chapter 15.

네트워크

- **TCP/IP** 모델에 대해서 알아본다.
- **TCP**와 **UDP**에 대해 알아본다.
- **InetAddress,URL,URLConnection** 클래스에 대해 알아본다.
- **Socket**과 **ServerSocket** 클래스의 흐름과 코딩 방법에 대해 알아본다.
- 유니캐스팅과 멀티캐스팅의 개념과 각각의 코딩 방법에 대해 알아본다.
- 프로토콜 개념과 설계 방법과 이를 이용한 로그인 인증 프로토콜을 만들어 본다.

❍ 네트워크

- 네트워크란 다른 장치로 데이터를 이동시킬 수 있는 컴퓨터들과 주변 장치들의 집합이다.
- 네트워크의 연결된 모든 장치들을 노드라고 한다.
- 다른 노드에게 하나 이상의 서비스를 해주는 노드를 호스트라 부른다.
- 하나의 컴퓨터에서 다른 컴퓨터로 데이터를 이동시킬 때 복잡한 계층을 통해 전송되는데, 이런 복잡한 레이어의 대표적인 모델이 OSI 계층 모델이다.
- OSI 계층 모델은 모두 7계층으로 이루어졌다.
- 데이터 통신을 이해하는데 OSI 계층 모델은 상당한 역할을 하지만, 인터넷 기반의 표준 모델로 사용하는 TCP/IP 계층 모델을 주로 사용하고 있다.
- 자바에서 이야기하는 네트워크 프로그래밍은 TCP/IP모델을 사용하고 있다.

❍ 인터넷 주소(IP 주소)

- 모든 호스트는 인터넷 주소(Host 또는 IP 주소)라 불리는 유일한 32 비트 숫자로 구성된 주소체계를 이용하여 서로를 구분할 수 있다.
- IP 주소는 32비트 숫자를 한번에 모두를 표현하는 것이 힘들기 때문에, 8 비트씩 끊어서 표현하고, 각 자리는 1바이트로 0~255 까지의 범위를 갖게 된다.
- 32비트의 주소 체계를 IP 버전 4(IPv4) 주소라고 한다.
- 오늘날 IPv4는 포화 상태이고, 이를 극복하고자 나온 것이 IP 버전 6(IPv6)이다.
- IPv6는 128 비트의 주소 체계를 관리하고 있으며, 16비트씩 8부분으로 나누어 16진수 표시한다.

```
FECD:BA98:7654:3210:FECD:BA98:7652:3210
```

- 각 호스트는 도메인 이름을 컴퓨터가 사용하는 주소(IP 주소)로 바꾸어 주어야 한다. 이렇게 IP 주소를 도메인 이름으로 바꾸어 주는 시스템을 DNS(Domain Name System)이라고 한다.

❍ 포트와 프로토콜

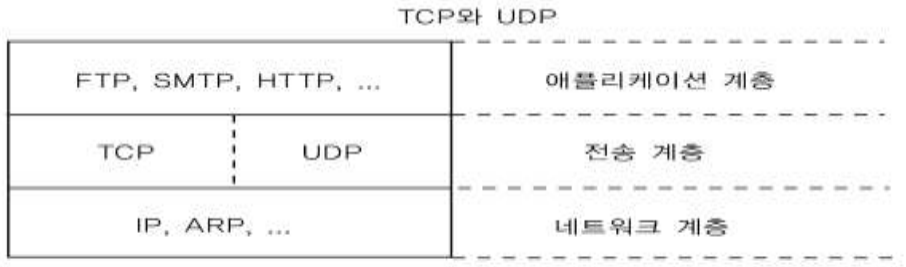
- 포트는 크게 두 가지로 구분된다.
- 컴퓨터의 주변자치를 접속하기 위한 '물리적인 포트'와 프로그램에서 사용되는 접속 장소인 '논리적인 포트'가 있다.
- 이 장에서 말하는 포트는 '논리적인 포트'를 말한다.
- 포트번호는 인터넷번호 할당 허가 위원회(IANA)에 의해 예약된 포트 번호를 가진다.
- 이런 포트번호를 '잘 알려진 포트들'라고 부른다.
- 예약된 포트번의 대표적인 예로는 80(HTTP), 21(FTP), 22(SSH), 23(TELNET)등이 있다.
- 포트번호는 0~65535까지 이며, 0~1023까지는 시스템에 의해 예약된 포트번호이기 때문에 될 수 있는 한 사용하지 않는 것이 바람직하다.

❍ 포트와 프로토콜

- 프로토콜은 클라이언트와 서버간의 통신 규약이다.
- 통신규약이란 상호 간의 접속이나 절단방식, 통신방식, 주고받을 데이터의 형식, 오류검출 방식, 코드변환방식, 전송속도 등에 대하여 정의하는 것을 말한다.
- 대표적인 인터넷 표준 프로토콜에는 TCP와 UDP가 있다.

❍ TCP와 UDP

- TCP/IP 계층 모델은 4계층의 구조를 가지고 있다.
- 애플리케이션, 전송, 네트워크 , 데이터 링크 계층이 있다.
- 이 중 전송계층에서 사용하는 프로토콜에는 TCP와 UDP가 있다.



[그림 15-2] TCP/IP 모델의 프로토콜

TCP

- TCP(Transmission Control Protocol)는 신뢰할 수 있는 프로토콜로서 , 데이터를 상대측까지 데대로 전달되었는지 확인 메시지를 주고 받음으로써 데이터의 송수신 상태를 점검한다.

UDP

- UDP(User Datagram Protocol)은 신뢰할 수 없는 프로토콜로서, 데이터를 보내기만 하고 확인 메시지를 주고 받지 않기 때문에 제대로 전달했는지 확인하지 않는다.
- TCP – 전화, UDP – 편지

InetAddress 클래스

- InetAddress 클래스는 IP 주소를 표현한 클래스이다.
- 자바에서는 모든 IP 주소를 InetAddress 클래스를 사용한다.

InetAddress 클래스의 생성자

- InetAddress 클래스의 생성자는 하나만 존재하지만, 특이하게 기본 생성자의 접근 제한자 default이기 때문에 new 연산자 객체를 생성할 수 없다.
- 따라서 InetAddress 클래스는 객체를 생성해 줄 수 있는 5개의 static 메서드를 제공하고 있다.

[표 15-1] InetAddress 객체를 생성하는 메서드

반환형	메서드	설명
static InetAddress[ ]	getAllByName(String host)	매개변수 host에 대응되는 InetAddress 배열을 반환한다.

static InetAddress	getByAddress(byte[ ] addr)	매개변수 addr에 대응되는 InetAddress 객체를 반환한다. 예를 들어, 209.249.116.141을 네 개의 바이트 배열로 만들어 매개변수로 지정하면 된다.
		`byte[] addr = new byte[4];`
		`addr[0] = (byte)209;`
		`addr[1] = (byte)249;`
		`addr[2] = (byte)116;`
		`addr[3] = (byte)141;`
		`InetAddress iaddr =`
		`InetAddress.getByAddress(addr);`

	getByAddress(String host, byte[ ] addr)	매개변수 host와 addr로 InetAddress객체를 생성한다.
static InetAddress	getByName(String host)	매개변수 host에 대응되는 InetAddress 객체를 반환한다.
	getLocalHost( )	로컬 호스트의 InetAddress 객체를 반환한다.

- 위의 5개의 static 메서드는 모두 UnkownHostException 예외를 발생시키기 때문에 반드시 예외처리를 해야 한다.

■ 예제[15-1] InetAddressEx.java

❍ InetAddress 주요 메서드

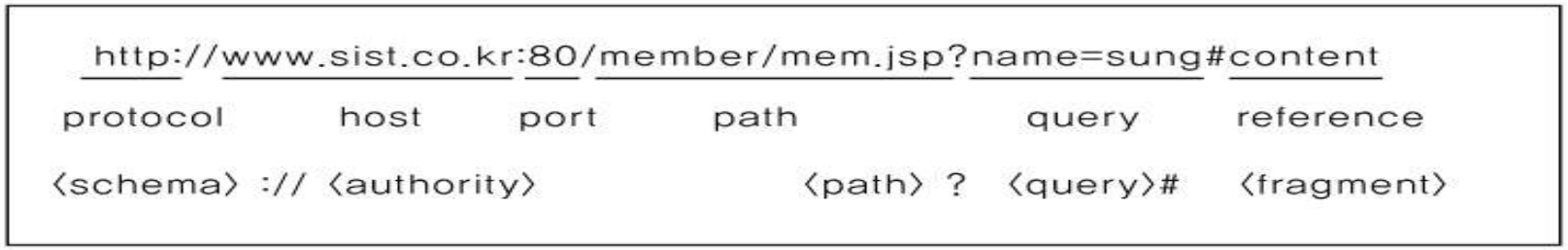
- InetAddress 클래스는 IP 주소를 객체화 했기 때문에 다양한 메서드를 제공하지 않는다.
- 다만 호스트 이름과 호스트에 대응하는 IP 주소를 알 수 있도록 메서드를 제공하고 있다.

[표 15-2] InetAddress 클래스의 주요 메서드

반환형	메서드	설명
byte[ ]	getAddress( )	InetAddress 객체의 실제 IP 주소를 바이트 배열로 리턴한다.
String	getHostAddress( )	IP 주소를 문자열로 반환한다.
	getHostName( )	호스트 이름을 문자열로 반환한다.
	toString( )	IP 주소를 스트링 문자열로 오버라이딩한 메서드다. 스트링 문자열의 형식은 '호스트 이름 /IP 주소'다. 따라서 InetAddress 객체를 화면에 출력하면 'java.sun.com /209. 249. 116.141'과 같이 출력된다.

❍ URL 클래스

- URL(Uniform Resource Locator)이란 인터넷에서 접근 가능한 자원의 주소를 표현할 수 있는 형식을 말한다.



[그림 15-4] URL의 구성요소

❍ URL 클래스의 생성자

- URL 클래스는 URL을 추상화 하여 만든 클래스다.
- URL 클래스는 final 클래스로 되어 있기 때문에 상속하여 사용할 수 없다.
- 모든 생성자는 MalformedURLException 예외를 발생하기 때문에 반드시 예외처리를 해야 한다.

[표 15-3] URL 클래스의 주요 생성자

생성자	설명
URL(String spec)	매개변수 spec으로 URL 객체를 생성한다. spec은 URL로 해석할 수 있는 문자열이어야 한다.
URL(String protocol, String host, int port, String file)	protocol과 host, port와 file로 URL 객체를 생성한다. 여기서 file이란 path와 query를 의미한다.
URL(String protocol, String host, String file)	protocol과 host, file로 URL 객체를 생성한다.

❍ URL 클래스의 주요 메서드

■ 예제[15-2] URLEx.java

[표 15-4] URL 클래스의 주요 메서드

반환형	메서드	설명
String	getAuthority( )	URL의 호스트명과 포트를 결합한 문자열을 반환한다.
int	getPort( )	URL에 명시된 포트를 반환한다. 만약에 없으면 -1을 반환한다. 일반적으로 HTTP 포트는 명시하지 않더라도 80번 포트를 인식하게 된다. 하지만 이 메서드는 URL에 보여진 포트를 리턴하는데, URL에 포트를 명시하지 않았다면 -1을 반환한다.
String	getDefaultPort( )	URL에 상관없이 프로토콜의 default 포트번호를 반환한다. 예) http → 80, ftp → 21
	getFile( )	URL의 path와 query를 결합한 문자열을 반환한다.
	getHost( )	URL의 host를 문자열로 반환한다.
	getPath( )	URL의 query를 문자열로 반환한다.
	getProtocol( )	URL의 protocol을 문자열로 반환한다.
	getQuery( )	URL의 query를 문자열로 반환한다.
	getRef( )	URL의 reference를 문자열로 반환한다.
URLConnection	openConnection( )	URLConnection 객체를 생성해준다. URLConnection 클래스는 추상 클래스이기 때문에 객체를 생성할 수 없고, URL 클래스의 openConnection( ) 메서드를 이용해서 객체를 생성할 수 있다.
InputStream	openStream( )	InputStream의 객체를 생성해준다. 이 메서드로 해당 URL의 자원(Resource)을 가져올 수 있다.
String	toExternalForm( )	URL을 문자열로 반환한다.

URLConnection 클래스

- URLConnection 클래스는 원격 자원에 접근하는 데 필요한 정보를 가지고 있다.
- 필요한 정보란 원격 서버의 헤더 정보, 해당 자원의 길이와 타입 정보, 언어 등을 얻어 올 수 있다.
- URL 클래스는 원격 서버 자원의 결과만을 가져 오지만, URLConnection 클래스는 원격 서버 자원의 결과와 원격 서버의 헤더 정보를 가져 올 수 있다.

URLConnection 클래스의 생성자

- URLConnection 클래스는 추상 클래스이기 때문에 단독적으로 객체를 생성할 수 없다.
- URL 클래스의 객체를 생성해서 URL 클래스의 openConnection() 메서드를 이용해서 객체를 생성해야 한다.
- URLConnection 객체가 생성 되었다면 URLConnection 클래스의 connect() 메서드를 호출해야 객체가 완성된다.

```
URL url = new URL("http://java.sun.com");
URLConnection urlCon = url.openConnection();
urlCon.connect();
```

URLConnection 클래스의 주요 메서드

■ 예제[15-3] URLConnectionEx.java

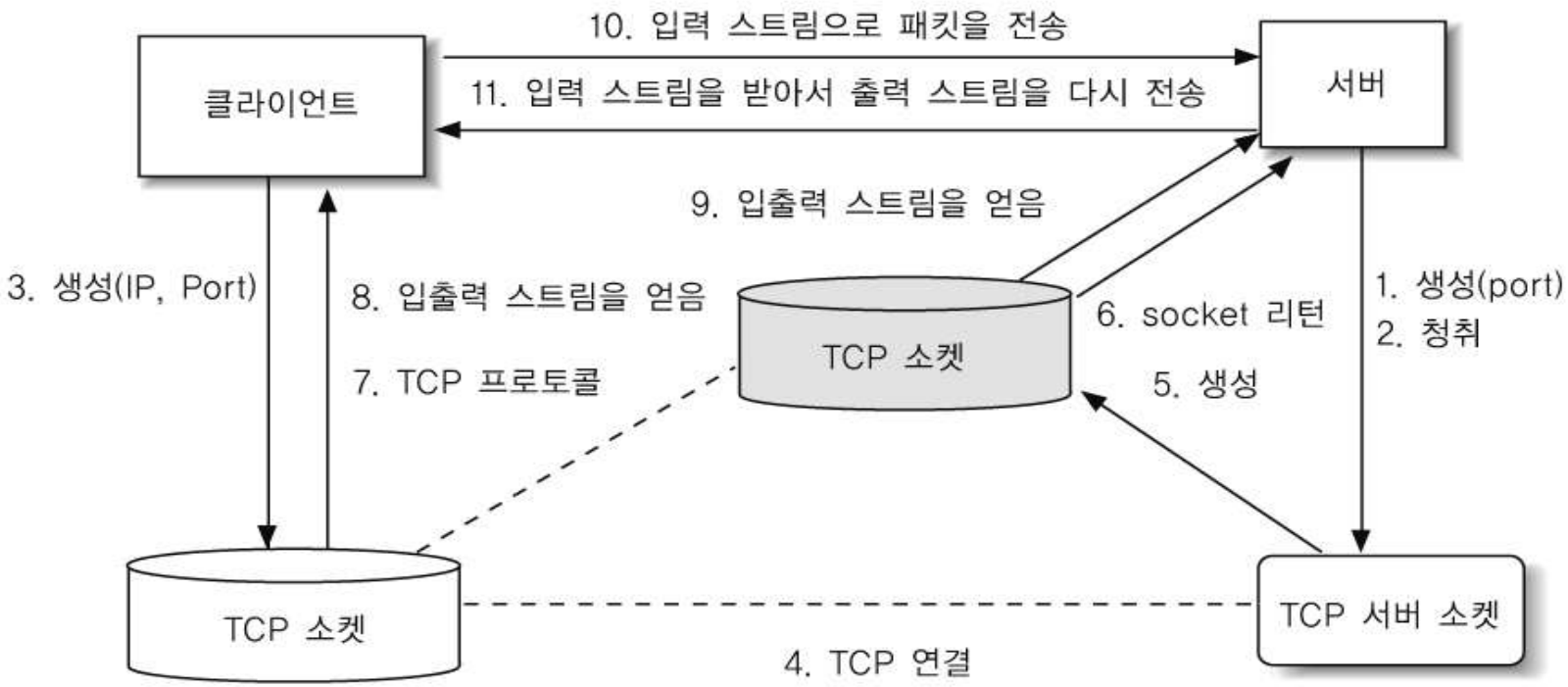
[표 15-6] URLConnection 클래스의 주요 메서드

반환형	메서드	설명
String	getContentEncoding( )	헤더 필드의 content-encoding에 대한 value를 반환한다.
int	getContentLength( )	헤더 필드의 content-length에 대한 value를 반환한다.
String	getHeaderField(String name)	헤더 필드의 이름(name)에 대한 value를 반환한다. 필드의 이름은 content-encoding, content-type 등이 올 수 있다.
Map<String, List<String>>	getHeaderFields( )	헤더 필드의 구조를 Map으로 반환한다.
InputStream	getInputStream( )	URLConnection 객체로부터 읽기 위한 InputStream 객체를 반환한다.
OutputStream	getOutputStream( )	URLConnection 객체로부터 출력(쓰기)하기 위한 OutputStream 객체를 반환한다.
URL	getURL( )	URLConnection의 멤버변수로 설정된 url 필드의 값을 리턴한다.

❍ 소켓

- 자바 프로그램은 소켓(Socket)이라는 개념을 통해서 네트워크 통신을 한다.
- 소켓은 네트워크 부분의 끝 부분을 나타내며, 실제 데이터가 어떻게 전송되는지 상관하지 않고 읽기/쓰기 인터페이스를 제공한다.
- 네트워크 계층과 전송 계층이 캡슐화 되어 있기 때문에 두 개의 계층을 신경 쓰지 않고 프로그램을 만들 수 있다.
- 소켓은 캘리포니아 대학교에서 빌 조이(Bill Joy)에 의해 개발되었다.
- 자바는 이식성과 크로스 플랫폼 네트워크 프로그램을 위해서 소켓을 핵심 라이브러리 만들었다.
- TCP/IP 계층의 TCP를 지원하기 위해서 Socket, ServerSocket 클래스를 제공하고 있다.
- 클라이언트는 Socket 객체를 생성하여 TCP 서버와 연결을 시도한다.
- 서버는 SocketServer 객체를 생성하여 TCP 연결을 청취하여 클라이언트와 서버가 연결된다.

❍Socket Stream



[그림 15-9] Socket과 ServerSocket의 통신방식

Socket 클래스

- TCP 소켓은 java.net.Socket 클래스를 의미한다.

Socket 클래스의 생성자

[표 15-7] Socket 클래스의 주요 생성자

생성자	설명
Socket(InetAddress address, int port)	InetAddress 객체와 port를 이용하여 Socket 객체를 생성한다.
Socket(String host , int port)	host와 port를 이용하여 Socket 객체를 생성한다.

- Socket 생성자는 두 가지 예외 처리가 발생한다.
- 첫 번째는 호스트를 찾을 수 없거나, 서버의 포트가 열려 있지 않은 경우 UnknownHostException 예외가 발생한다.
- 두 번째는 네트워크의 실패, 방화벽 때문에 서버에 접근 할 수 없을 때 IOException 예외가 발생한다.

예제[15-4] SocketScanEx.java

❍ Socket 클래스의 주요 메서드

[표 15-8] Socket 클래스의 주요 메서드

반환형	메서드	설명
void	close( )	소켓 객체를 닫는다.
InetAddress	getInetAddress( )	소켓 객체를 InetAddress 객체로 반환한다.
InputStream	getInputStream( )	소켓 객체로부터 입력할 수 있는 InputStream 객체를 반환한다.
InetAddress	getLocalAddress( )	소켓 객체의 로컬 주소를 반환한다.
int	getPort( )	소켓 객체의 포트를 반환한다.
boolean	isClosed( )	소켓 객체가 닫혀있으면 true를, 열려있으면 false를 반환한다.
	isConnected( )	소켓 객체가 연결되어 있으면 true, 연결되어 있지 않으면 false를 반환한다.
void	setSoTimeout(int timeout)	소켓 객체의 시간을 밀리 세컨드로 설정한다.

소켓을 이용한 입출력 스트림 생성

- TCP 소켓은 두 개의 네트워크 사이에서 바이트 스트림 통신을 제공한다.
- Socket 클래스는 바이트를 읽기 위한 메서드와 쓰기 위한 메서드를 제공한다.
- 두 가지 메서드를 이용하여 클라이언트와 서버간에 통신을 할 수 있다.

```
java.io.InputStream getInputStream() throws IOException;
java.io.OutputStream getOutputStream() throws IOException;
```

```
Socket socket = new Socket("211.238.132.50",4000);
InputStream in = socket.getIputStream();
OutputStream os = socket.getOutputStream();
```

❍ 소켓정보

- Socket 클래스는 로컬의 IP 주소와 포트를 알 수 있는 메서드와 Socket 으로 연결된 호스트의 IP 주소와 포트를 알 수 있는 메서드를 제공한다.
- 원격 호스트의 IP 주소를 알 수 있는 메서드이다.

```
public InetAddress getInetAddres() throws IOException;
```

- 원격 호스트의 PORT 번호를 알 수 있는 메서드이다.

```
public int getPort() throws IOException;
```

- 로컬 IP 주소를 알 수 있는 메서드이다.

```
public InetAddress getLocalAddres() throws IOException;
```

- 로컬 PORT 번호를 알 수 있는 메서드이다.

```
public int getLocalPort() throws IOException;
```

❍ 소켓 종료

- 소켓의 사용이 끝나면 연결을 끊기 위해서는 소켓의 close() 메서드를 호출해야 한다.
- 소켓 종료는 일반적으로 finally 블록에서 처리하고, close() 메서는 IOException를 발생하기 때문에 예외처리를 반드시 해야 한다.
- 소켓은 시스템에 의해 자동으로 종료되는 경우가 있다.
- 프로그램이 종료되거나, 가비지 컬렉터에 의해 처리되는 경우에 소켓이 자동으로 종료된다.
- 소켓이 시스템에 의해 자동으로 닫히는 것은 바람직 하지 않고, close() 메서드를 호출해서 정확히 소켓종료를 해야 한다.
- 소켓이 닫히더라도 getIntAddress() , getPort() 메서드는 사용할 수 있으나, getInputStream(), getOutputStream() 메서드는 사용할 수 없다.

TCP Server Socket

- ServerSocket 클래스가 TCP 서버 소켓을 의미한다.
- 클라이언트의 TCP 연결을 받기 위해서는 java.net.ServerSocket 클래스의 객체를 생성해야 한다.
- ServerSocket 클래스는 네트워크 통신을 수행하기 위해 자신을 바로 사용하는 것이 아니라 클라이언트의 TCP 요청에 대한 Socket 객체를 생성하는 역할을 한다.
- ServerSocket 객체를 생성했다면 ServerSocket 클래스의 accept() 메서드는 클라이언트의 TCP 요청이 있을 때 까지 블로킹 되는 메서드이다.
- 클라이언트의 TCP 요청이 오면 accept() 메서드는 클라이언트와 통신할 수 있는 TCP 소켓을 반환한다.
- 그런 후에 다른 클라이언트의 요청을 기다리게 되므로 일반적으로 accept() 메서드는 무한루프로 처리하게 된다.
- 클라이언트의 소켓과 서버에서는 accept() 메서드에 의해 반환 소켓을 가지고 스트림을 생성하여 통신하게 된다.

ServerSocket 클래스의 생성자

- 서버 소켓 생성자는 TCP 포트번호를 매개변수로 받는다.
- 만약, 기존의 TCP 포트번호가 사용중이라면 IOException을 발생하게 된다.

[표 15-9] ServerSocket 클래스의 주요 생성자

생성자	설명
ServerSocket(int port)	port를 이용하여 ServerSocket 객체를 생성한다.

■ 예제[15-5] ServerSocketScanEx.java

ServerSocket 클래스의 주요 메서드

- ServerSocket 클래스의 가장 중요한 메서드는 accept() 메서드이며, accept() 메서드의 시간 설정을 할 수 있는 메서드,ServerSocket를 종료할 수 있는 메서드를 제공하고 있다.

[표 15-10] ServerSocket 클래스의 주요 메서드

반환형	메서드	설명
Socket	accept( )	클라이언트의 Socket 객체가 생성될 때까지 블로킹되는 메서드다. 클라이언트의 Socket 객체가 생성되면 서버에서 클라이언트와 통신할 수 있는 Socket 객체를 반환하게 된다.
void	close( )	ServerSocket 객체를 닫는다.
int	getLocalPort( )	ServerSocket 객체가 청취하고 있는 포트번호를 반환한다.
	getSoTimeout( )	ServerSocket 클래스의 accept( ) 메서드가 유효할 수 있는 시간을 밀리 세컨드로 반환한다. 만약, 0이면 무한대를 의미한다.
boolean	isClosed( )	ServerSocket 객체의 닫힌 상태를 반환한다.
void	setSoTimeout(int timeout)	ServerSocket 클래스의 accept( ) 메서드가 유효할 수 있는 시간을 밀리 세컨드로 설정해야 한다. 만약, 시간이 지나면 java.net.SocketTimeoutException 예외가 발생하는데, 이 예외가 발생하더라도 ServerSocket 객체는 계속 유효하다.

ServerSocket 연결받기

- 서버 소켓의 주요 작업은 들어오는 연결 요청들을 수신하고 각 요청으로 부터 Socket 객체를 생성하는 것이다.
- 이런 역할을 수행하는 것이 ServerSocket 클래스의 accept() 메서드이다.
- 클라이언트에 들어오는 요청이 없다면 요청이 올 때까지 accept() 메서드는 블럭화 되거나 타임아웃이 되면 종료된다.

```
Socket accept() throws IOException,SecurityException;
```

- accpet() 메서드는 일반적으로 무한루프로 처리한다.
- 클라이언트의 TCP 요청이 오면 accept() 메서드를 통해 Socket 객체를 생성한 후에 다른 클라이언트의 TCP 요청을 기다리게 되므로 accept() 메서드를 무한루프로 처리해야 한다.

ServerSocket 정보

- 서버의 포트를 알 수 있는 메서드를 제공한다.

```
public int getLocalPort();
```

- accept() 메서드의 시간을 설정할 수 있는 메서드를 제공한다.

```
public void setSoTimeout(int timeout) throws SocketException;
```

- accept() 메서드의 시간을 알 수 있는 메서드를 제공하는데, 만약 0을 반환한다면 accept() 메서드의 유효시간은 무한대를 의미한다.

```
public int getSoTimeout() throws IOException;
```

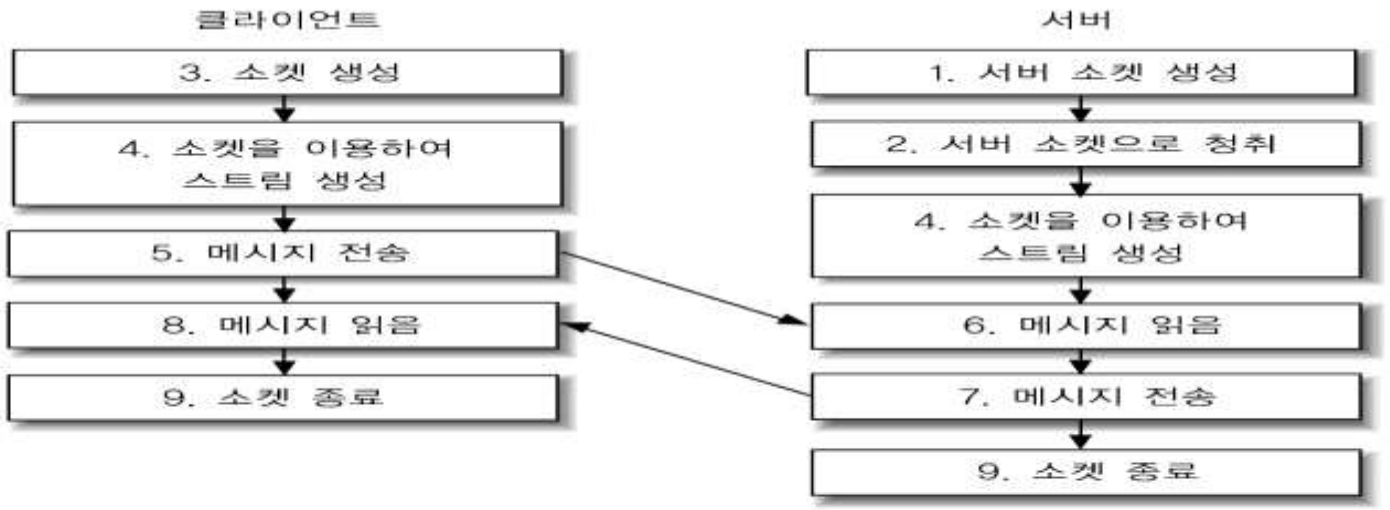
- 예제[15-6] EchoServer.java
- 예제[15-7] EchoClient.java

❍ ServerSocket 종료

- Socket 클래스의 종료처럼 서버 소켓의 종료는 close() 메서드로 간단하게 종료할 수 있다.

```
public void close() throws IOException;
```

❍ Socket과 ServerSocket을 이용한 간단한 에코 프로그램



[그림 15-12] 클라이언트와 서버의 흐름도

UDP(User Datagram Protocol)

- UDP는 비 연결 지향이고, IP 위에 매우 얇은 레이어로 구성되어 있다.
- UDP를 사용하는 애플리케이션은 TCP 프로그램에 비해 제어를 할 수 있는 부분이 적다.
- UDP는 데이터를 전송할 때에 데이터가 잘 도착했는지 알아낼 방법이 없으며, 데이터를 보낸 순서대로 도착한다는 보장도 할 수 없다.
- UDP는 TCP에 비해 훨씬 빠르게 전달된다는 장점이 있다.

DatagramPacket 클래스

- UDP 데이터그램은 java.net.DatagramPacket 클래스로 추상화한 것이다.
- DatagramPacket 클래스는 애플리케이션에서 주고 받을 데이터와 관련된 클래스이고, DatagramSocket 클래스는 실제 데이터의 전송을 책임지게 된다.
- DatagramPacket 클래스는 데이터를 송신하기 위한 기능과 수신을 하기 위한 기능으로 분리된다.

❍DatagramPacket 클래스의 생성자

- DatagramPacket의 생성자는 데이터를 보내기 위한 생성자와 데이터를 받기 위한 생성자로 구분된다.

[표 15-11] DatagramPacket 클래스의 주요 생성자

생성자	설명
DatagramPacket(byte[ ] buf, int length)	데이터를 수신하기 위한 생성자로 바이트 배열 buf의 length 만큼 저장한다.
DatagramPacket(byte[ ] buf, int length, InetAddress address, int port)	데이터를 송신하기 위한 생성자로 address와 port로 바이트 배열 buf의 length만큼 저장한다.
DatagramPacket(byte[ ] buf, int offset, int length)	데이터를 수신하기 위한 생성자로 바이트 배열 buf의 offset 위치에서 length만큼 저장한다.
DatagramPacket(byte[ ] buf, int offset, intlength, InetAddress address, int port)	데이터를 송신하기 위한 생성자로 address와 port로 바이트 배열 buf의 offset 위치에서 length만큼 저장한다.

❍ DatagramPacket 클래스의 메서드

- IP 헤더에 출발지 주소와 목적지 주소를 설정하거나 주소를 얻어오는 메서드, 출발지 포트와 목적지 포트를 설정하거나 얻어오는 다양한 메서드 제공한다.

[표 15-12] DatagramPacket 클래스의 주요 메서드

반환형	메서드	설명
InetAddress	getAddress( )	데이터그램에 대한 목적지 또는 출발지 주소를 반환한다.
byte[ ]	getData( )	버퍼에 들어있는 실제 데이터를 바이트 배열로 반환한다.
int	getLength( )	버퍼에 들어있는 실제 데이터의 길이를 반환한다.
	getOffset( )	버퍼에 들어있는 실제 데이터의 시작 위치를 반환한다
	getPort( )	데이터그램에 대한 목적지 또는 출발지 포트를 반환한다.
void	setAddress (InetAddress iaddr)	데이터그램을 보낸 호스트 주소를 설정한다.
	setData(byte[ ] buf)	버퍼에 들어있는 실제 데이터를 바이트 배열 buffer로 설정한다.
void	setData(byte[ ] buf, int offset, int length)	버퍼에 들어있는 실제 데이터를 바이트 배열 buffer의 offset 위치에서 length만큼 설정한다.
	setLength(int length)	버퍼에 들어있는 실제 데이터의 길이를 설정한다.
	setPort(int port)	데이터그램에 대한 목적지 또는 출발지 포트를 설정한다.

❍ DatagramSocket 클래스

- TCP 스트림 소켓과 달리 서버와 클라이언트 데이터 그램 소켓 사이에는 차이가 없으며 모든 데이터그램 소켓은 데이터그램을 전송할 뿐만 아니라 수신에서 사용할 수 있다.

❍ DatagramSocket 클래스의 생성자

- 모든 DatagramSocket 객체는 데이터그램을 수신하기 위해서 사용될 수 있기 때문에 로컬 호스트 내의 유일한 UDP 포트와 연관되어 있다.

[표 15-13] DatagramSocket 클래스의 주요 생성자

생성자	설명
DatagramSocket( )	할당된 특정한 포트번호가 중요하지 않다면 사용 가능한 임시 UDP 포트로 소켓을 생성하여 DatagramSocket 객체를 생성한다.
DatagramSocket(int port)	매개변수 포트로 소켓을 생성하여 DatagramSocket 객체를 생성한다.
DatagramSocket(int port, InetAddress iaddr)	매개변수 port와 iaddr로 소켓을 생성하여 DatagramSocket 객체를 생성한다.

❍ DatagramSocket 클래스의 주요 메서드

- DatagramSocket 클래스의 주요 메서드 기능은 DatagramPacket을 보내거나 받을 수 있는 메서드를 제공하는 것이다.

[표 15-14] DatagramSocket 클래스의 주요 메서드

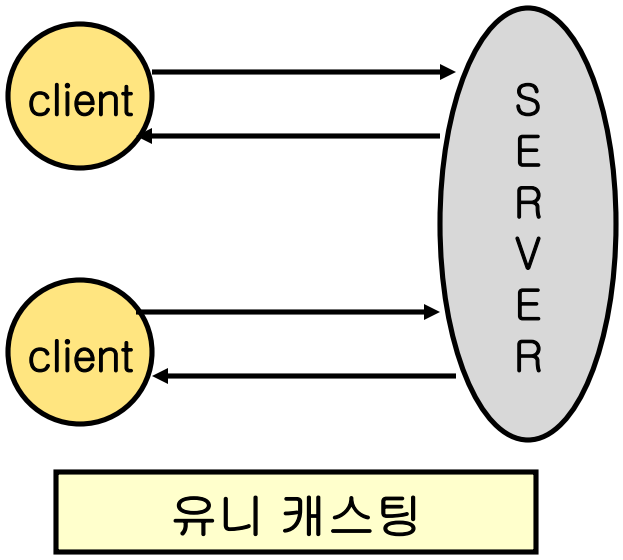
반환형	메서드	설명
void	send(DatagramPacket dp)	UDP 데이터그램(dp)을 전송하는 메서드다.
	receive(DatagramPacket dp)	UDP 데이터그램을 받아서 이미 존재하는 DatagramPacket 객체인 dp에 저장한다.
	close( )	데이터그램 소켓이 점유하고 있는 포트를 자유롭게 놓아준다.
int	getLocalPort( )	현재 소켓이 데이터그램을 기다리고 있는 로컬 포트가 몇 번인지를 리턴한다.
void	connect(InetAddress address, int port)	DatagramSocket이 지정된 호스트의 지정된 포트하고만 패킷을 주고받을 것이라고 정한다.
	disconnect( )	현재 연결된 DatagramSocket의 연결을 끊는다. 연결이 끊기면 아무것도 하지 못하게 된다.
int	getPort( )	DatagramSocket이 연결되어 있다면 소켓이 연결되어 있는 원격지 포트번호를 반환한다.
InetAddress	getInetAddress( )	DatagramSocket이 연결되어 있다면 소켓이 연결되어 있는 원격지 주소를 반환한다.

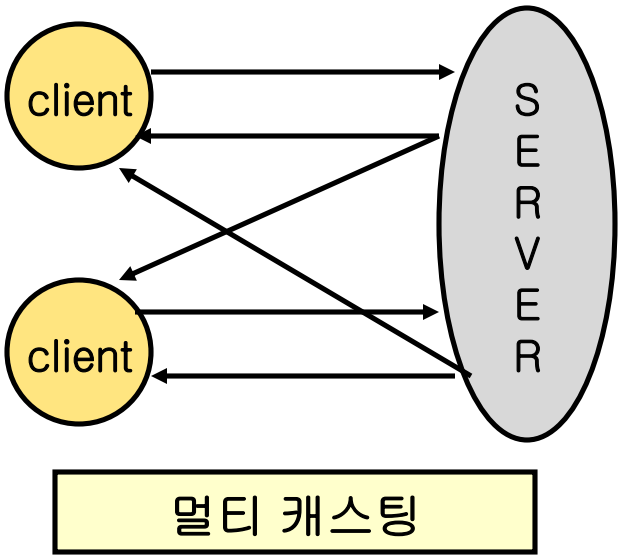
- DatagramPacket과 DatagramSocket을 이용한 에코 프로그램

  - 예제[15-8] UDPEchoServer.java
  - 예제[15-7] UDPEchoClient.java

유니 캐스팅관 멀티 캐스팅

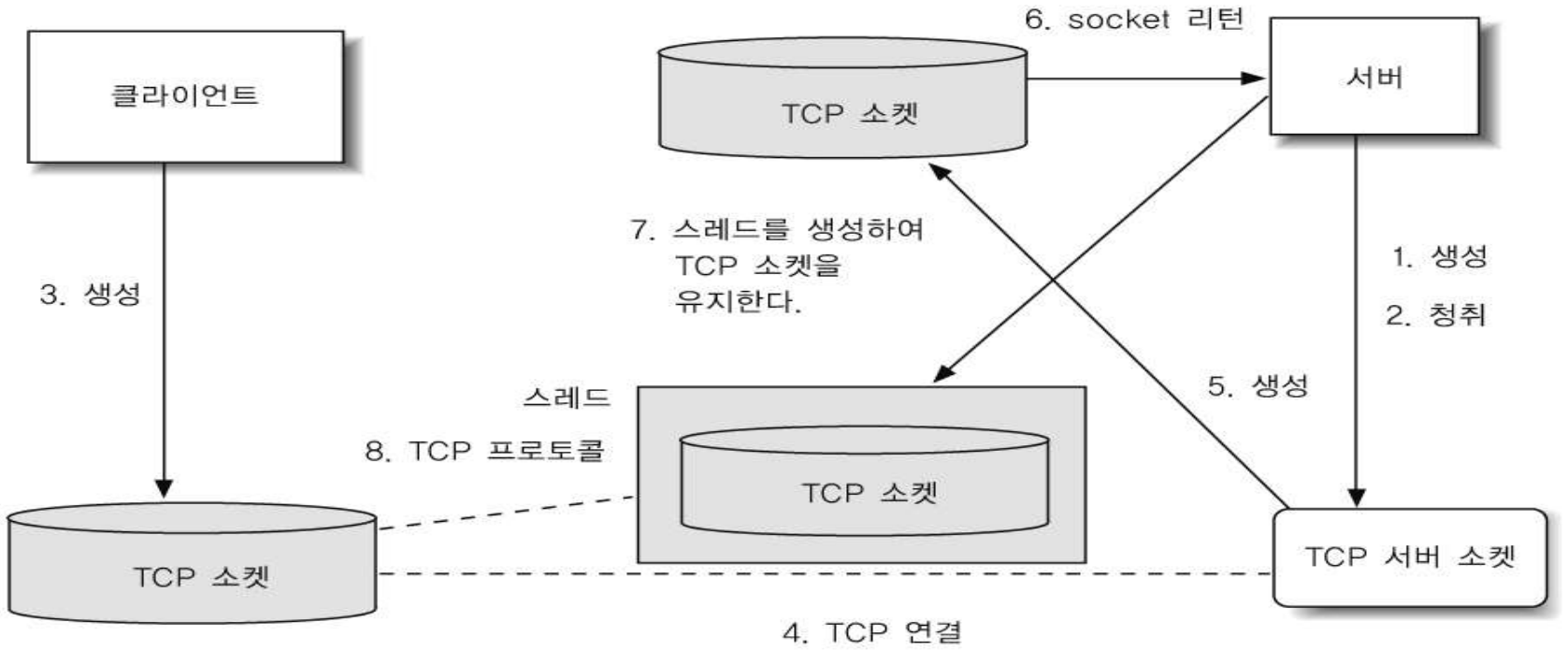
- 클라이언트와 서버간의 지속적으로 일대 일로 통신하는 개념을 유니 캐스팅이라고 한다.
- 일대 다의 통신을 멀티 캐스팅이라고 한다.





❍ 유니 캐스팅

- 유니 캐스팅을 구현하기 위해서는 필수 조건이 서버측에 스레드를 생성해서 TCP 소켓을 유지해야 한다.



[그림 15-18] 스레드를 이용한 유니캐스트 흐름도

❍유니 캐스트 프로그램

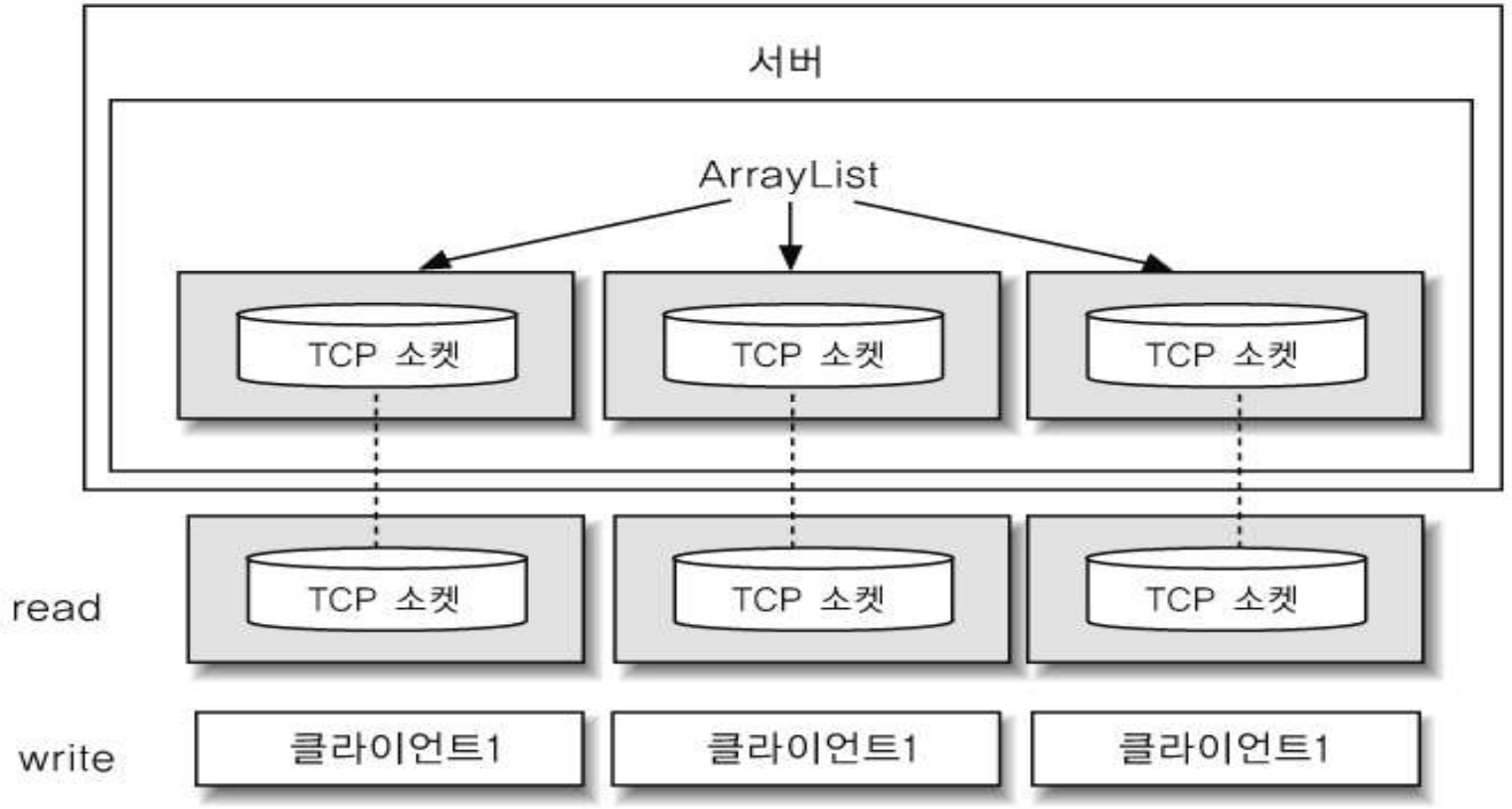
[표 15-15] 유니캐스트 예제에 필요한 클래스 구조

클래스명	설명
UnicastServer	모든 클라이언트의 TCP 요청을 받아 Socket 객체를 생성하고, Socket를 유지하기 위한 스레드를 생성하는 클래스다.
UnicastServerThread	각각의 클라이언트의 Socket 객체를 유지하기 위한 클래스다.
UnicastClient	클라이언트가 콘솔에서 접속하기위한 클래스다.

- 예제[15-10] UnicastServer.java
- 예제[15-11] UnicastServerThread.java
- 예제[15-12] UnicastClient.java

멀티 캐스팅

- 유니 캐스트 모델은 실시간 프로그램에서 서버의 정보를 모든 클라이언트가 공유할 때 문제점이 있다.
- 이런 문제를 해결할 수 있는 방법이 일대 다 전송을 지원하는 멀티 캐스팅 방법이다.
- 한명의 클라이언트가 서버의 정보를 변경했을 경우 모든 클라이언트에게 전송함으로써 서로가 변경된 정보를 공유할 수 있는 애플리케이션을 만들 때 적합하다.
- 멀티 캐스팅 프로그램을 작성하기 위해서는 유니캐스트에서 생성된 스레드를 저장하기 위한 공간(ArrayList)이 필요하며, 클라이언트에서는 자신이 보낸 메시지나 다른 클라이언트가 보낸 메시지를 받기 위한 스레드가 필요하다.



[그림 15-21] ArrayList를 이용한 멀티캐스트 흐름도

❍멀티 캐스트 프로그램

[표 15-16] 멀티캐스트 프로그램에 필요한 클래스 구조

클래스명	설명
MultiServer	모든 클라이언트의 TCP 요청을 받아 소켓 객체를 생성한다. 소켓을 유지하기 위한 스레드를 생성하고 이 스레드를 저장할 Collection(ArrayList)을 생성하는 클래스다.
MultiServerThread	각각의 클라이언트의 소켓 객체를 유지하기 위한 클래스다. 이 클래스는 멀티서버에 있는 컬렉션을 가지고 있기 때문에 다른 클라이언트에게 메시지를 보낼 수 있다.
MultiClient	스윙으로 구현된 클라이언트 클래스다. 이 클래스에서는 메시지를 보낼 때는 이벤트에서 처리했고, 다른 클라이언트가 보낸 메시지를 받기 위해 MultiClientThread 객체를 생성했다.
MultiClientThread	다른 클라이언트의 메시지를 받기 위한 클래스다.

- 예제[15-13] MulticastServer.java
- 예제[15-14] MulticastServerThread.java
- 예제[15-15] MultiClient.java
- 예제[15-16] MultiClientThread.java

❍ 프로토콜의 설계 정의

- 프로토콜이란 클라이언트와 서버간의 통신 규약이다.

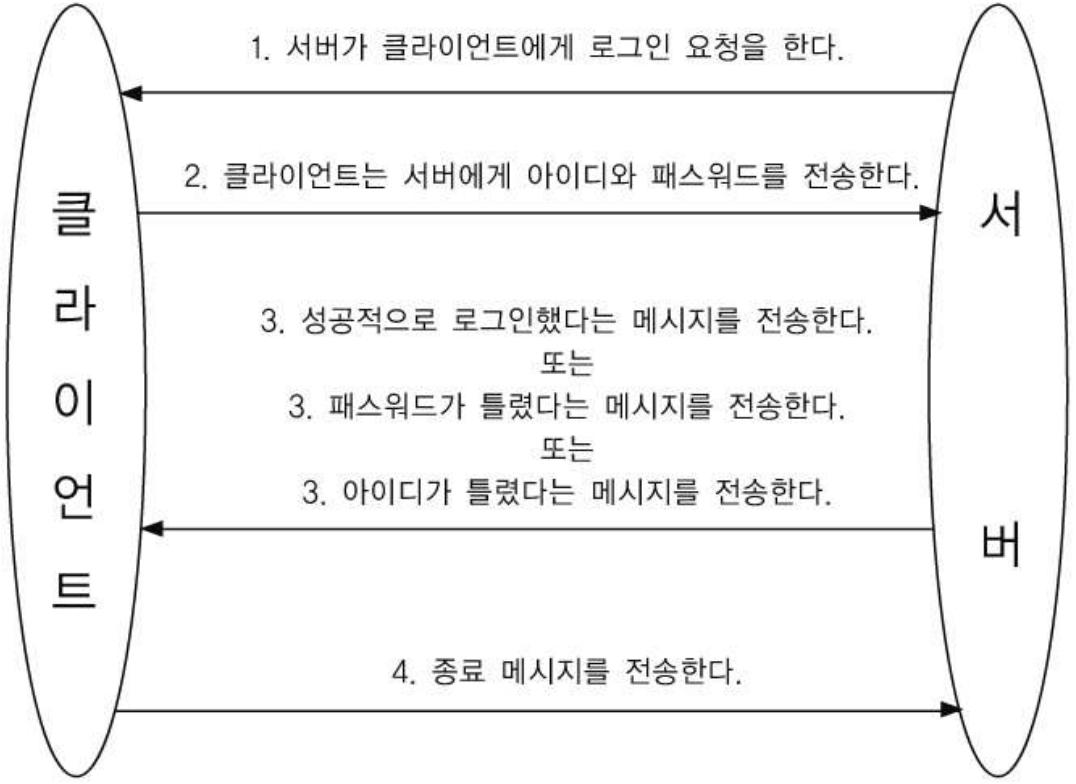
[그림 15-25] 프로토콜 분석 (1)

- 클라이언트가 보낸 메시지를 서버에서는 "##"를 구분자로 문자열을 토큰하여 문자를 분석하게 된다.
- 만약 이런 규약을 클라이언트가 위배하여 메시지의 순서를 바꾼다든지, 부적절한 메시지를 보내게 되면 서버에서는 이를 파악하지 못하고 항상 동일한 처리를 하게 되기 때문에 다른 클라이언트에게 적절치 못한 메시지를 전송하게 된다.

❍ 프로토콜의 설계 기법

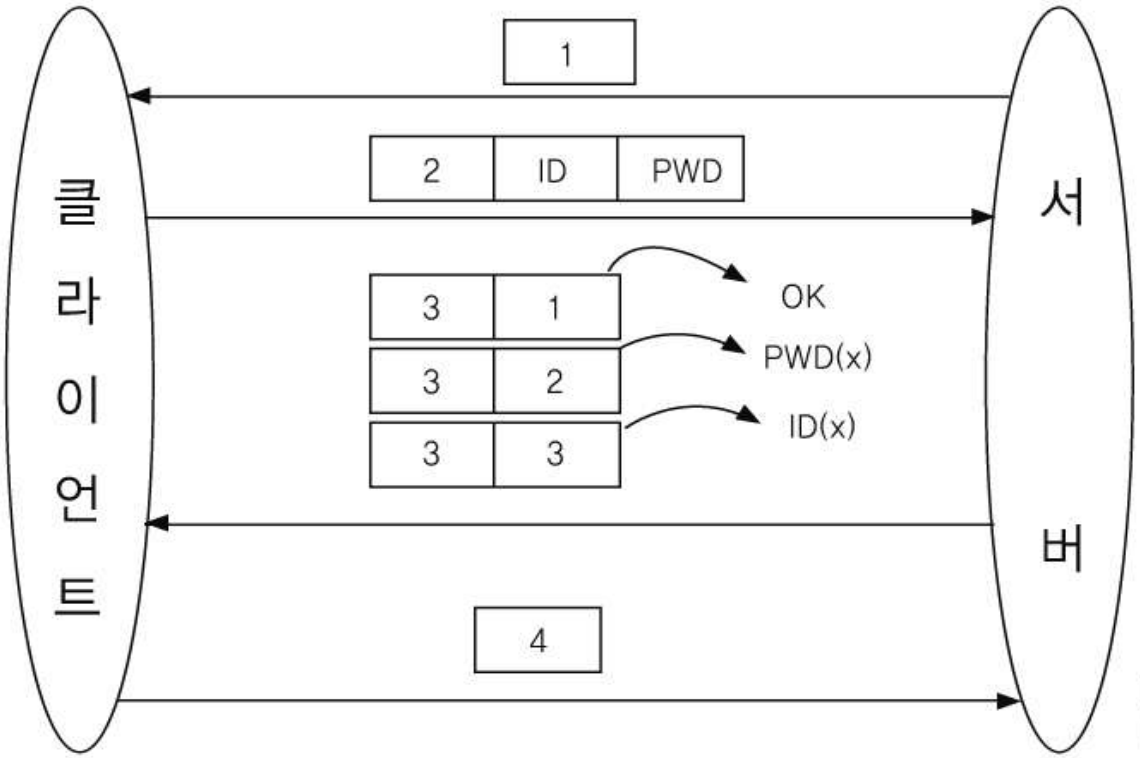
- 프로토콜의 설계는 클라이언트와 서버간의 통신 규약을 만드는데 필요한 데이터가 무엇인지를 설계하는 것이다.
- 프로토콜의 설계 하기위해서는 클라이언트에서 필요한 기능이 무엇인지를 생각하고, 그 기능에 대해 필요한 데이터가 어떤 것이 있는지를 분석할 수 있는 능력이 있어야 한다.

❍로그인 인증을 위한 요구사항



[그림 15-27] 로그인 인증을 위한 요구사항

❍로그인 인증에 필요한 데이터



[그림 15-28] 로그인 인증을 위한 데이터 흐름

❍ 프로토콜을 이용한 클라이언트/서버 구축

- 요구사항과 각각의 요구사항에 필요한 데이터의 설계가 끝났다면 이것을 가지고 프로토콜을 정의해야 한다.
- 프로토콜을 정의한 클래스가 Protocol.java 파일이다.

- 예제[15-17] Protocol.java
- 예제[15-18] LoginServer.java
- 예제[15-19] LoginClient.java