

Java



TCP/IP 프로토콜에 대하여 TCP/IP 통신 프로그램의 작성 방법



Network Introduction

● Network

- ▶ Network란 다른 장치로 Data를 이동시킬 수 있는 Computer들과 주변 장치들의 집합이다.
- ▶ Network의 연결된 모든 장치들을 Node라고 한다.
- ▶ 다른 Node에게 하나 이상의 Service를 해주는 Node를 Host라 부른다.
- ▶ 하나의 Computer에서 다른 Computer로 Data를 이동시킬 때 복잡한 계층을 통해 전송되는데, 이런 복잡한 Layer의 대표적인 Model이 OSI7 layer Model이다.
- ▶ OSI 계층 Model은 모두 7계층으로 이루어졌다.
- ▶ Data 통신을 이해하는데 OSI7 Layer Model은 상당한 역할을 하지만, Internet 기반의 표준 Model로 사용하는 TCP/IP 계층 Model을 주로 사용하고 있다.
- ▶ Java에서 이야기하는 Network Programming은 TCP/IP Model을 사용하고 있다.

Network Introduction

Internet Address(IP Address)

- ▶ 모든 Host는 Internet Address(Host 또는 IP Address)라 불리는 유일한 32bit 숫자로 구성된 Address체계를 이용하여 서로를 구분할 수 있다.
- ▶ IP Address는 32bit 숫자를 한번에 모두를 표현하는 것이 힘들기 때문에, 8 bit씩 끊어서 표현하고, 각 자리는 1byte로 0~255 까지의 범위를 갖게 된다.
- ▶ 32bit의 Address 체계를 IP Version 4(IPv4) Address라고 한다.
- ▶ 오늘날 IPv4는 포화 상태이고, 이를 극복하고자 나온 것이 IP Version 6(IPv6)이다.
- ▶ IPv6는 128 bit의 Address 체계를 관리하고 있으며, 16bit씩 8부분으로 나누어 16진수 표시한다.

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

- 각 Host는 Domain 이름을 Computer가 사용하는 Address(IP Address)로 바꾸어 주어야 한다. 이렇게 IP Address를 Domain 이름으로 바꾸어 주는 System을 DNS(Domain Name System)이라고 한다.

Network Introduction

Port와 Protocol

- ▶ Port는 크게 두 가지로 구분된다.
- ▶ Computer의 주변장치를 접속하기 위한 '물리적인 Port'와 Program에서 사용되는 접속 장소인 '논리적인 Port'가 있다.
- ▶ 이 장에서 말하는 Port는 '논리적인 Port'를 말한다.
- ▶ Port번호는 Internet번호 할당 허가 위원회(IANA)에 의해 예약된 Port번호를 가진다.
- ▶ 이런 Port번호를 '잘 알려진 Port들'라고 부른다.
- ▶ 예약된 Port번의 대표적인 예로는 80(HTTP), 21(FTP), 22(SSH), 23(TELNET)등이 있다.
- ▶ Port번호는 0~65535까지이며, 0~1023까지는 System에 의해 예약된 Port번호이기 때문에 될 수 있는 한 사용하지 않는 것이 바람직하다.

Network Introduction

Port와 Protocol

- ▶ Protocol은 Client와 Server간의 통신 규약이다.
- ▶ 통신규약이란 상호 간의 접속이나 절단방식, 통신방식, 주고받을 Data의 형식, 오류검출 방식, 코드변환방식, 전송속도 등에 대하여 정의하는 것을 말한다.
- ▶ 대표적인 Internet 표준 Protocol에는 TCP와 UDP가 있다.

TCP와 UDP

- ▶ TCP/IP 계층 Model은 4계층의 구조를 가지고 있다.
- ▶ Application, Transfer, Network, Data Link 계층이 있다.
- ▶ 이 중 Transfer계층에서 사용하는 Protocol에는 TCP와 UDP가 있다.

TCP와 UDP

FTP, SMTP, HTTP, ...		애플리케이션 계층
TCP	UDP	전송 계층
IP, ARP, ...		네트워크 계층

Network Introduction

● TCP

- ▶ TCP(Transmission Control Protocol)는 신뢰할 수 있는 Protocol로서, Data를 상대 측까지 제대로 전달되었는지 확인 Message를 주고 받음으로써 Data의 송수신 상태를 점검한다.

● UDP

- ▶ UDP(User Datagram Protocol)은 신뢰할 수 없는 Protocol로서, Data를 보내기만 하고 확인 Message를 주고 받지 않기 때문에 제대로 전달했는지 확인하지 않는다.
- ▶ TCP - 전화, UDP - 편지

InetAddress class

■ InetAddress class

- ➡ InetAddress class는 IP Address를 표현한 class이다.
- ➡ Java에서는 모든 IP Address를 InetAddress class를 사용한다.

■ InetAddress class의 Constructor

- ➡ InetAddress class의 Constructor는 하나만 존재하지만, 특이하게 기본 Constructor의 Access Modifier가 default이기 때문에 new 연산자 Object를 생성할 수 없다.
- ➡ 따라서 InetAddress class는 Object를 생성해 줄 수 있는 5개의 static method를 제공하고 있다.

InetAddress class

※ 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.
	getByName (String host)	매개변수 host에 대응되는 InetAddress 객체를 반환한다.
	getLocalHost()	로컬 호스트의 InetAddress 객체를 반환한다.

▶ 위의 5개의 static method는 모두 UnknownHostException 예외를 발생시키기 때문에 반드시 예외처리를 해야 한다.

InetAddress class

● InetAddress 주요 method

- ➡ InetAddress class는 IP Address를 Object화 했기 때문에 다양한 method를 제공하지 않는다.
- ➡ 다만 Host 이름과 Host에 대응하는 IP Address를 알 수 있도록 method를 제공하고 있다.

※ InetAddress 객체를 생성하는 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 class

● URL class

- ➡ URL(Uniform Resource Locator)이란 Internet에서 접근 가능한 Resource의 Address를 표현할 수 있는 형식을 말한다.

<u>http://www.sist.co.kr:80/member/mem.jsp?name=sung#content</u>					
protocol	host	port	path	query	reference
<schema>	://	<authority>	<path>	? <query>	# <fragment>

URL class

● URL class의 Constructor

- ▶ URL class는 URL을 추상화 하여 만든 class다.
- ▶ URL class는 final class로 되어 있기 때문에 상속하여 사용할 수 없다.
- ▶ 모든 Constructor는 MalformedURLException 예외를 발생하기 때문에 반드시 예외처리를 해야 한다.

※ 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 class

● URL class의 주요 method

※ 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. ex) http → 80, ftp → 21
	getFile()	URL의 path와 query를 결합한 문자열을 반환한다.
	getHost()	URL의 host를 문자열로 반환한다.
	getPath()	URL의 path를 문자열로 반환한다.
	getProtocol()	URL의 protocol를 문자열로 반환한다.
	getQuery()	URL의 query를 문자열로 반환한다.
URLConnection	getRef()	URL의 reference를 문자열로 반환한다.
	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 class

● URLConnection class

- ➡ URLConnection class는 원격 Resource에 접근하는 데 필요한 정보를 가지고 있다.
- ➡ 필요한 정보란 원격 Server의 Header 정보, 해당 Resource의 길이와 타입 정보, 언어 등을 얻어 올 수 있다.
- ➡ URL class는 원격 Server Resource의 결과만을 가져 오지만, URLConnection class는 원격 Server Resource의 결과와 원격 Server의 Header 정보를 가져 올 수 있다.

URLConnection class

- URLConnection class의 Constructor
 - ▶ URLConnection class는 Abstract class이기 때문에 단독적으로 Object를 생성할 수 없다.
 - ▶ URL class의 Object를 생성해서 URL class의 openConnection() method를 이용해서 Object를 생성해야 한다.
 - ▶ URLConnection Object가 생성 되었다면 URLConnection class의 connect() method를 호출해야 Object가 완성된다.

```
URL url = new URL( "http://edu.kosta.or.kr" );  
URLConnection urlCon = url.openConnection();  
urlCon.connect();
```

URLConnection class

URLConnection class의 주요 method

※ 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.

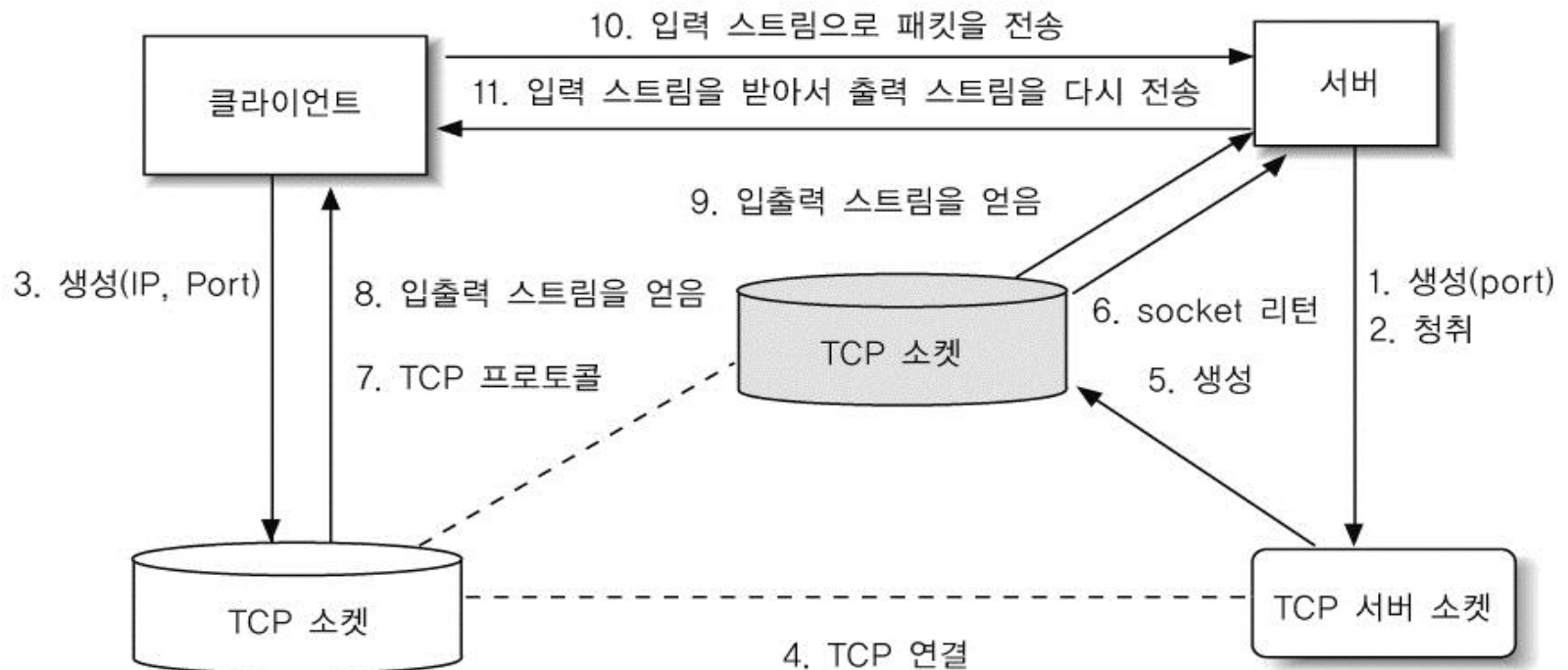
TCP

■ Socket

- ▶ Java Program은 Socket이라는 개념을 통해서 Network 통신을 한다.
- ▶ Socket은 Network 부분의 끝 부분을 나타내며, 실제 Data가 어떻게 전송되는지 상관하지 않고 읽기/쓰기 Interface를 제공한다.
- ▶ Network 계층과 Transfer 계층이 캡슐화 되어 있기 때문에 두 개의 계층을 신경 쓰지 않고 Program을 만들 수 있다.
- ▶ Socket은 캘리포니아 대학교에서 Bill Joy에 의해 개발되었다.
- ▶ Java는 이식성과 Cross Platform Network Program을 위해서 Socket을 핵심 library로 만들었다.
- ▶ TCP/IP 계층의 TCP를 지원하기 위해서 Socket, ServerSocket class를 제공하고 있다.
- ▶ Client는 Socket Object를 생성하여 TCP Server와 연결을 시도한다.
- ▶ Server는 SocketServer Object를 생성하여 TCP 연결을 청취하여 Client와 Server가 연결된다.

TCP

Socket Stream



TCP

● Socket class

➡ TCP Socket은 `java.net.Socket` class를 의미한다.

● Socket class의 Constructor

※ Socket 클래스의 주요 생성자

생성자	설명
<code>Socket(InetAddress address, int port)</code>	<code>InetAddress</code> 객체와 <code>port</code> 를 이용하여 <code>Socket</code> 객체를 생성한다.
<code>Socket(String host, int port)</code>	<code>host</code> 와 <code>port</code> 를 이용하여 <code>Socket</code> 객체를 생성한다.

- `Socket` Constructor는 두 가지 예외 처리가 발생한다.
- 첫 번째는 `Host`를 찾을 수 없거나, `Server`의 `port`가 열려 있지 않은 경우 `UnknownHostException` 예외가 발생한다.
- 두 번째는 `Network`의 실패, 방화벽 때문에 `Server`에 접근 할 수 없을 때 `IOException` 예외가 발생한다.

TCP

● Socket class의 주요 method

※ 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.

TCP

- Socket을 이용한 Input/Output Stream 생성
 - TCP Socket은 두 개의 Network 사이에서 Byte Stream 통신을 제공한다.
 - Socket class는 Byte를 읽기 위한 method와 쓰기 위한 method를 제공한다.
 - 두 가지 method를 이용하여 Client와 Server간에 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.getInputStream();  
OutputStream os = socket.getOutputStream();
```

TCP

Socket 정보

- ▶ Socket class는 local의 IP Address와 Port를 알 수 있는 method와 Socket 으로 연결된 Host의 IP Address와 Port를 알 수 있는 method를 제공한다.

- 원격 Host의 IP Address를 알 수 있는 method이다.

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

- 원격 Host의 PORT number를 알 수 있는 method이다.

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

- local IP Address를 알 수 있는 method이다.

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

- local PORT number를 알 수 있는 method이다.

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

TCP

Socket 종료

- Socket의 사용이 끝나면 연결을 끊기 위해서는 Socket의 `close()` method를 호출해야 한다.
- Socket 종료는 일반적으로 `finally Block`에서 처리하고, `close()` 메서드는 `IOException`를 발생하기 때문에 예외처리를 반드시 해야 한다.
- Socket은 System에 의해 자동으로 종료되는 경우가 있다.
- Program이 종료되거나, Garbage Collector에 의해 처리되는 경우에 Socket이 자동으로 종료된다.
- Socket이 System에 의해 자동으로 닫히는 것은 바람직 하지 않고, `close()` method를 호출해서 정확히 Socket종료를 해야 한다.
- Socket이 닫히더라도 `getInetAddress()`, `getPort()` method는 사용할 수 있으나, `getInputStream()`, `getOutputStream()` method는 사용할 수 없다.

TCP

■ TCP Server Socket

- ➡ `ServerSocket` class가 TCP Server Socket을 의미한다.
- ➡ Client의 TCP 연결을 받기 위해서는 `java.net.ServerSocket` class의 `Object`를 생성해야 한다.
- ➡ `ServerSocket` class는 Network 통신을 수행하기 위해 자신을 바로 사용하는 것이 아니라 Client의 TCP 요청에 대한 `Socket Object`를 생성하는 역할을 한다.
- ➡ `ServerSocket Object`를 생성했다면 `ServerSocket` class의 `accept()` method는 Client의 TCP 요청이 있을 때 까지 Blocking 되는 method이다.
- ➡ Client의 TCP 요청이 오면 `accept()` method는 Client와 통신할 수 있는 TCP Socket을 반환한다.
- ➡ 그런 후에 다른 Client의 요청을 기다리게 되므로 일반적으로 `accept()` method는 무한 loop로 처리하게 된다.
- ➡ Client의 Socket과 Server에서는 `accept()` method에 의해 반환 Socket을 가지고 Stream을 생성하여 통신하게 된다.

● ServerSocket class의 Constructor

- Server Socket Constructor는 TCP Port number를 parameter로 받는다.
- 만약, 기존의 TCP Port번호가 사용 중이라면 IOException을 발생하게 된다.

※ ServerSocket 클래스의 주요 생성자

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

TCP

■ ServerSocket class의 주요 method

- ServerSocket class의 가장 중요한 method는 accept() method이며, accept() method의 시간 설정을 할 수 있는 method, ServerSocket를 종료할 수 있는 method를 제공하고 있다.

※ 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 밀리세컨드로 반환한다. 만약, 0이면 무한대를 의미한다.
boolean	isClosed()	ServerSocket 객체의 닫힌 상태를 반환한다.
void	setSoTimeout(int timeout)	ServerSocket 클래스의 accept() 메서드가 유효할 수 있는 시간을 밀리세컨드로 설정해야 한다. 만약, 시간이 지나면 java.net.SocketTimeoutException 예외가 발생하는데, 이 예외가 발생하더라도 ServerSocket 객체는 계속 유효하다.

TCP

● ServerSocket 연결받기

- ▶ ServerSocket의 주요 작업은 들어오는 연결 요청들을 수신하고 각 요청으로 부터 Socket Object를 생성하는 것이다.
- ▶ 이런 역할을 수행하는 것이 ServerSocket class의 accept() method이다.
- ▶ Client에 들어오는 요청이 없다면 요청이 올 때까지 accept() method는 Block화 되거나 타임아웃이 되면 종료된다.

Socket accept() throws IOException, SecurityException;

- accept() method는 일반적으로 무한 loop로 처리한다.
- Client의 TCP 요청이 오면 accept() method를 통해 Socket Object를 생성한 후에 다른 Client의 TCP 요청을 기다리게 되므로 accept() method를 무한 loop로 처리해야 한다.

TCP

● ServerSocket 정보

- ➡ Server의 Port를 알 수 있는 method를 제공한다.

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

- accept() method의 시간을 설정할 수 있는 method를 제공한다.

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

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

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

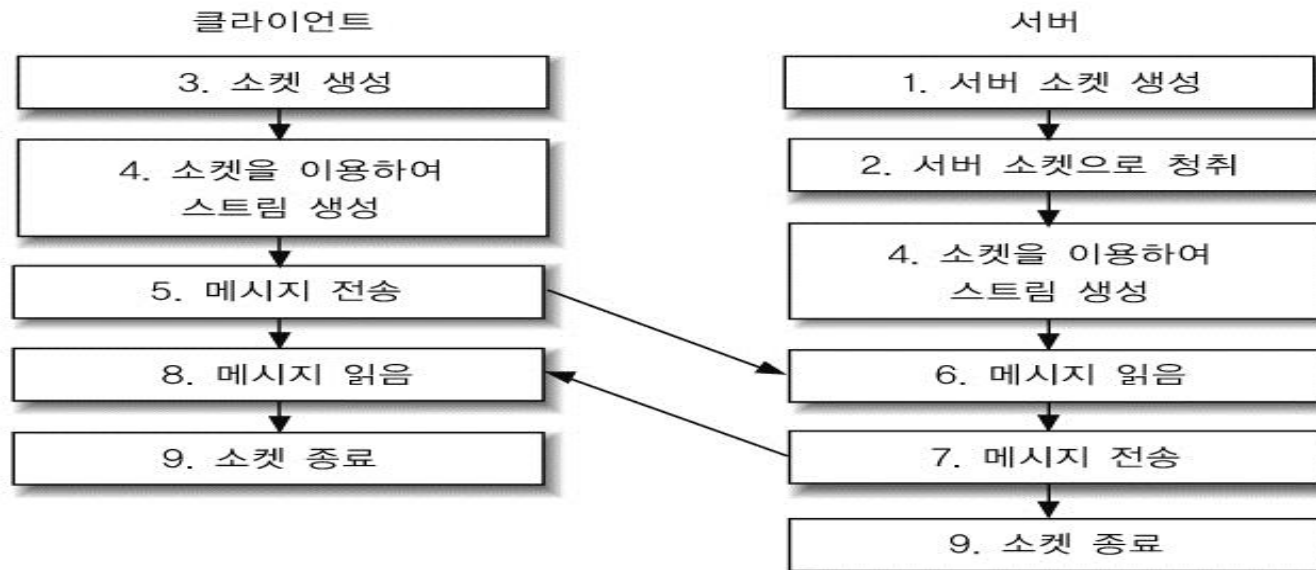
TCP

● ServerSocket 종료

- ➡ Socket class의 종료처럼 ServerSocket의 종료는 close() method로 간단하게 종료할 수 있다.

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

❖ Socket과 ServerSocket을 이용한 간단한 Echo Program



UDP (User Datagram Protocol)

- ➡ UDP는 비 연결 지향이고, IP 위에 매우 얇은 Layer로 구성되어 있다.
- ➡ UDP를 사용하는 Application은 TCP Program에 비해 제어를 할 수 있는 부분이 적다.
- ➡ UDP는 Data를 전송할 때에 Data가 잘 도착했는지 알아낼 방법이 없으며, Data를 보낸 순서대로 도착한다는 보장도 할 수 없다.
- ➡ UDP는 TCP에 비해 훨씬 빠르게 전달된다는 장점이 있다.

❁ DatagramPacket class

- ➡ UDP Datagram은 `java.net.DatagramPacket` class로 추상화한 것이다.
- ➡ `DatagramPacket` class는 Application에서 주고 받을 `Data`와 관련된 class이고, `DatagramSocket` class는 실제 `Data`의 전송을 책임지게 된다.
- ➡ `DatagramPacket` class는 `Data`를 송신하기 위한 기능과 수신을 하기 위한 기능으로 분리된다.

❁ DatagramPacket class의 Constructor

- ➡ DatagramPacket의 Constructor는 Data를 보내기 위한 Constructor와 Data를 받기 위한 Constructor로 구분된다.

※ DatagramPacket 클래스의 주요 생성자

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

❖ DatagramPacket class의 method

- ➡ IP Header에 출발지 Address와 목적지 Address를 설정하거나 Address를 얻어오는 method, 출발지 Port와 목적지 Port를 설정하거나 얻어오는 다양한 method 제공한다.

※ 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.
	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.

UDP

❁ DatagramSocket class

- ➡ TCP Stream Socket과 달리 Server와 Client Datagram Socket 사이에는 차이가 없으며 모든 Datagram Socket은 Datagram을 전송할 뿐만 아니라 수신에서 사용할 수 있다.

❁ DatagramSocket class의 Constructor

- ➡ 모든 DatagramSocket Object는 Data그램을 수신하기 위해서 사용될 수 있기 때문에 local Host 내의 유일한 UDP Port와 연관되어 있다.

※ 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.

UDP

❁ DatagramSocket class의 주요 method

- ➡ DatagramSocket class의 주요 method 기능은 DatagramPacket 을 보내거나 받을 수 있는 method를 제공하는 것이다.

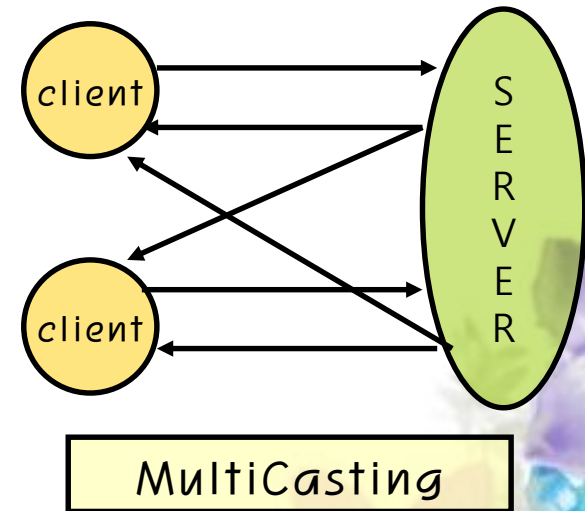
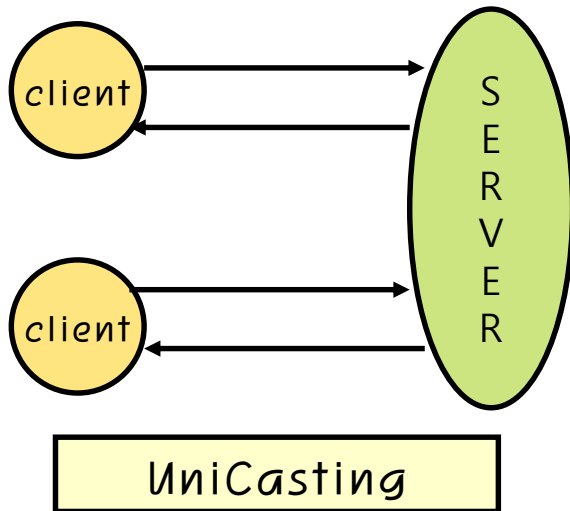
※ DatagramSocket 클래스의 주요 메서드

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.

MultiCasting

UniCasting과 MultiCasting

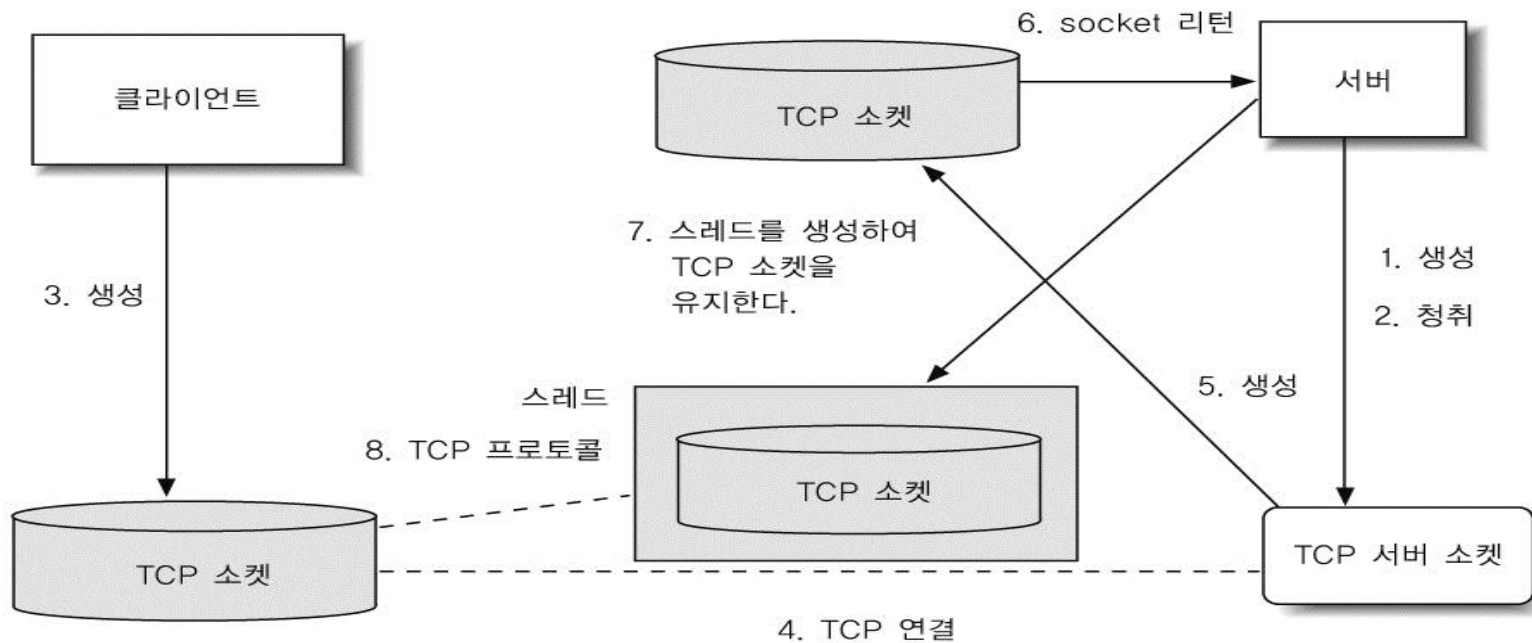
- Client와 Server간의 지속적으로 1 : 1로 통신하는 개념을 UniCasting이라고 한다.
- 1 : 다의 통신을 MultiCasting이라고 한다.



MultiCasting

UniCasting

- UniCasting을 구현하기 위해서는 필수 조건이 Server측에 Thread를 생성해서 TCP Socket을 유지해야 한다.



MultiCasting

● UniCast Program

※ 유니캐스트 예제에 필요한 클래스 구조

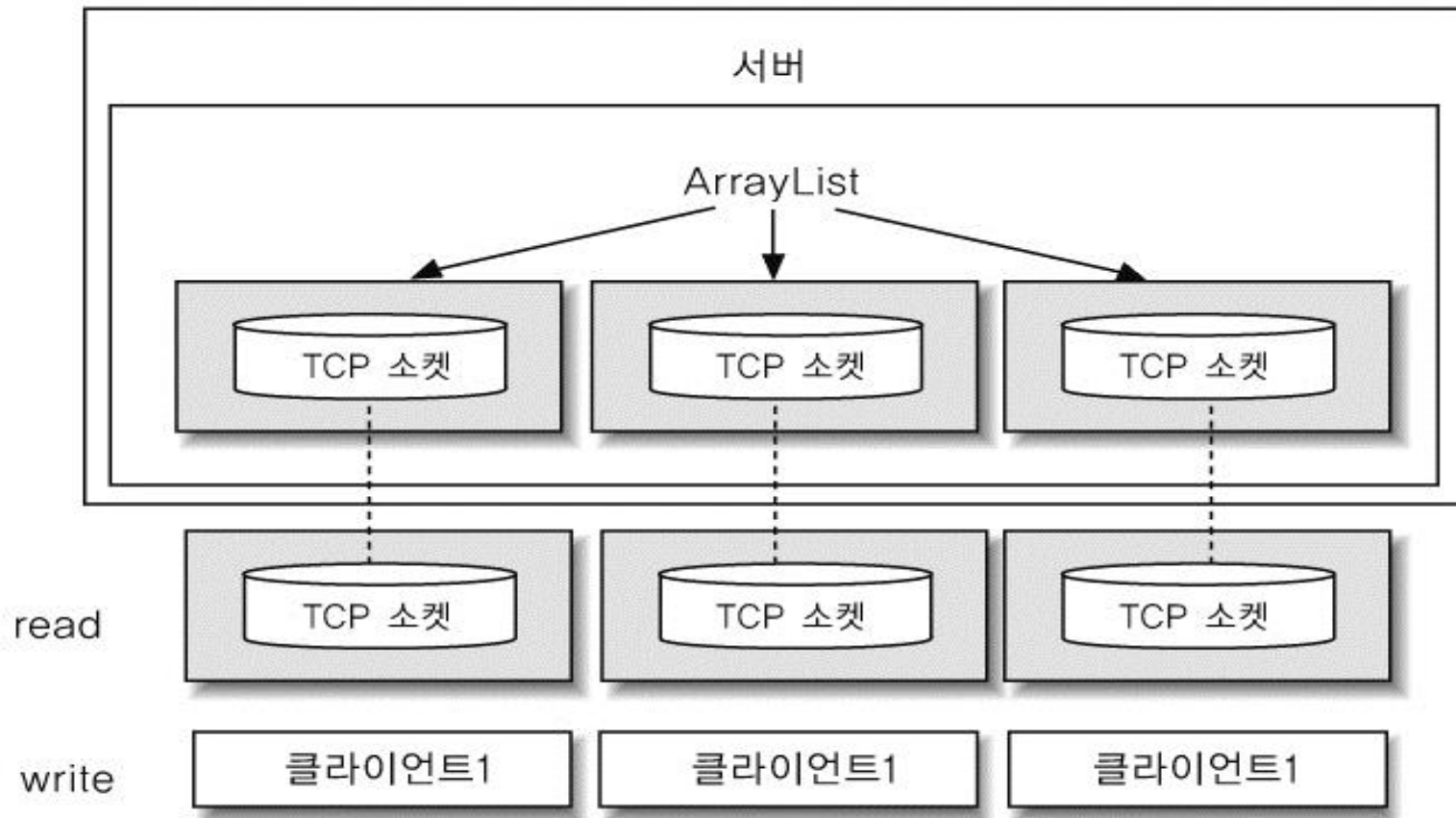
클래스명	설명
UnicastServer	모든 클라이언트의 TCP 요청을 받아 Socket 객체를 생성하고, Socket을 유지하기 위한 스레드를 생성하는 클래스다.
UnicastServerThread	각각의 클라이언트의 Socket 객체를 유지하기 위한 클래스다.
UnicastClient	클라이언트가 콘솔에서 접속하기 위한 클래스다.

MultiCasting

MultiCasting

- ▶ UniCast Model은 실시간 Program에서 Server의 정보를 모든 Client가 공유할 때 문제점이 있다.
- ▶ 이런 문제를 해결할 수 있는 방법이 1 : 다 전송을 지원하는 MultiCasting 방법이다.
- ▶ 한명의 Client가 Server의 정보를 변경했을 경우 모든 Client에게 전송함으로써 서로가 변경된 정보를 공유할 수 있는 Application을 만들 때 적합하다.
- ▶ MultiCasting Program을 작성하기 위해서는 UniCast에서 생성된 Thread를 저장하기 위한 공간(ArrayList)이 필요하며, Client에서는 자신이 보낸 Message나 다른 Client가 보낸 Message를 받기 위한 Thread가 필요하다.

MultiCasting



MultiCasting

● MultiCast Program

※ 멀티캐스트 프로그램에 필요한 클래스 구조

생성자	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.

01. TCP/IP 프로토콜에 대하여

TCP/IP 프로토콜

- 인터넷에서 사용되는 프로토콜(protocol, 통신 규칙)
- TCP 프로토콜과 IP 프로토콜을 함께 부르는 이름

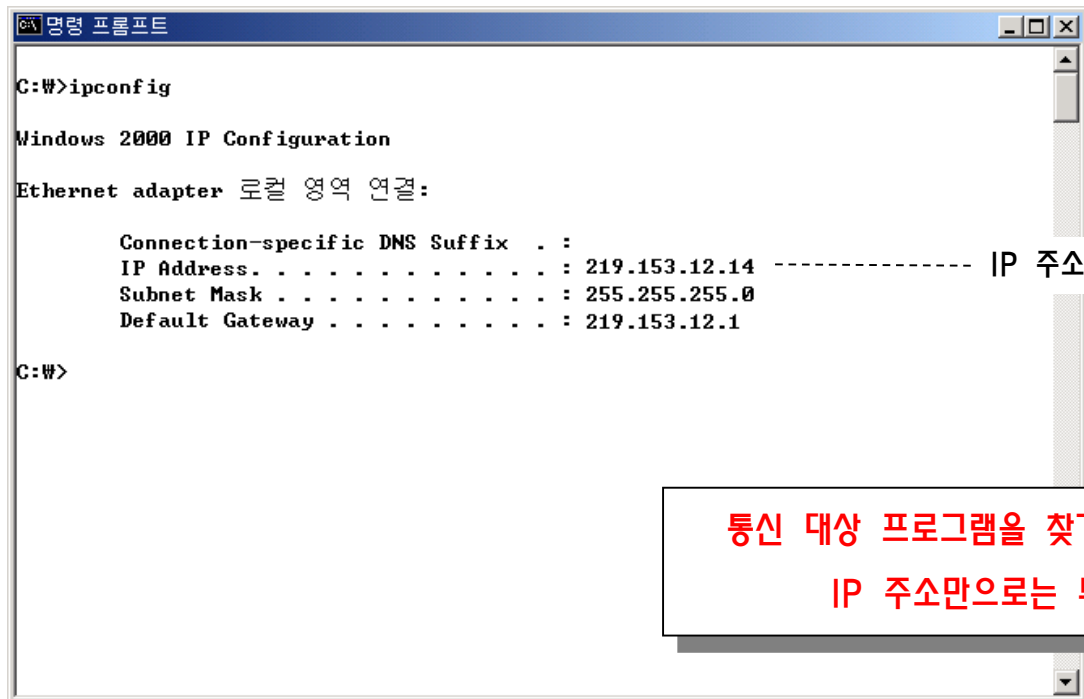
↑
네트워크에 연결된 각각의 컴퓨터에
IP 주소(IP address)를 붙여서 관리하는 규칙

네트워크 통신 프로그래밍

01. TCP/IP 프로토콜에 대하여

TCP/IP 프로토콜

- 컴퓨터의 IP 주소 확인하는 방법



```
C:\W>ipconfig

Windows 2000 IP Configuration

Ethernet adapter 로컬 영역 연결:

    Connection-specific DNS Suffix  . : 
    IP Address. . . . . : 219.153.12.14 ----- IP 주소
    Subnet Mask . . . . . : 255.255.255.0
    Default Gateway . . . . . : 219.153.12.1

C:\W>
```

통신 대상 프로그램을 찾기 위해서는
IP 주소만으로는 부족

● 01. TCP/IP 프로토콜에 대하여

● TCP/IP 프로토콜

- 포트(port) : 네트워크를 통해 데이터를 주고 받을 때 사용되는 가상의 출구
 - 0 ~ 65535 범위의 정수 번호로 식별됨

네트워크 통신 프로그래밍

01. TCP/IP 프로토콜에 대하여

TCP/IP 프로토콜

- 컴퓨터에서 사용 중인 포트 번호 확인하는 방법

```
C:\>netstat -na

Active Connections

Proto Local Address          Foreign Address         State
TCP   0.0.0.0:135             0.0.0.0:0               LISTENING
TCP   0.0.0.0:445             0.0.0.0:0               LISTENING
UDP   0.0.0.0:445             *:*
```

사용 중인 포트 번호

주의: 0 ~ 1023까지는 시스템이 사용하는 포트번호

01. TCP/IP 프로토콜에 대하여

네트워크 통신 프로그램

• 통신 프로그램의 작성 방법

- 1) 한 프로그램은 연결을 요청하고 다른 프로그램은 그 요청을 받아서 연결을 맺습니다.
- 2) 그 연결을 통해 데이터를 주고 받습니다.

• 용어 설명

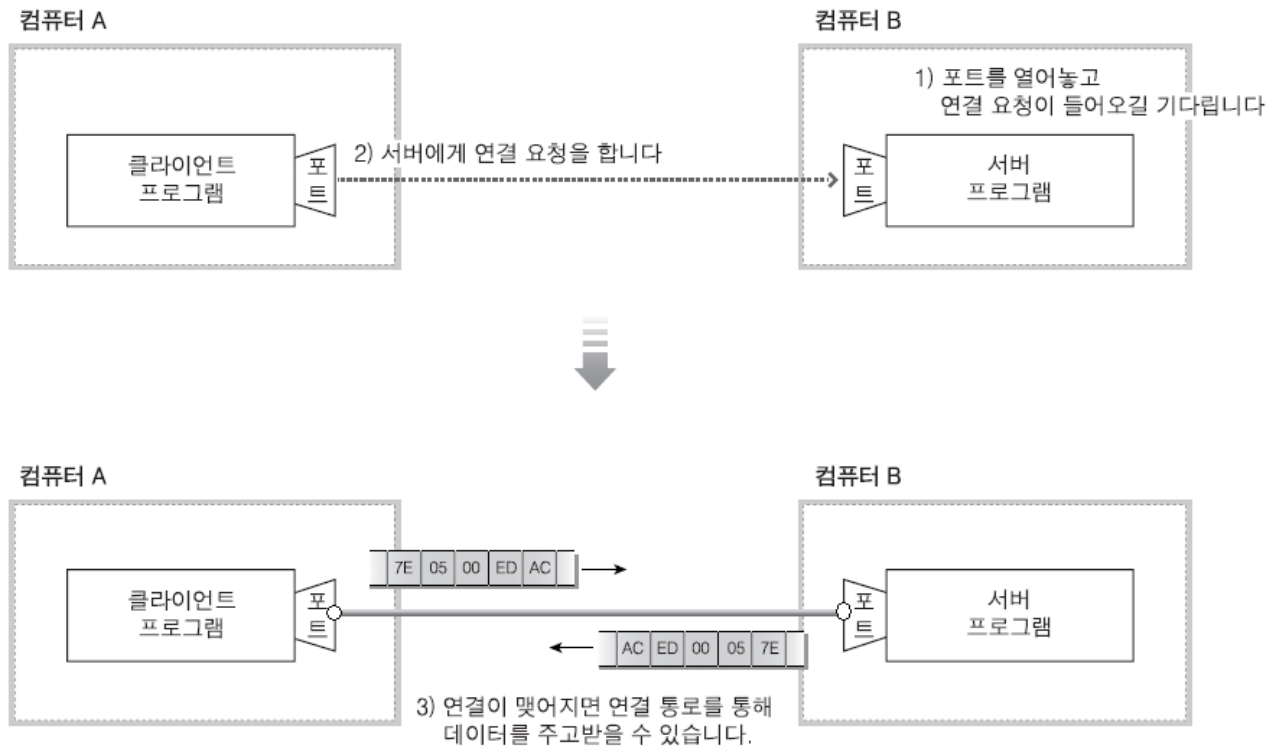
- 클라이언트 프로그램(client program) : 연결을 요청하는 통신 프로그램
- 서버 프로그램(server program) : 연결 요청을 기다리는 통신 프로그램

네트워크 통신 프로그래밍

01. TCP/IP 프로토콜에 대하여

네트워크 통신 프로그램

- 클라이언트 프로그램과 서버 프로그램의 통신 과정



02. TCP/IP 통신 프로그램의 작성 방법

소켓에 대하여

• 소켓(socket) : 프로그램 내에서 보았을 때의 데이터 통신 출입구

- 서버 소켓, 클라이언트 소켓 두 종류

• 서버 소켓

- 서버 프로그램에서만 사용되는 소켓
- 연결 요청을 기다리다가, 연결 요청이 오면 연결을 맺고 또 다른 소켓을 생성

• 클라이언트 소켓

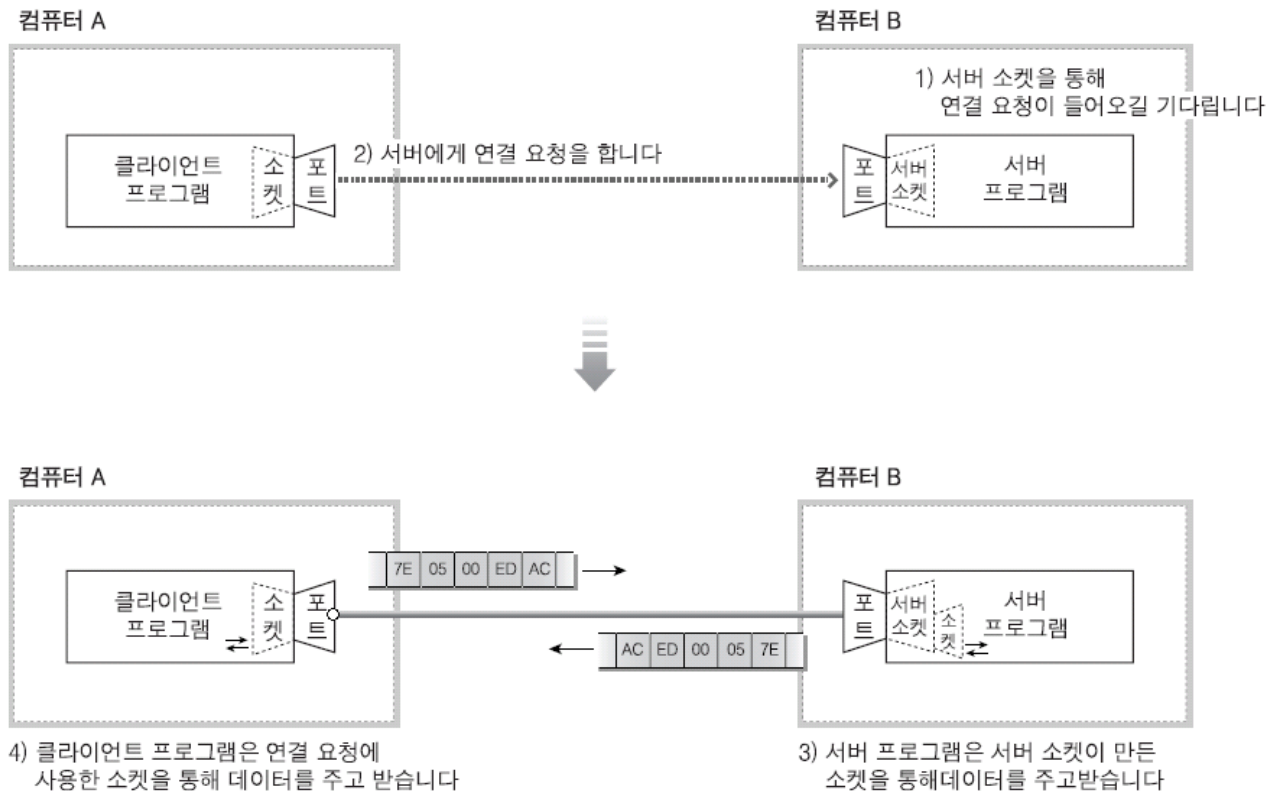
- 클라이언트 프로그램과 서버 프로그램에서 모두 사용되는 소켓
- 실제 데이터 전송에 사용되는 것은 이 소켓임
- 서버 프로그램에서는 서버 소켓에 의해 생성됨
- 클라이언트 프로그램에서는 직접 생성해야 함

네트워크 통신 프로그래밍

02. TCP/IP 통신 프로그램의 작성 방법

소켓을 이용한 통신

• 소켓을 이용한 클라이언트 프로그램과 서버 프로그램의 통신 과정



02. TCP/IP 통신 프로그램의 작성 방법

소켓을 이용한 통신 : 서버 프로그램

• 서버 소켓을 생성하고 사용하는 방법

- 1) `ServerSocket` 객체를 생성합니다.

```
ServerSocket serverSocket = new ServerSocket(9000);
```

↑
포트 번호

02. TCP/IP 통신 프로그램의 작성 방법

소켓을 이용한 통신 : 서버 프로그램

- 서버 소켓을 생성하고 사용하는 방법
 - 2) `ServerSocket` 객체에 대해 `accept` 메소드를 호출합니다.

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

↑
서버 소켓으로 연결 요청이 들어오면 연결을 맺고,
클라이언트 소켓을 생성해서 리턴하는 메소드

02. TCP/IP 통신 프로그램의 작성 방법

소켓을 이용한 통신 : 클라이언트/서버 프로그램

- 클라이언트 소켓을 생성하고 사용하는 방법
 - 1) Socket 객체를 생성합니다.

```
Socket socket = new Socket("219.153.21.14", 9000);
```

↑
서버 프로그램이 있는
컴퓨터의 IP 주소

↑
서버 프로그램이
열어 놓은 포트 번호

네트워크 통신 프로그래밍

02. TCP/IP 통신 프로그램의 작성 방법

소켓을 이용한 통신 : 클라이언트/서버 프로그램

• 클라이언트 소켓을 생성하고 사용하는 방법

- 2) 데이터 송수신에 사용할 입력 스트림 객체와 출력 스트림 객체를 얻습니다.

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

↑
데이터 수신에 사용할
입력 스트림 객체를 리턴하는 메소드

```
OutputStream out = socket.getOutputStream();
```

↑
데이터 송신에 사용할
출력 스트림 객체를 리턴하는 메소드

02. TCP/IP 통신 프로그램의 작성 방법

소켓을 이용한 통신 : 클라이언트/서버 프로그램

• 클라이언트 소켓을 생성하고 사용하는 방법

- 3) write 메소드와 read 메소드를 호출하여 데이터 송신 또는 수신합니다.

```
out.write(data);
```

↑
파라미터로 넘겨준
데이터를 송신합니다

```
int data = in.read();
```

↑
수신된 데이터를
읽어서 리턴합니다

02. TCP/IP 통신 프로그램의 작성 방법

소켓을 이용한 통신 : 클라이언트/서버 프로그램

• 클라이언트 소켓을 생성하고 사용하는 방법

- 4) 클라이언트 소켓을 닫습니다.

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

↑
소켓을 닫는 메소드

02. TCP/IP 통신 프로그램의 작성 방법

소켓을 이용한 통신 : 서버 프로그램

- 서버 소켓도 모두 사용하고 난 후에는 닫아야 합니다.

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

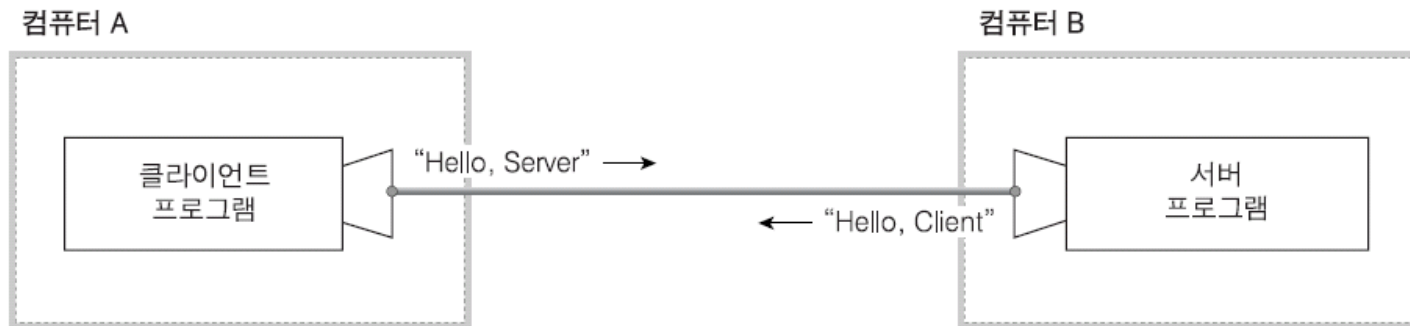
↑
서버 소켓을 닫는 메소드

네트워크 통신 프로그래밍

02. TCP/IP 통신 프로그램의 작성 방법

클라이언트/서버 프로그램 작성

•• 지금부터 작성할 예제가 하는 일



네트워크 통신 프로그래밍

02. TCP/IP 통신 프로그램의 작성 방법

클라이언트/서버 프로그램 작성

- [예제 20-1] TCP/IP로 통신하는 클라이언트 프로그램과 서버 프로그램 (1)

클라이언트 프로그램

```
1 import java.io.*;
2 import java.net.*;
3 class ClientExample1 {
4     public static void main(String[] args) {
5         Socket socket = null;
6         try {
7             socket = new Socket("###.###.###.###", 9000); ----- 소켓을 생성합니다
8             InputStream in = socket.getInputStream();
9             OutputStream out = socket.getOutputStream();
10            String str = "Hello, Server";
11            out.write(str.getBytes()); ----- 데이터를 송신합니다
12            byte arr[] = new byte[100];
13            in.read(arr); ----- 수신된 데이터를 출력합니다
14            System.out.println(new String(arr));
15        }
16        catch (Exception e) {
17            System.out.println(e.getMessage());
18        }
19        finally {
20            try {
21                socket.close(); ----- 소켓을 닫습니다
22            }
23            catch (Exception e) {
24            }
25        }
26    }
27 }
```

서버 프로그램

```
1 import java.io.*;
2 import java.net.*;
3 class ServerExample1 {
4     public static void main(String[] args) {
5         ServerSocket serverSocket = null;
6         Socket socket = null;
7         try {
8             serverSocket = new ServerSocket(9000); ----- 서버 소켓을 생성합니다
9             socket = serverSocket.accept(); ----- 연결 요청이 오면 소켓을 생성합니다
10            InputStream in = socket.getInputStream();
11            OutputStream out = socket.getOutputStream();
12            byte arr[] = new byte[100];
13            in.read(arr); ----- 수신된 데이터를 출력합니다
14            System.out.println(new String(arr));
15            String str = "Hello, Client";
16            out.write(str.getBytes()); ----- 데이터를 송신합니다
17        }
18        catch (Exception e) {
19            System.out.println(e.getMessage());
20        }
21        finally {
22            try {
23                socket.close(); ----- 소켓을 닫습니다
24            }
25            catch (Exception ignored) {
26            }
27            try {
28                serverSocket.close(); ----- 서버 소켓을 닫습니다
29            }
30            catch (Exception ignored) {
31            }
32        }
33    }
34 }
```

네트워크 통신 프로그래밍

02. TCP/IP 통신 프로그램의 작성 방법

클라이언트/서버 프로그램 작성

- [예제 20-1] TCP/IP로 통신하는 클라이언트 프로그램과 서버 프로그램 (1) - 실행 결과

컴퓨터 B

```
컴퓨터 B
C:\명령 프롬프트 - java ServerExample1
E:\work\chap20\20-2-1\example1\server>java ServerExample1
```

1) 서버 프로그램을 먼저 실행합니다.

```
C:\명령 프롬프트
E:\work\chap20\20-2-1\example1\server>java ServerExample1
Hello, Server
E:\work\chap20\20-2-1\example1\server>_
```

3) 서버 프로그램은 클라이언트 프로그램으로부터 전송된 메시지를 출력합니다.

컴퓨터 A

```
컴퓨터 A
C:\명령 프롬프트
E:\work\chap20\20-2-1\example1\client>java ClientExample1_
```

2) 클라이언트 프로그램을 실행합니다.

```
C:\명령 프롬프트
E:\work\chap20\20-2-1\example1\client>java ClientExample1
Hello, Client
E:\work\chap20\20-2-1\example1\client>
```

4) 클라이언트 프로그램도 서버 프로그램으로부터 전송된 메시지를 출력합니다.

네트워크 통신 프로그래밍

02. TCP/IP 통신 프로그램의 작성 방법

클라이언트/서버 프로그램 작성

•• 바이트 스트림을 문자 스트림으로 바꾸는 방법

- - InputStream 객체를 가지고 InputStreamReader 객체를 만들 수 있습니다.

```
InputStreamReader reader = new InputStreamReader(in);
```

↑
InputStream 객체

- - OutputStream 객체를 가지고 PrintWriter 객체를 만들 수 있습니다.

```
PrintWriter writer = new PrintWriter(out);
```

↑
OutputStream 객체

네트워크 통신 프로그래밍

02. TCP/IP 통신 프로그램의 작성 방법

클라이언트/서버 프로그램 작성

- [예제 20-2] TCP/IP로 통신하는 클라이언트 프로그램과 서버 프로그램 (2)

클라이언트 프로그램

```
1 import java.io.*;
2 import java.net.*;
3 class ClientExample2 {
4     public static void main(String[] args) {
5         Socket socket = null;
6         try {
7             socket = new Socket("###.###.###.###", 9000);
8             BufferedReader reader = new BufferedReader(
9                 new InputStreamReader(socket.getInputStream()));
10            PrintWriter writer =
11                new PrintWriter(socket.getOutputStream());
12            writer.println("Hello, Server");
13            String str = reader.readLine();
14            System.out.println(str);
15        } catch (Exception e) {
16            System.out.println(e.getMessage());
17        } finally {
18            try {
19                socket.close();
20            } catch (Exception ignored) {}
21        }
22    }
23 }
```

데이터를 송신합니다

수신된 데이터를 출력합니다

서버 프로그램

```
1 import java.io.*;
2 import java.net.*;
3 class ServerExample2 {
4     public static void main(String[] args) {
5         ServerSocket serverSocket = null;
6         Socket socket = null;
7         try {
8             serverSocket = new ServerSocket(9000);
9             socket = serverSocket.accept();
10            BufferedReader reader = new BufferedReader(
11                new InputStreamReader(socket.getInputStream()));
12            PrintWriter writer =
13                new PrintWriter(socket.getOutputStream());
14            String str = reader.readLine();
15            System.out.println(str);
16            writer.println("Hello, Client");
17            writer.flush();
18        } catch (Exception e) {
19            System.out.println(e.getMessage());
20        } finally {
21            try {
22                socket.close();
23            } catch (Exception ignored) {}
24        }
25        try {
26            serverSocket.close();
27        } catch (Exception ignored) {}
28    }
29 }
```

수신된 데이터를 출력합니다

데이터를 송신합니다

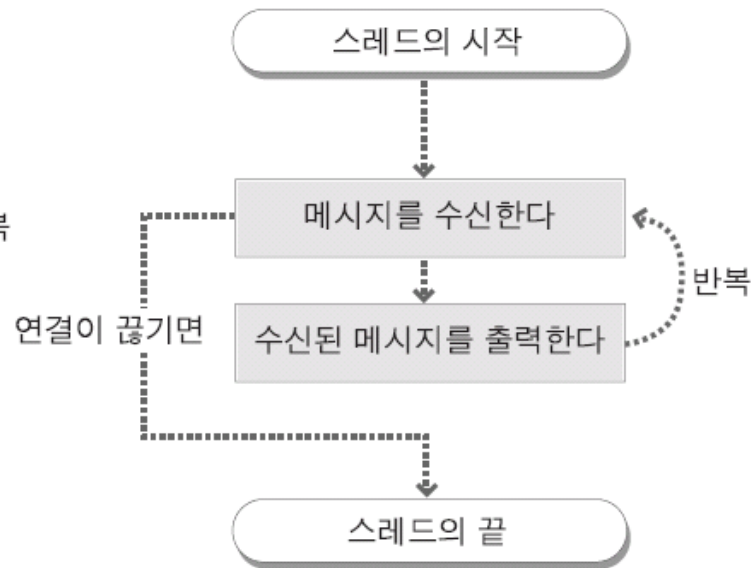
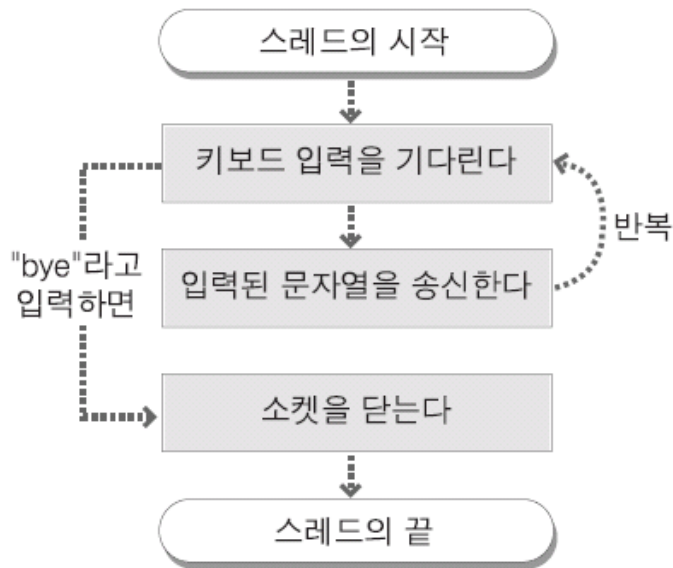
실행 결과는 [예제 20-1]과 동일

네트워크 통신 프로그래밍

02. TCP/IP 통신 프로그램의 작성 방법

송신과 수신을 동시에 하는 프로그램

• 클라이언트 프로그램과 서버 프로그램의 실행 흐름



네트워크 통신 프로그래밍

02. TCP/IP 통신 프로그램의 작성 방법

송신과 수신을 동시에 하는 프로그램

• [예제 20-3] 일대일 채팅 프로그램 – 클라이언트 프로그램

클라이언트 프로그램

```
1 import java.net.*;
2 class ClientExample3 {
3     public static void main(String[] args) {
4         Socket socket = null;
5         try {
6             socket = new Socket("###.###.###.###", 9001);
7             Thread thread1 = new SenderThread(socket);
8             Thread thread2 = new ReceiverThread(socket);
9             thread1.start();
10            thread2.start();
11        }
12        catch (Exception e) {
13            System.out.println(e.getMessage());
14        }
15    }
16 }
```

메시지를 송신하는 스레드 클래스

```
1 import java.io.*;
2 import java.net.*;
3 class SenderThread extends Thread {
4     Socket socket;
5     SenderThread(Socket socket) {
6         this.socket = socket;
7     }
8     public void run() {
9         try {
10             BufferedReader reader = new BufferedReader(
11                 new InputStreamReader(System.in));
12             PrintWriter writer =
13                 new PrintWriter(socket.getOutputStream());
14             while (true) {
15                 String str = reader.readLine();
16                 if (str.equals("bye"))
17                     break;
18                 writer.println(str);
19                 writer.flush();
20             }
21         }
22         catch (Exception e) {
23             System.out.println(e.getMessage());
24         }
25         finally {
26             try {
27                 socket.close();
28             }
29             catch (Exception ignored) {}
30         }
31     }
32 }
```

메시지를 수신하는 스레드 클래스

```
1 import java.io.*;
2 import java.net.*;
3 class ReceiverThread extends Thread {
4     Socket socket;
5     ReceiverThread(Socket socket) {
6         this.socket = socket;
7     }
8     public void run() {
9         try {
10             BufferedReader reader = new BufferedReader(
11                 new InputStreamReader(socket.getInputStream()));
12             while (true) {
13                 String str = reader.readLine();
14                 if (str == null)
15                     break;
16                 System.out.println("수신>" + str);
17             }
18         }
19         catch (Exception e) {
20             System.out.println(e.getMessage());
21         }
22     }
23 }
```

네트워크 통신 프로그래밍

02. TCP/IP 통신 프로그램의 작성 방법

송신과 수신을 동시에 하는 프로그램

• [예제 20-4] 일대일 채팅 프로그램 - 서버 프로그램

클라이언트 프로그램

```
1 import java.net.*;
2 class ServerExample3 {
3     public static void main(String[] args) {
4         ServerSocket serverSocket = null;
5         Socket socket = null;
6         try {
7             serverSocket = new ServerSocket(9001);
8             socket = serverSocket.accept();
9             Thread thread1 = new SenderThread(socket);
10            Thread thread2 = new ReceiverThread(socket);
11            thread1.start();
12            thread2.start();
13        }
14        catch (Exception e) {
15            System.out.println(e.getMessage());
16        }
17        finally {
18            try {
19                serverSocket.close();
20            }
21            catch (Exception ignored) {
22            }
23        }
24    }
25 }
```

메시지를 송신하는 쓰레드 클래스

```
1 import java.io.*;
2 import java.net.*;
3 class SenderThread extends Thread {
4     Socket socket;
5     SenderThread(Socket socket) {
6         this.socket = socket;
7     }
8     public void run() {
9         try {
10            BufferedReader reader = new BufferedReader(
11                new InputStreamReader(System.in));
12            PrintWriter writer =
13                new PrintWriter(socket.getOutputStream());
14            while (true) {
15                String str = reader.readLine();
16                if (str.equals("bye"))
17                    break;
18                writer.println(str);
19                writer.flush();
20            }
21        }
22        catch (Exception e) {
23            System.out.println(e.getMessage());
24        }
25        finally {
26            try {
27                socket.close();
28            }
29            catch (Exception ignored) {
30            }
31        }
32    }
33 }
```

메시지를 수신하는 쓰레드 클래스

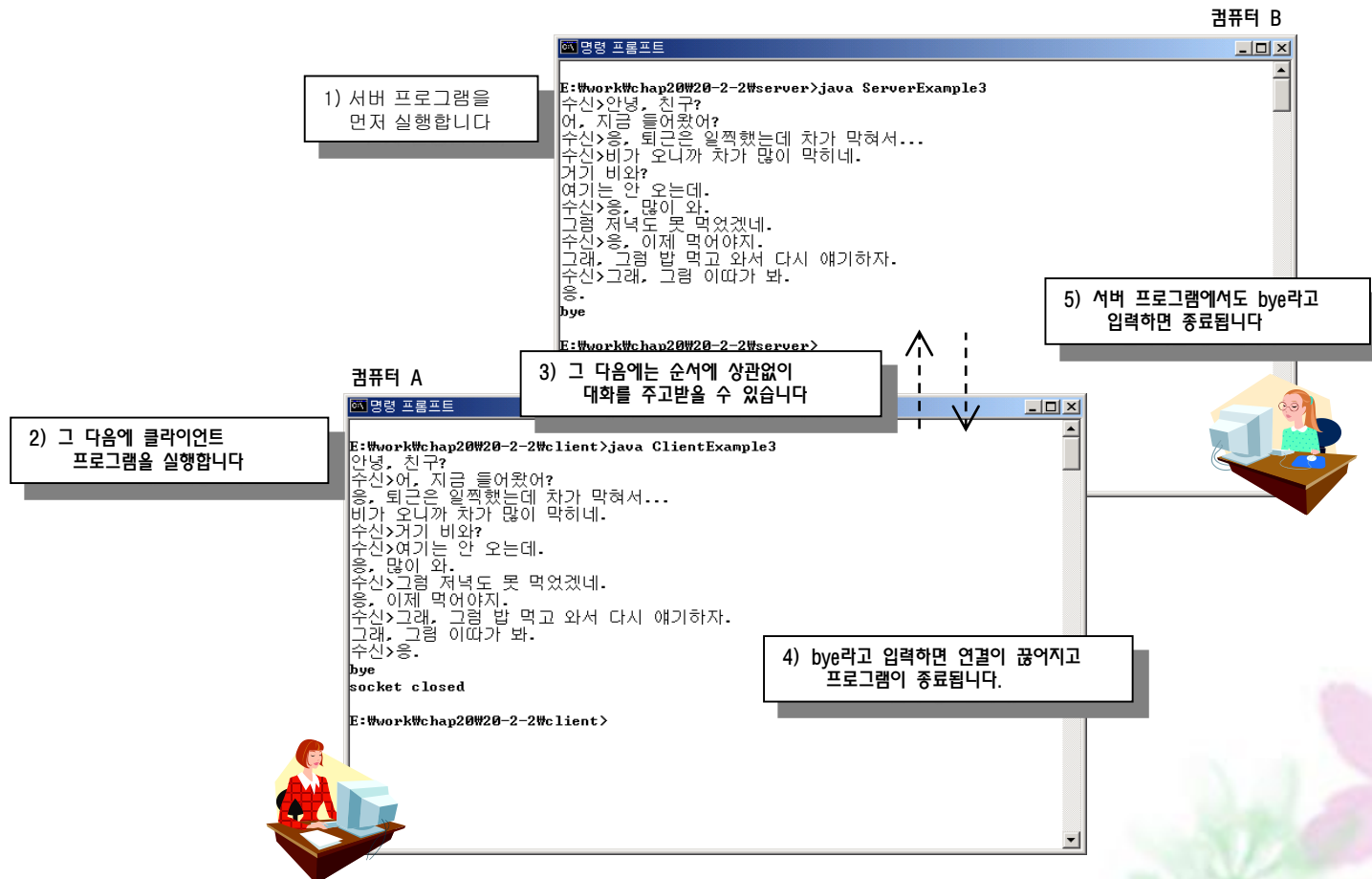
```
1 import java.io.*;
2 import java.net.*;
3 class ReceiverThread extends Thread {
4     Socket socket;
5     ReceiverThread(Socket socket) {
6         this.socket = socket;
7     }
8     public void run() {
9         try {
10            BufferedReader reader = new BufferedReader(
11                new InputStreamReader(socket.getInputStream()));
12            while (true) {
13                String str = reader.readLine();
14                if (str == null)
15                    break;
16                System.out.println("수신>" + str);
17            }
18        }
19        catch (Exception e) {
20            System.out.println(e.getMessage());
21        }
22    }
23 }
```


네트워크 통신 프로그래밍

02. TCP/IP 통신 프로그램의 작성 방법

송신과 수신을 동시에 하는 프로그램

• [예제 20-3,4] 일대일 채팅 프로그램 - 실행 결과

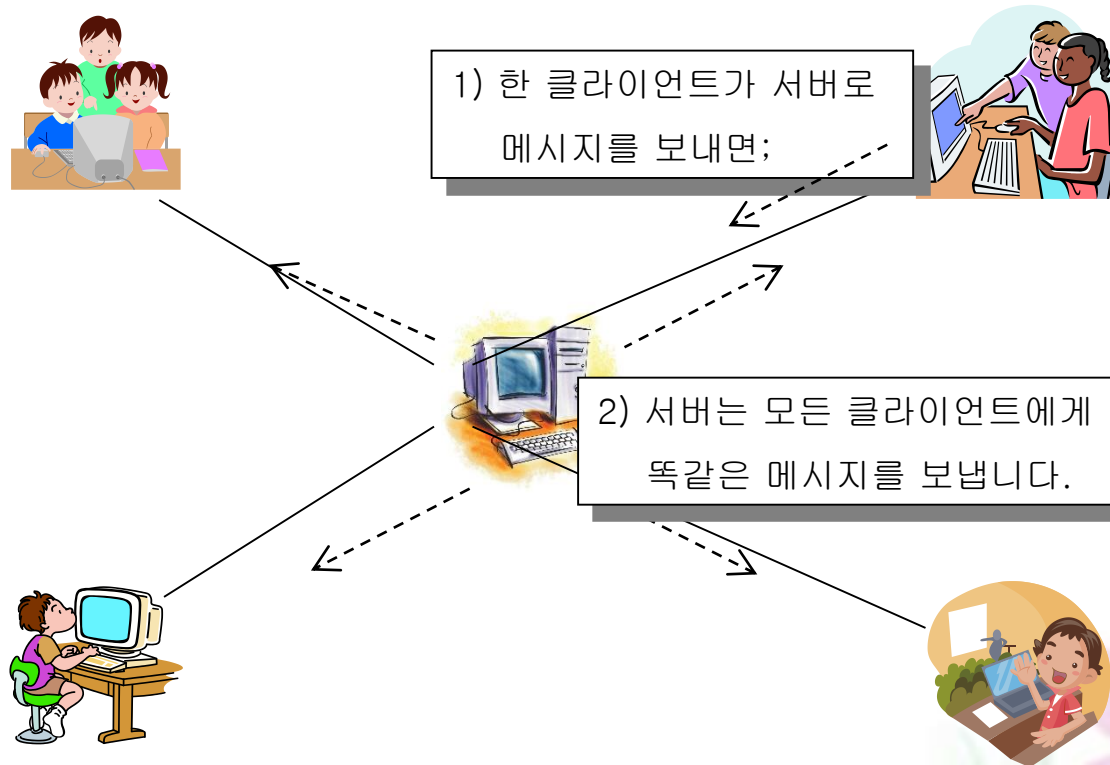


네트워크 통신 프로그래밍

02. TCP/IP 통신 프로그램의 작성 방법

여러 명이 참여하는 채팅 프로그램

• 일반적인 채팅 프로그램의 작동 방식

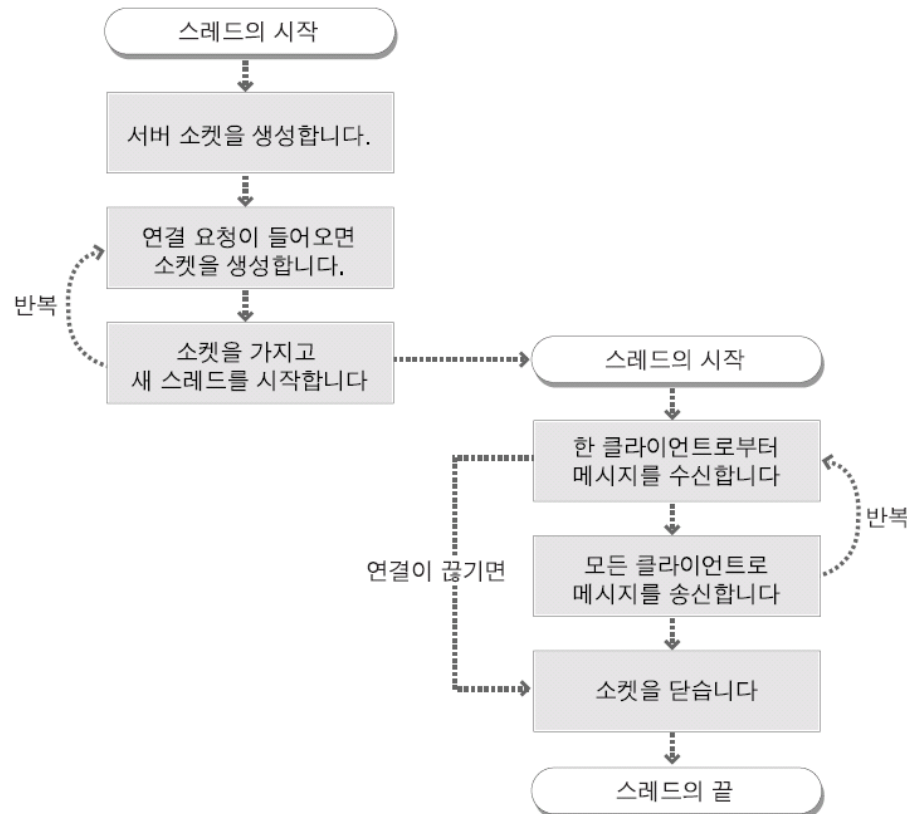


네트워크 통신 프로그래밍

02. TCP/IP 통신 프로그램의 작성 방법

여러 명이 참여하는 채팅 프로그램

•• 여러 명이 참여하는 채팅 프로그램의 실행 흐름



네트워크 통신 프로그래밍

02. TCP/IP 통신 프로그램의 작성 방법

여러 명이 참여하는 채팅 프로그램

- [예제 20-5] 여러 사용자가 함께 채팅하는 프로그램 – 서버 프로그램 (미완성)

클라이언트 프로그램

```
1 import java.io.*;
2 import java.net.*;
3 class ClientExample1 {
4     public static void main(String[] args) {
5         Socket socket = null;
6         try {
7             socket = new Socket("###.###.###.###", 9000); ----- 소켓을 생성합니다
8             InputStream in = socket.getInputStream();
9             OutputStream out = socket.getOutputStream();
10            String str = "Hello, Server";
11            out.write(str.getBytes()); ----- 데이터를 송신합니다
12            byte arr[] = new byte[100];
13            in.read(arr); ----- 수신된 데이터를 출력합니다
14            System.out.println(new String(arr));
15        }
16        catch (Exception e) {
17            System.out.println(e.getMessage());
18        }
19        finally {
20            try {
21                socket.close(); ----- 소켓을 닫습니다
22            }
23            catch (Exception e) {
24            }
25        }
26    }
27 }
```

서버 프로그램

```
1 import java.io.*;
2 import java.net.*;
3 class ServerExample1 {
4     public static void main(String[] args) {
5         ServerSocket serverSocket = null;
6         Socket socket = null;
7         try {
8             serverSocket = new ServerSocket(9000); ----- 서버 소켓을 생성합니다
9             socket = serverSocket.accept(); ----- 연결 요청이 오면 소켓을 생성합니다
10            InputStream in = socket.getInputStream();
11            OutputStream out = socket.getOutputStream();
12            byte arr[] = new byte[100];
13            in.read(arr); ----- 수신된 데이터를 출력합니다
14            System.out.println(new String(arr));
15            String str = "Hello, Client";
16            out.write(str.getBytes()); ----- 데이터를 송신합니다
17        }
18        catch (Exception e) {
19            System.out.println(e.getMessage());
20        }
21        finally {
22            try {
23                socket.close(); ----- 소켓을 닫습니다
24            }
25            catch (Exception ignored) {
26            }
27            try {
28                serverSocket.close(); ----- 서버 소켓을 닫습니다
29            }
30            catch (Exception ignored) {
31            }
32        }
33    }
34 }
```

02. TCP/IP 통신 프로그램의 작성 방법

여러 명이 참여하는 채팅 프로그램

- ArrayList 객체의 멀티 스레드 접근을 안전하게 만드는 방법

```
List list2 = Collections.synchronizedList(list1);
```

↑
동기화된 ArrayList 객체

↑
ArrayList 객체

네트워크 통신 프로그래밍

02. TCP/IP 통신 프로그램의 작성 방법

여러 명이 참여하는 채팅 프로그램

- [예제 20-6] 여러 사용자가 함께 채팅하는 프로그램 – 서버 프로그램 (완성)

클라이언트 프로그램

```
1 import java.io.*;
2 import java.net.*;
3 class ClientExample1 {
4     public static void main(String[] args) {
5         Socket socket = null;
6         try {
7             socket = new Socket("###.###.###.###", 9000); ----- 소켓을 생성합니다
8             InputStream in = socket.getInputStream();
9             OutputStream out = socket.getOutputStream();
10            String str = "Hello, Server";
11            out.write(str.getBytes()); ----- 데이터를 송신합니다
12            byte arr[] = new byte[100];
13            in.read(arr); ----- 수신된 데이터를 출력합니다
14            System.out.println(new String(arr));
15        }
16        catch (Exception e) {
17            System.out.println(e.getMessage());
18        }
19        finally {
20            try {
21                socket.close(); ----- 소켓을 닫습니다
22            }
23            catch (Exception e) {
24            }
25        }
26    }
27 }
```

서버 프로그램

```
1 import java.io.*;
2 import java.net.*;
3 class ServerExample1 {
4     public static void main(String[] args) {
5         ServerSocket serverSocket = null;
6         Socket socket = null;
7         try {
8             serverSocket = new ServerSocket(9000); ----- 서버 소켓을 생성합니다
9             socket = serverSocket.accept(); ----- 연결 요청이 오면 소켓을 생성합니다
10            InputStream in = socket.getInputStream();
11            OutputStream out = socket.getOutputStream();
12            byte arr[] = new byte[100];
13            in.read(arr); ----- 수신된 데이터를 출력합니다
14            System.out.println(new String(arr));
15            String str = "Hello, Client";
16            out.write(str.getBytes()); ----- 데이터를 송신합니다
17        }
18        catch (Exception e) {
19            System.out.println(e.getMessage());
20        }
21        finally {
22            try {
23                socket.close(); ----- 소켓을 닫습니다
24            }
25            catch (Exception ignored) {
26            }
27            try {
28                serverSocket.close(); ----- 서버 소켓을 닫습니다
29            }
30            catch (Exception ignored) {
31            }
32        }
33    }
34 }
```

네트워크 통신 프로그래밍

02. TCP/IP 통신 프로그램의 작성 방법

여러 명이 참여하는 채팅 프로그램

• [예제 20-7] 여러 사용자가 함께 채팅하는 프로그램 – 클라이언트 프로그램

클라이언트 프로그램

```
1 import java.net.*;
2 class ClientExample4 {
3     public static void main(String[] args) {
4         if (args.length != 1) {
5             System.out.println(
6                 "Usage: java ClientExample4 <user-name>");
7             return;
8         }
9         try {
10             Socket socket = new Socket("###.###.###.###",
11                                     9002);
12             Thread thread1 = new SenderThread(socket,
13                                     args[0]);
14             Thread thread2 = new ReceiverThread(socket);
15             thread1.start();
16             thread2.start();
17         }
18         catch (Exception e) {
19             System.out.println(e.getMessage());
20         }
21     }
22 }
```

메시지를 송신하는 쓰레드 클래스

```
1 import java.net.*;
2 import java.io.*;
3 class SenderThread extends Thread {
4     Socket socket;
5     String name;
6     SenderThread(Socket socket, String name) {
7         this.socket = socket;
8         this.name = name;
9     }
10    public void run() {
11        try {
12            BufferedReader reader = new BufferedReader(
13                new InputStreamReader(System.in));
14            PrintWriter writer =
15                new PrintWriter(socket.getOutputStream());
16            writer.println(name);
17            writer.flush();
18            while (true) {
19                String str = reader.readLine();
20                if (str.equals("bye"))
21                    break;
22                writer.println(str);
23                writer.flush();
24            }
25        }
26        catch (Exception e) {
27            System.out.println(e.getMessage());
28        }
29        finally {
30            try {
31                socket.close();
32            }
33            catch (Exception ignored) {}
34        }
35    }
36 }
```

메시지를 수신하는 쓰레드 클래스

```
1 import java.io.*;
2 import java.net.*;
3 class ReceiverThread extends Thread {
4     Socket socket;
5     ReceiverThread(Socket socket) {
6         this.socket = socket;
7     }
8     public void run() {
9         try {
10             BufferedReader reader = new BufferedReader(
11                 new InputStreamReader(socket.getInputStream()));
12             while (true) {
13                 String str = reader.readLine();
14                 if (str == null)
15                     break;
16                 System.out.println(str);
17             }
18         }
19         catch (IOException e) {
20             System.out.println(e.getMessage());
21         }
22     }
23 }
```

네트워크 통신 프로그래밍

02. TCP/IP 통신 프로그램의 작성 방법

여러 명이 참여하는 채팅 프로그램

• [예제 20-6,7] 실행 결과

컴퓨터 A

```
E:\work\chap20\20-2-3\example2\server>java ServerExample4
```

1) 서버 프로그램을 먼저 실행합니다

컴퓨터 C

```
E:\work\chap20\20-2-3\example2\client>java ClientExample4 마술피리
#마술피리님이 들어오셨습니다
안녕하세요?
마술피리>안녕하세요?
채리트리>안녕하세요, 마술피리님
#스칼리님이 들어오셨습니다
스칼리>안녕하세요?
스칼리>제가 너무 늦었죠?
아니요, 저도 지금 막...
마술피리>아니요, 저도 지금 막...
채리트리>...
마술피리>말더듬이요?
마술피리>말더듬이요?
오해를 못 한다고 전해달라고 했는데...
#왜요?
아근이래요...
그럼 우리끼리 얘기해서 결정한 다음에 말더듬이께는 메일로 보내기로 하죠.
```

3) 마찬가지로 다른 컴퓨터에서 클라이언트 프로그램을 실행합니다

컴퓨터 R

```
E:\work\chap20\20-2-3\example2\client>java ClientExample4 채리트리
#채리트리님이 들어오셨습니다
안녕하세요?
채리트리>안녕하세요?
아무도 없나요?
채리트리>아무도 없나요?
...
채리트리>...
#마술피리님이 들어오셨습니다
마술피리>안녕하세요?
안녕하세요, 마술피리님
채리트리>안녕하세요, 마술피리님
#스칼리님이 들어오셨습니다
스칼리>안녕하세요?
스칼리>제가 너무 늦었죠?
마술피리>아니요, 저도 지금 막...
채리트리>...
마술피리>말더듬이요?
스칼리>오해를 못 한다고 전해달라고 했는데...
마술피리>왜요?
스칼리>아근이래요...
그럼 우리끼리 얘기해서 결정한 다음에 말더듬이께는 메일로 보내기로 하죠.
채리트리>그럼 우리끼리 얘기해서 결정한 다음에 말더듬이께는 메일로 보내기로 하죠.
```

2) 닉네임을 명령행 파라미터로 사용하여 클라이언트 프로그램을 시작합니다

컴퓨터 D

```
E:\work\chap20\20-2-3\example2\client>java ClientExample4 스칼리
#스칼리님이 들어오셨습니다
안녕하세요?
스칼리>안녕하세요?
제가 너무 늦었죠?
마술피리>아니요, 저도 지금 막...
채리트리>...
마술피리>말더듬이요?
오해를 못 한다고 전해달라고 했는데...
마술피리>왜요?
아근이래요...
스칼리>아근이래요...
그럼 우리끼리 얘기해서 결정한 다음에 말더듬이께는 메일로 보내기로 하죠.
```

4) 또 다른 컴퓨터에서도 같은 방법으로 클라이언트 프로그램을 실행합니다



정기

이오

환기

시

다

^^!