

first Java

1

CHAPTER 15.

네트워크 프로그래밍

목차

15.1 네트워크 기초

15.2 InetAddress 클래스

15.3 URL

15.4 Socket Programming

15.1 네트워크 기초



JAVA의 네트워크

- 네트워킹이란 다양한 단말기들을 통해서 서로 데이터를 주고받는 작업.
- 네트워크 프로그램은 둘 이상의 단말기(컴퓨터)를 서로 연결해서 데이터의 전송과 수신할 수 있도록 하는 프로그램. 채팅 프로그램이 대표적인 예.
- JAVA 네트워킹은 리소스를 공유할 수 있도록 두 개 이상의 컴퓨팅장치를 함께 연결하는 개념으로 설계.
- JAVA API는 네트워크 통신에 필요한 다양한 라이브러리들을 제공하는데 그 중 JAVA의 소켓 프로그래밍은 서로 다른 단말기(컴퓨터)간에 데이터를 공유하는 기능을 제공.
- java.net 패키지는 JAVA의 네트워크 응용프로그램을 개발하기 위한 많은 클래스를 제공.

15.1 네트워크 기초



TCP/IP 4계층

애플리케이션 계층

전송 계층 : TCP, UDP

인터넷 계층

네트워크 인터페이스

15.1 네트워크 기초



TCP/IP 4계층

- 네트워크 인터페이스 계층
 - 네트워크 인터페이스가 물리적인 연결을 통해 로컬 네트워크 및 외부 IP 데이터그램을
 - 전송하는 방법을 제공.
 - TCP/IP 패킷을 네트워크 매체로 전달하는 것과 네트워크 매체에서 TCP/IP 패킷을 받아들이는 과정을 담당.
- 인터넷 계층(Internet)
 - 어드레싱(addressing), 패키징(packaging), 라우팅(routing) 기능을 제공.
 - 논리적 주소인 IP를 이용한 노드간 전송과 라우팅 기능을 처리.
 - 네트워크상 최종 목적지까지 정확하게 연결되도록 연결성을 제공.

15.1 네트워크 기초



TCP/IP 4계층

- 전송계층(Transport)
 - 데이터의 송수신을 담당.
 - 애플리케이션 계층의 세션과 데이터그램(datagram) 통신서비스 제공.
 - 핵심 프로토콜로 TCP/UDP가 있음.

- **TCP**

1:1 연결 지향, 신뢰할 수 있는 통신 서비스 제공.

TCP 연결 확립과 보내진 패킷의 확인, 순서화, 전달 중 손상된 패킷을 복구하는 역할을 함.

- **UDP**

1:1, 1:N의 비 연결 지향, 신뢰할 수 없는 통신서비스 제공.

전달해야 할 데이터의 크기(하나의 패킷으로 보낼 수 있는 데이터와 같은 경우)가 작을 때 사용.

TCP 연결 확립에 의한 부하를 피하려고 할 때 사용.

15.1 네트워크 기초



TCP/IP 4계층

- 전송계층(Transport)

- 데이터의 송수신을 담당.
- 애플리케이션 계층의 세션과 데이터그램(datagram) 통신서비스 제공.
- 핵심 프로토콜로 TCP/UDP가 있음.

- **TCP**

1:1 연결 지향, 신뢰할 수 있는 통신 서비스 제공.

TCP 연결 확립과 보내진 패킷의 확인, 순서화, 전달 중 손상된 패킷을 복구하는 역할.

- **UDP**

1:1, 1:N의 비 연결 지향, 신뢰할 수 없는 통신서비스 제공.

전달해야 할 데이터의 크기가 작을 때 사용.

TCP 연결 확립에 의한 부하를 피하려고 할 때 사용.

15.1 네트워크 기초



TCP/IP 4계층

- 응용프로그램 계층(Application)
 - 다른 계층의 서비스에 접근할 수 있게 하는 애플리케이션을 제공하는 계층.
 - 애플리케이션들이 데이터를 교환하기 위해 사용하는 프로토콜들을 정의하고 있는데 아래와 같은 프로토콜이 정의되어 있음.
 - **HTTP(HyperText transfer Protocol)**
WWW의 Web 페이지 파일을 전송하는데 사용되는 프로토콜.
 - **FTP(File transfer Protocol)**
상호 파일 전송을 위해 사용되는데 사용되는 프로토콜.
 - **SMTP(Simple Mail transfer Protocol)**
메일 메시지와 그에 추가된 첨부 파일을 전송하기 위해 사용되는 프로토콜.
 - **Telnet(Terminal emulation protocol)**
네트워크 호스트에 원격 접속하기 위해 사용되는 프로토콜.
 - **DNS(Domain Name System)**
호스트 이름을 IP 주소를 변환하기 위해 사용되는 프로토콜.

15.2 InetAddress 클래스

InetAddress 클래스

- java.net.InetAddress 클래스는 IP 주소를 저장하는데 호스트 이름으로 IP 주소를 얻을 수 있는 기능을 제공.

메소드	설명
<code>public static InetAddress getByName(String host) throws UnknownHostException</code>	전달받은 호스트 주소의 문자열로 IP와 이름을 포함하는 InetAddress 의 인스턴스를 반환합니다.
<code>public static InetAddress getLocalHost() throws UnknownHostException</code>	로컬 호스트 이름과 주소가 포함 된 InetAddress 인스턴스를 반환합니다. 만약 방화벽으로 가려진 경우 127.0.0.1을 반환합니다.
<code>public String getHostName()</code>	IP 주소의 호스트 이름을 반환합니다.
<code>public String getHostAddress()</code>	IP 주소를 문자열 형식으로 반환합니다.

15.2 InetAddress 클래스

10



InetAddress 클래스

```
package chapter15;

import java.net.InetAddress;

public class InetAddressTest {

    public static void main(String[] args) {


        String urlStr = "www.yundu.co.kr";

        try {
            InetAddress ip = InetAddress.getByName(urlStr);

            System.out.println("Host Name: " + ip.getHostName());
            System.out.println("IP Address: " + ip.getHostAddress());
        } catch (Exception e) {
            System.out.println(e);
        }

    }

}
```

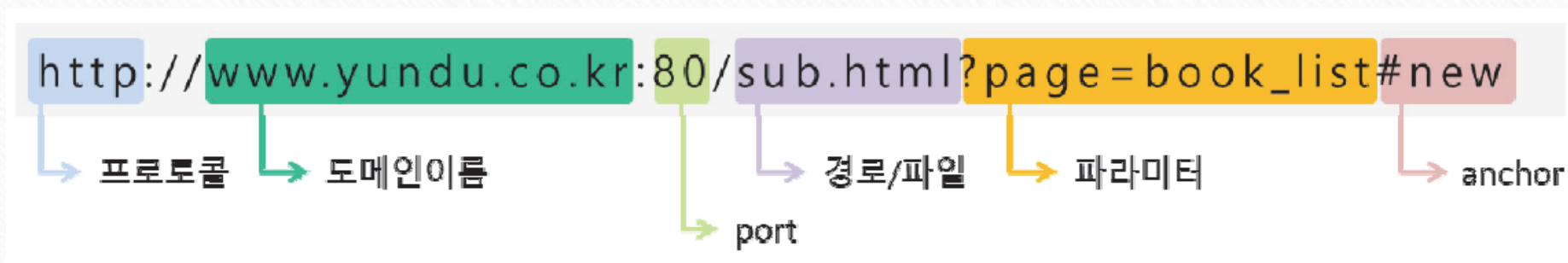
 결과(Console)

```
Host Name: www.naver.com
IP Address: 125.209.222.142
```


15.3 URL

URL

- URL(Uniform Resource Locator)은 인터넷 상의 자원에 대한 주소.
- URL 에는 프로토콜의 종류, 도메인 이름, 포트, 경로명, 파일이름, 쿼리 스트링과 같은 정보를 가지고 있음.



15.3 URL

URL

- http
 - http는 프로토콜(규약) 으로 브라우저가 어떤 통신 규약(약속)을 사용해야 하는지를 나타내는데 보통 웹사이트들을 HTTP 프로토콜이나 HTTPS 프로토콜을 사용.
 - 경우에 따라 메일 전송을 위한 mailto: 또는 파일을 전송하기 위해 ftp: 와 같은 다른 프로토콜을 사용.
- 도메인이름
 - www.yundu.co.kr은 도메인이름으로 웹 서버의 주소를 가리키는데 직접 IP address를 사용하는 것도 가능하지만 일반적으로 도메인이름을 사용해서 웹 서버에 접속.
- port
 - :80은 포트라고 하는데 웹 서버의 자원에 접근하기 위해 사용하는 "관문(gate)"역할을 함.
 - 웹 서버의 자원의 접근하기 위해 표준 HTTP 포트(80) 또는 HTTPS를 사용한다면 포트(443) 포트번호는 보통 생략하고 다른 포트의 경우에는 필수적으로 작성해주어 야함.

15.3 URL

URL

- 경로/파일
 - /sub.html은 웹 서버의 자원에 대한 경로로 초창기에는 html 문서의 물리적인 위치를 나타냈지만, 최근에는 동적 페이지의 주소를 가리킴.
- 파라미터
 - page=book_list 는 웹 서버에 요청하는 추가 데이터.
 - 파라미터들은& 기호로 구분된 키/값으로 짝을 이룬 리스트.
 - 웹 서버는 동적으로 자원을 반환하기 전에 이런 파라미터들을 받아 응답처리.
- anchor
 - #new 는 html 문서 내의 특정 위치에 대한 anchor(닻)로 웹 페이지 안에서 "bookmark"의 역할을 함.

15.3 URL



URL 클래스

- JAVA의 URL클래스는 URL정보를 저장하고 표현하는데 사용.
- 월드 와이드 웹의 리소스를 가리킴.

`http://www.yundu.co.kr/index.html`

- URL 클래스는 아래와 같은 정보를 포함 하고 있음.
 - 프로토콜 : 위의 URL에서는 http가 프로토콜.
 - 서버 이름 또는 IP 주소 : www.yundu.co.kr가 서버의 도메인이름.
 - 포트 번호 : 웹 서버의 경우 기본 포트는 80을 사용. 생략이 가능.
 - 파일 이름 또는 디렉토리 이름 : 위의 URL에서는 index.html 이 파일 이름.

15.3 URL



URL 클래스

- URL 클래스의 생성자

생성자	설명
<code>URL(String protocol, String host, int port, String file)</code>	매개변수의 프로토콜, 호스트, 포트 번호 및 파일 정보를 기반으로 URL 인스턴스를 생성합니다.
<code>URL(String protocol, String host, int port, String file, URLStreamHandler handler)</code>	매개변수의 프로토콜, 호스트, 포트 번호, 파일 및 <code>URLStreamHandler</code> 인스턴스를 기반으로 URL 인스턴스를 생성합니다.
<code>URL(String protocol, String host, String file)</code>	매개변수의 프로토콜 이름, 호스트 이름 및 파일 이름으로 URL 인스턴스를 생성합니다.
<code>URL(URL context, String spec)</code>	매개변수의 컨텍스트 내에서 지정된 스펙을 구문 분석하여 URL 인스턴스를 생성합니다.
<code>URL(URL context, String spec, URLStreamHandler handler)</code>	매개변수의 컨텍스트 내에서 지정된 스펙을 구문 분석하고, 지정된 핸들러를 사용하여 URL 인스턴스를 생성합니다.

15.3 URL



URL 클래스

- URL 클래스의 메소드

메소드	설명
<code>public String getProtocol()</code>	URL의 프로토콜을 반환합니다.
<code>public String getHost()</code>	URL의 호스트 이름을 반환합니다.
<code>public String getPort()</code>	URL의 포트 번호를 반환합니다.
<code>public String getFile()</code>	URL의 파일 이름을 반환합니다.
<code>public String getAuthority()</code>	URL의 권한을 반환합니다.
<code>public String toString()</code>	URL의 문자열 표현을 반환합니다.
<code>public String getQuery()</code>	URL의 쿼리 문자열을 반환합니다.
<code>public String getDefaultPort()</code>	URL의 기본 포트를 반환합니다.
<code>public URLConnection openConnection()</code>	URLConnection의 인스턴스를 반환합니다.
<code>public boolean equals(Object obj)</code>	주어진 객체와 URL을 비교합니다.
<code>public Object getContent()</code>	URL의 내용을 반환합니다.
<code>public String getRef()</code>	URL의 앵커 또는 참조를 반환합니다.
<code>public URI toURI()</code>	URL의 URI를 반환합니다.

15.3 URL

17



URL 클래스

```
package chapter15;

import java.net.URL;

public class URLTest {
    public static void main(String[] args) {
        String urlStr =
            "https://news.naver.com/main/read.nhn?mode=LS2D&mid=sec&sid1=105&sid2=228&oid=584&aid=0000008620";
        try {
            URL url = new URL(urlStr);
            System.out.println("Protocol: " + url.getProtocol());
            System.out.println("Host Name: " + url.getHost());
            System.out.println("Port Number: " + url.getPort());
            System.out.println("Default Port Number: " + url.getDefaultPort());
            System.out.println("Query String: " + url.getQuery());
            System.out.println("Path: " + url.getPath());
            System.out.println("File: " + url.getFile());

        } catch (Exception e) {
            System.out.println(e);
        }
    }
}
```

결과(Console)

```
Protocol: https
Host Name: news.naver.com
Port Number: -1
Default Port Number: 443
Query String: mode=LS2D&mid=sec&sid1=105&sid2=228&oid=584&aid=0000008620
Path: /main/read.nhn
File: /main/read.nhn?mode=LS2D&mid=sec&sid1=105&sid2=228&oid=584&aid=0000008620
```

15.3 URL

18



URLConnection 클래스

- URLConnection 클래스는 URL과 응용 프로그램 사이의 통신 링크를 나타냄.
- URLConnection 클래스는 URL이 참조하는 리소스에서 데이터를 읽고 쓰는데 사용할 수 있음.
- URL 클래스의 openConnection() 메소드는 URLConnection 클래스의 객체를 반환.

```
public URLConnection openConnection() throws IOException { }
```

- URLConnection 클래스는 많은 메소드를 제공하며 getInputStream() 메소드를 사용하여 서버에서 응답의 결과로 반환하는 웹 페이지의 모든 데이터를 표시할 수 있음.
- getInputStream() 메소드는 읽고 표시할 수 있는 스트림에 지정된 URL의 모든 데이터를 반환.

15.3 URL

19



URLConnection 클래스


```
package chapter15;

import java.io.InputStream;
import java.net.URL;
import java.net.URLConnection;

public class URLConnectionTest {

    public static void main(String[] args) {
        String urlStr = "https://www.google.com";
        try {
            URL url = new URL(urlStr);
            URLConnection urlcon = url.openConnection();
            InputStream stream = urlcon.getInputStream();
            int i;
            while ((i = stream.read()) != -1) {
                System.out.print((char) i);
            }
        } catch (Exception e) {
            System.out.println(e);
        }
    }
}
```

**URLConnection 인스턴스는 URL
인스턴스의 openConnection()
메소드를 이용해서 얻을 수 있음**

 결과(Console)

```
<!doctype html><html itemscope=""
  itemtype="http://schema.org/WebPage" lang="ko"><head><meta
  content="text/html; charset=UTF-8" http-equiv="Content-Type">
... 이하 생략
```

15.3 URL



URLConnection 클래스

- HTTP 프로토콜에서만 처리되는 URLConnection 클래스.
- 헤더 정보, 상태 코드, 응답 코드 등과 같은 모든 HTTP URL 정보를 얻을 수 있음.
- URLConnection 클래스를 상속해서 구현한 서브 클래스이기 때문에 HttpURLConnection 클래스의 인스턴스를 얻기 위해서는 URL클래스의 openConnection() 메소드를 호출하면 URLConnection클래스의 인스턴스를 받아 사용할 수 있음.

```
public URLConnection openConnection()throws IOException{
```



```
URL url = new URL("https://news.naver.com/main/read.nhn" );  
  
HttpURLConnection huc = (HttpURLConnection) url.openConnection ();
```


15.3 URL

21



URLConnection 클래스

```
package chapter15;

import java.net.HttpURLConnection;
import java.net.URL;

public class HttpURLConnectionTest {
    public static void main(String[] args) {
        String urlStr =
            "https://news.naver.com/main/read.nhn?mode=LS2D&mid=sec&sid1=105&sid2=228&oid=584&aid=0000008620";
        try {
            URL url = new URL(urlStr);
            // HttpURLConnection 인스턴스도 URL 인스턴스의 openConnection() 메소드로 얻을 수 있음
            HttpURLConnection huc = (HttpURLConnection) url.openConnection();
            for (int i = 1; i <= 8; i++) {
                System.out.println(huc.getHeaderFieldKey(i) + " = " + huc.getHeaderField(i));
            }
            huc.disconnect();
        } catch (Exception e) {
            System.out.println(e);
        }
    }
}
```

결과(Console)

```
date = Sat, 02 May 2020 21:39:04 GMT
cache-control = no-cache
expires = Thu, 01 Jan 1970 00:00:00 GMT
set-cookie = JSESSIONID=376D3F75FA551E51B0E8D4C66941804C; Path=/main;
    HttpOnly
content-language = ko-KR
vary = Accept-Encoding
transfer-encoding = chunked
content-type = text/html; charset=EUC-KR
```

15.4 Socket Programming

22



Socket Programming

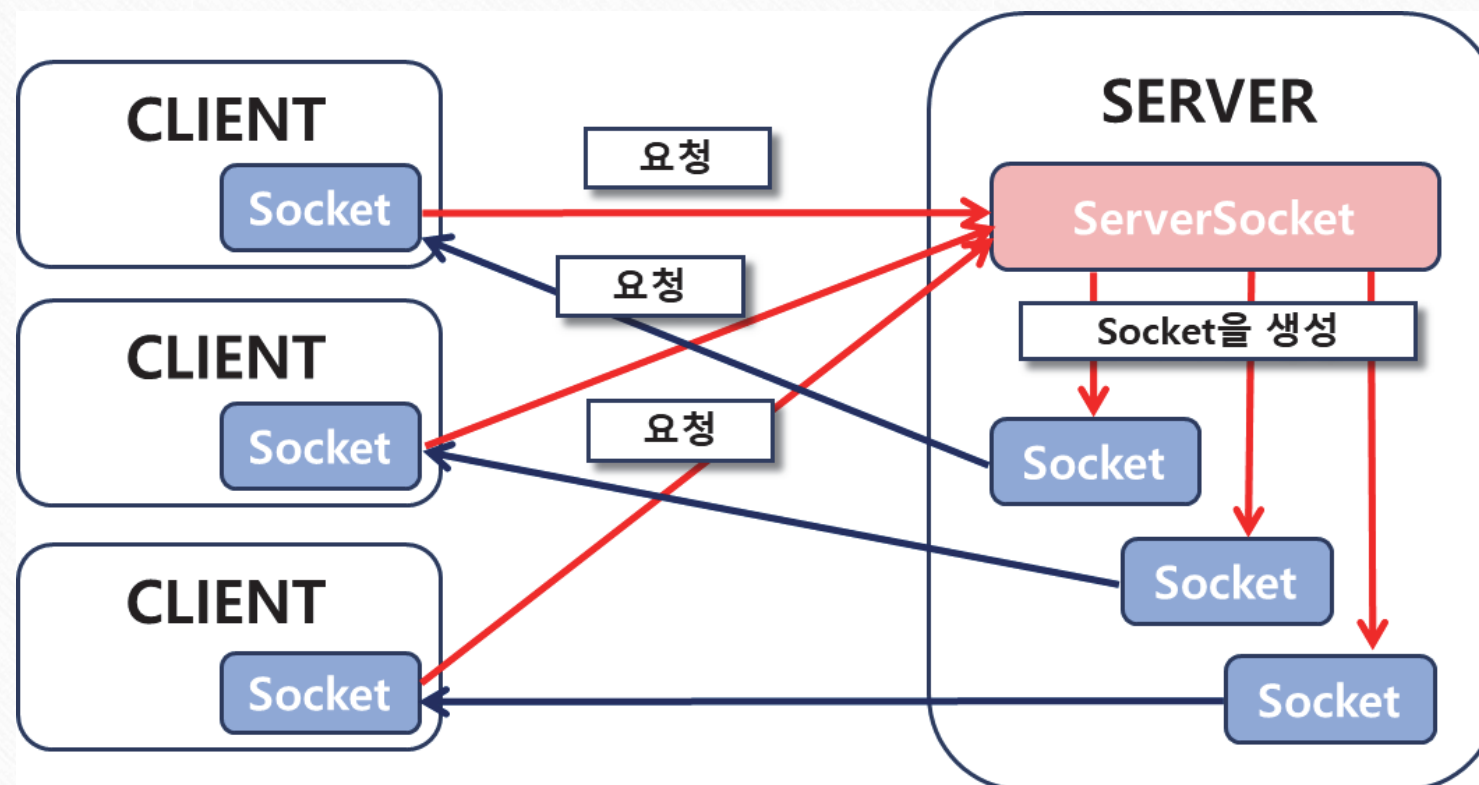
- JAVA 소켓 프로그래밍은 다른 JRE에서 실행되는 응용 프로그램 간의 통신에 사용.
- Socket 클래스 및 ServerSocket 클래스는 연결 지향 소켓 프로그래밍에 사용.
- 소켓 프로그래밍의 클라이언트는 서버의 IP 주소, 포트 번호 두 가지 정보를 알아야 통신을 할 수 있음.
- 프로그램에서 클라이언트는 서버에 메시지를 보내고 서버는 메시지를 읽고 출력.
- 통신을 하기 위해서는 Socket 클래스와 ServerSocket 클래스를 사용.

15.4 Socket Programming



Socket Programming

- Socket 클래스는 클라이언트와 서버 간의 통신할 때 사용하고 ServerSocket 클래스는 서버 측에서 사용.
- ServerSocket 클래스의 accept() 메소드는 클라이언트가 연결될 때까지 대기하고 콘솔을 차단하고 있다가 클라이언트가 성공적으로 연결되면 서버 측에서 Socket 인스턴스를 반환하고 연결 상태를 유지하고 통신을 함.



15.4 Socket Programming



socket 클래스

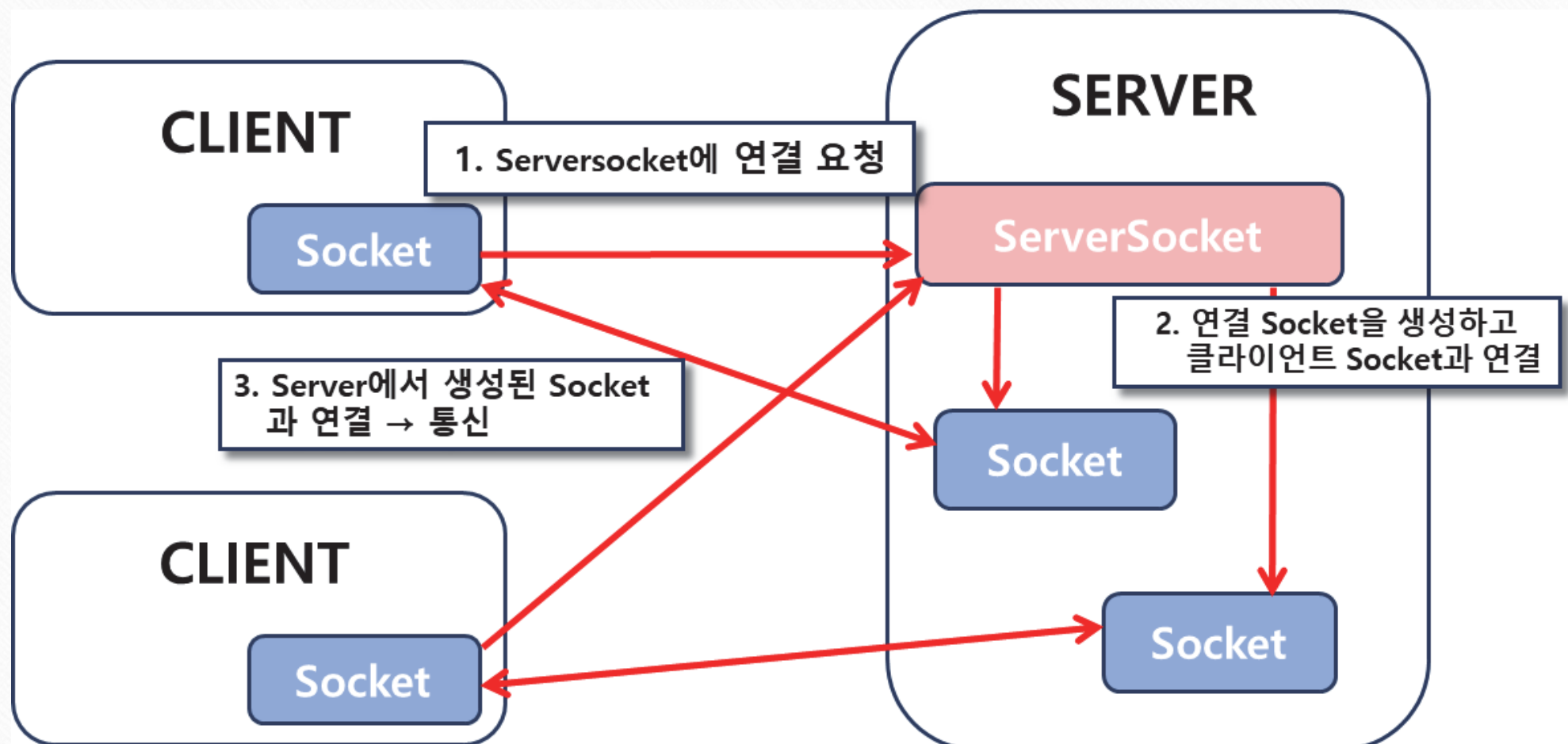
- 소켓은 단순히 시스템 간의 통신을 위한 끝점을 의미함.
- Socket 클래스를 사용하여 소켓을 만들 수 있음.
- 소켓 인스턴스를 생성하기 위해서는 서버 소켓의 IP주소와 port 번호가 있어야 하고, 소켓 연결이 가능한 서버가 실행 중이어야 함.
- 클라이언트 애플리케이션을 생성하려면 Socket 클래스의 인스턴스를 생성해야 함.
- 이때 생성자에 서버의 IP 주소 또는 호스트 이름과 포트 번호를 전달해야 함.

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


15.4 Socket Programming

ServerSocket 클래스

- ServerSocket 클래스를 사용하여 서버 소켓을 만들 수 있음.
- 이 인스턴스는 클라이언트 소켓과 연결하여 통신할 수 있는 소켓 인스턴스를 생성.



15.4 Socket Programming

26



ServerSocket 클래스

- 소켓 통신을 위한 서버 애플리케이션을 만들려면 ServerSocket 클래스의 인스턴스를 생성해야 하는데 통신을 위해 9999 포트 번호를 사용해서 ServerSocket 인스턴스를 생성.
- ServerSocket 인스턴스의 accept() 메소드를 이용해 클라이언트의 소켓연결 요청을 기다림.
- 클라이언트가 주어진 포트 번호로 연결하면 클라이언트와 연결할 수 있는 Socket의 인스턴스를 반환.

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

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


15.4 Socket Programming

27



Socket Programming – Server

```
package chapter15;

import java.io.DataInputStream;
import java.net.ServerSocket;
import java.net.Socket;

public class MySocketServer {

    public static void main(String[] args) {

        try {
            ServerSocket ss = new ServerSocket(9999);
            Socket s = ss.accept(); // Socket 연결 요청이 오면 Socket 인스턴스를 반환합니다.
            DataInputStream dis = new DataInputStream(s.getInputStream());
            String str = (String) dis.readUTF();
            System.out.println("message= " + str);
            ss.close();
        } catch (Exception e) {
            System.out.println(e);
        }

    }

}
```

15.4 Socket Programming

28



Socket Programming – Client

```
package chapter15;

import java.io.DataOutputStream;
import java.net.Socket;

public class ClientSocket {

    public static void main(String[] args) {
        try {
            // SocketServer의 ip와 port 정보를 이용해 ServerSocket에 연결할 Socket 인스턴스를 생성
            Socket s = new Socket("localhost", 9999);
            DataOutputStream dout = new DataOutputStream(s.getOutputStream());
            dout.writeUTF("안녕하세요~ Client에서 보내는 메시지 입니다.");
            dout.flush();
            dout.close();
            s.close();
        } catch (Exception e) {
            System.out.println(e);
        }
    }
}
```

결과(Console) - MySocketServer

message= 안녕하세요~ Client에서 보내는 메시지입니다.

15.4 Socket Programming

29



Socket Programming – 메시지를 주고받는 프로그램

```
package chapter15;

import java.io.DataInputStream;
import java.io.DataOutputStream;
import java.net.ServerSocket;
import java.net.Socket;
import java.util.Scanner;

public class MessageServer {

    public static void main(String args[]) throws Exception {

        ServerSocket ss = new ServerSocket(8888);
        Socket s = ss.accept();

        DataInputStream din = new DataInputStream(s.getInputStream());
        DataOutputStream dout = new DataOutputStream(s.getOutputStream());

        Scanner sc = new Scanner(System.in);

        String str = "", str2 = "";
```

15.4 Socket Programming

30



Socket Programming – 메시지를 주고받는 프로그램

```
while (!str.equals("exit")) {  
    str = din.readUTF();  
    System.out.println("client message: " + str);  
    str2 = sc.nextLine();  
    dout.writeUTF(str2);  
    dout.flush();  
}  
din.close();  
s.close();  
ss.close();
```

```
}
```

```
}
```


15.4 Socket Programming

31



Socket Programming – 메시지를 주고받는 프로그램

```
package chapter15;

import java.io.DataInputStream;
import java.io.DataOutputStream;
import java.net.Socket;
import java.util.Scanner;

public class MessageClient {

    public static void main(String args[]) throws Exception {

        Socket s = new Socket("localhost", 8888);

        DataInputStream din = new DataInputStream(s.getInputStream());
        DataOutputStream dout = new DataOutputStream(s.getOutputStream());
        Scanner sc = new Scanner(System.in);

        String str = "", str2 = "";
```

15.4 Socket Programming

32



Socket Programming – 메시지를 주고받는 프로그램

```
while (!str.equals("exit")) {  
    str = sc.nextLine();  
    dout.writeUTF(str);  
    dout.flush();  
    str2 = din.readUTF();  
    System.out.println("Server message: " + str2);  
}  
dout.close();  
s.close();  
}  
}
```

결과(Console) – MessageServer

client message: 안녕~!!
안녕하세요~

결과(Console) – MessageClient

안녕~!!
Server message: 안녕하세요~

감사합니다
