");
```

<예제> 멀티 스레드를 이용한 에코 서버 만들기

```
021:while(true){
022:    try{
023:        //클라이언트의 요청이 없으면 대기 상태에 들어감
024:        //클라이언트가 접속하는 순간 클라이언트와 통신할 수 있는 소켓을 반환함
025:        child = server.accept( );
026:        /*****
027:        //접속을 계속 유지하면서 데이터 송수신 하기 위해서 스레드 객체 생성
028:        EchoServerThread childThread=new EchoServerThread(child);
029:        Thread t = new Thread(childThread);
030:        t.start();
031:        *****/
032:    }catch(Exception e){
033:        e.printStackTrace(); //에러 메시지를 출력하고
034:        System.exit(0);      //프로그램을 종료한다.
035:    }
036:} //while 끝
037:} //생성자 끝
038:public static void main(String[] args) {
039:    new MultiEchoServerEx( );
040:} //main 메서드 끝
041:} //MultiEchoServerEx 클래스 끝
```


<예제> 멀티 스레드를 이용한 에코 서버 만들기

```
043://접속을 계속 유지하면서 데이터 송수신 하기 위해서 스레드 클래스 생성
044:class EchoServerThread implements Runnable{
045:    Socket child;           //클라이언트와 통신하기 위한 소켓
046:
047:    InputStream is;         //클라이언트와 연결된 입력 스트림 저장
048:    ObjectInputStream ois;   //클라이언트로부터 데이터를 전송받기 위한 스트림
049:
050:    OutputStream os;        //클라이언트와 연결된 출력 스트림 저장
051:    ObjectOutputStream oos; //클라이언트에게 데이터를 전송하기 위한 스트림
052:
053:    String receiveData;      //클라이언트로부터 전송받은 데이터를 저장하기 위한 변수
054:
055:    public EchoServerThread(Socket s){ //접속 요청한 소켓 객체가 생성자로 보내진다.
056:        child=s; //클라이언트와 통신할 수 있는 소켓 정보를 child에 저장함
057:        try{
058:            //3. 접속이 되면 클라이언트로부터 아이피 주소를 얻어 출력함
059:            System.out.println(child.getInetAddress()+"로부터 연결요청 받음");
060:
061:            is = child.getInputStream(); //클라이언트로부터 데이터를 받기위한 입력 스트림
062:            ois = new ObjectInputStream(is);
063:
064:            os = child.getOutputStream();//클라이언트로부터 받은메시지를 보내기위한 출력스트림
065:            oos = new ObjectOutputStream(os);
066:        }catch(IOException e){
067:            e.printStackTrace();
068:        }
069:    } //생성자 끝
```


<예제> 멀티 스레드를 이용한 에코 서버 만들기

```
071:public void run(){
072:    try{
073:        while(true){
074:            receiveData = (String)ois.readObject();//클라이언트로부터 데이터를 받아
075:            System.out.println(child.getInetAddress()+"의 메시지:"+receiveData);
076:            oos.writeObject(receiveData); //클라이언트에게 데이터를 송신함-> 에코
077:            oos.flush();
078:        }//while
079:    }catch(Exception e){
080:        System.out.println("클라이언트가 강제 종료");
081:    }
082:    finally{
083:        try{
084:            is.close();
085:            os.close();
086:            ois.close();
087:            oos.close();
088:            child.close();
089:        }catch(IOException e){
090:            e.printStackTrace();
091:        }
092:    }//finally
093: }//run 메서드 끝
094: }//EchoServerThread 클래스 끝
```