

Lecture 11

네트워크 프로그래밍




2018년도 2학기

컴퓨터프로그래밍2

김 영 국

충남대학교 컴퓨터공학과

이번 주에 학습할 내용

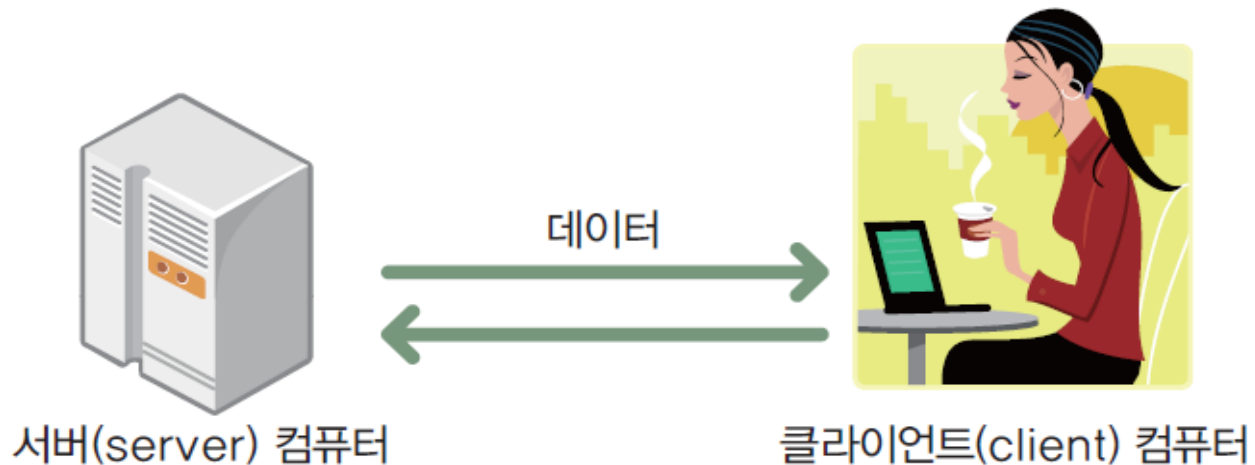
- 
- 네트워크 프로그래밍의 개요
 - URL 클래스
 - TCP를 이용한 통신
 - TCP를 이용한 서버 제작
 - TCP를 이용한 클라이언트 제작
 - UDP를 이용한 통신

자바를
이용하여서
TCP/IP 통신을
이용하는 응용
프로그램을
작성하여
봅시다.



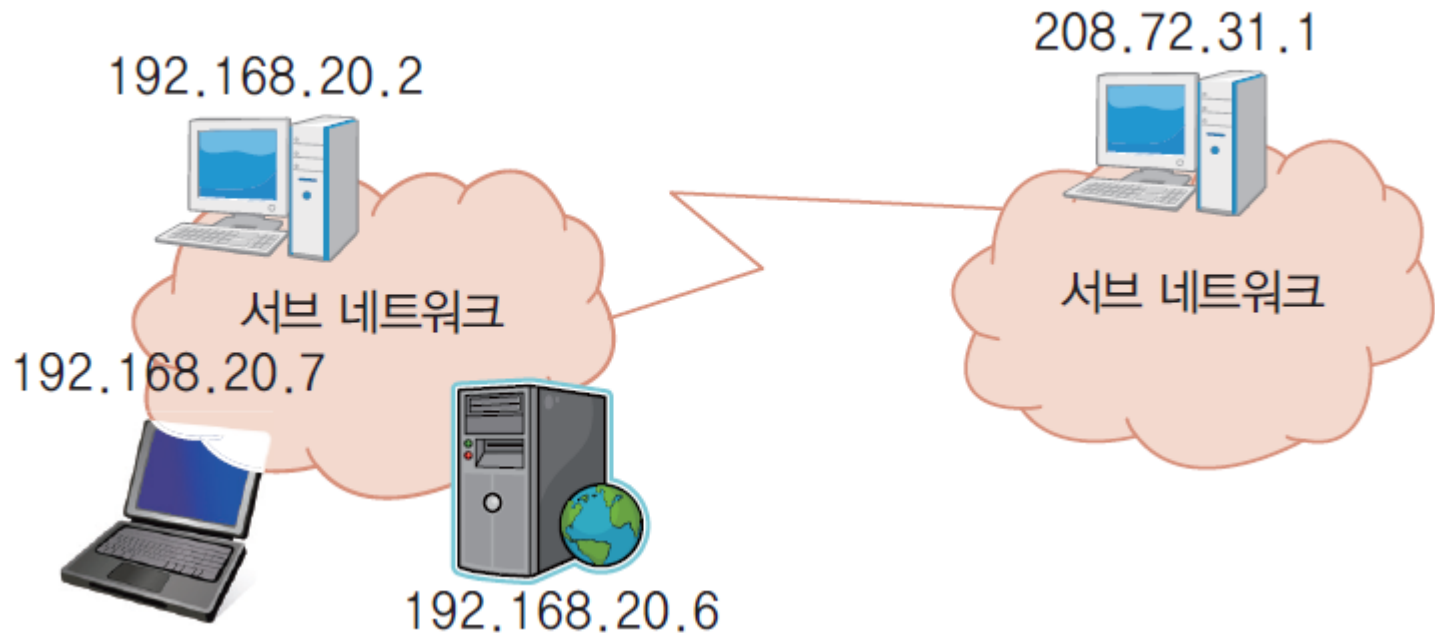
서버와 클라이언트

- **서버(Server):** 사용자들에게 서비스를 제공하는 컴퓨터
- **클라이언트(Client):** 서버에게 서비스를 요청해서 사용하는 컴퓨터
- (예) 웹서버와 클라이언트



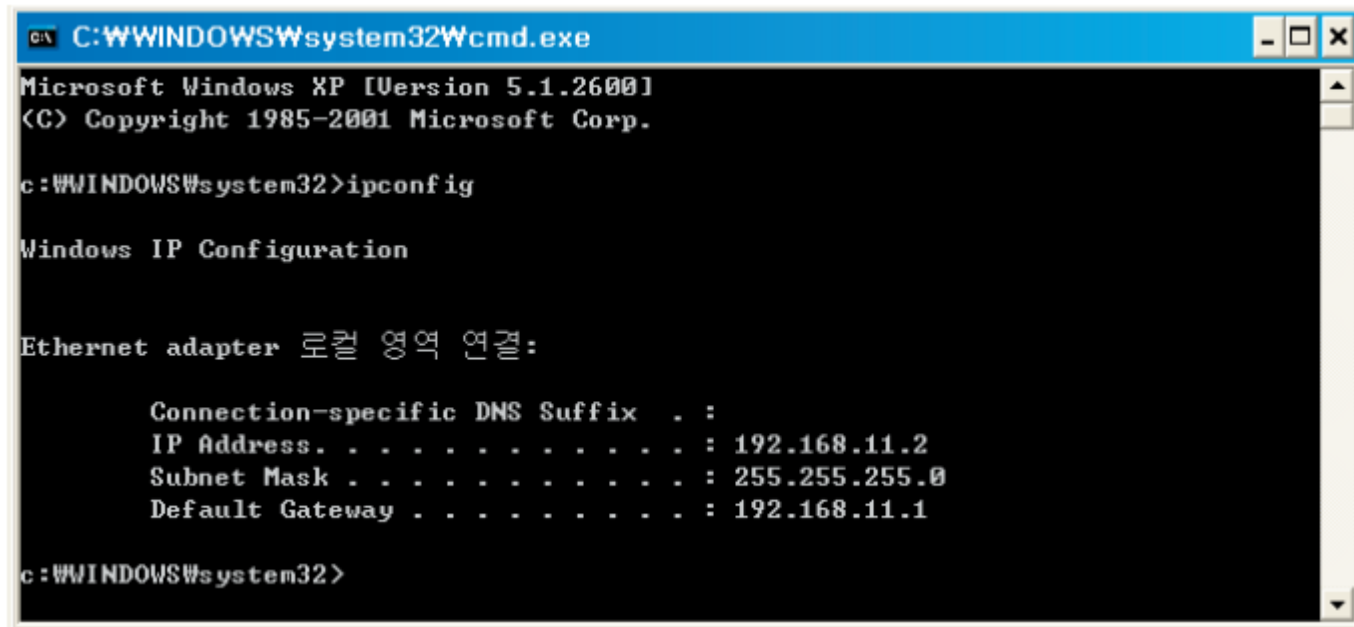
IP 주소

- IP 주소: 인터넷에서 컴퓨터의 주소



IP 주소

- IP 주소: 인터넷에서 컴퓨터의 주소



```
C:\WINDOWS\system32\cmd.exe
Microsoft Windows XP [Version 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.

c:\WINDOWS\system32>ipconfig

Windows IP Configuration

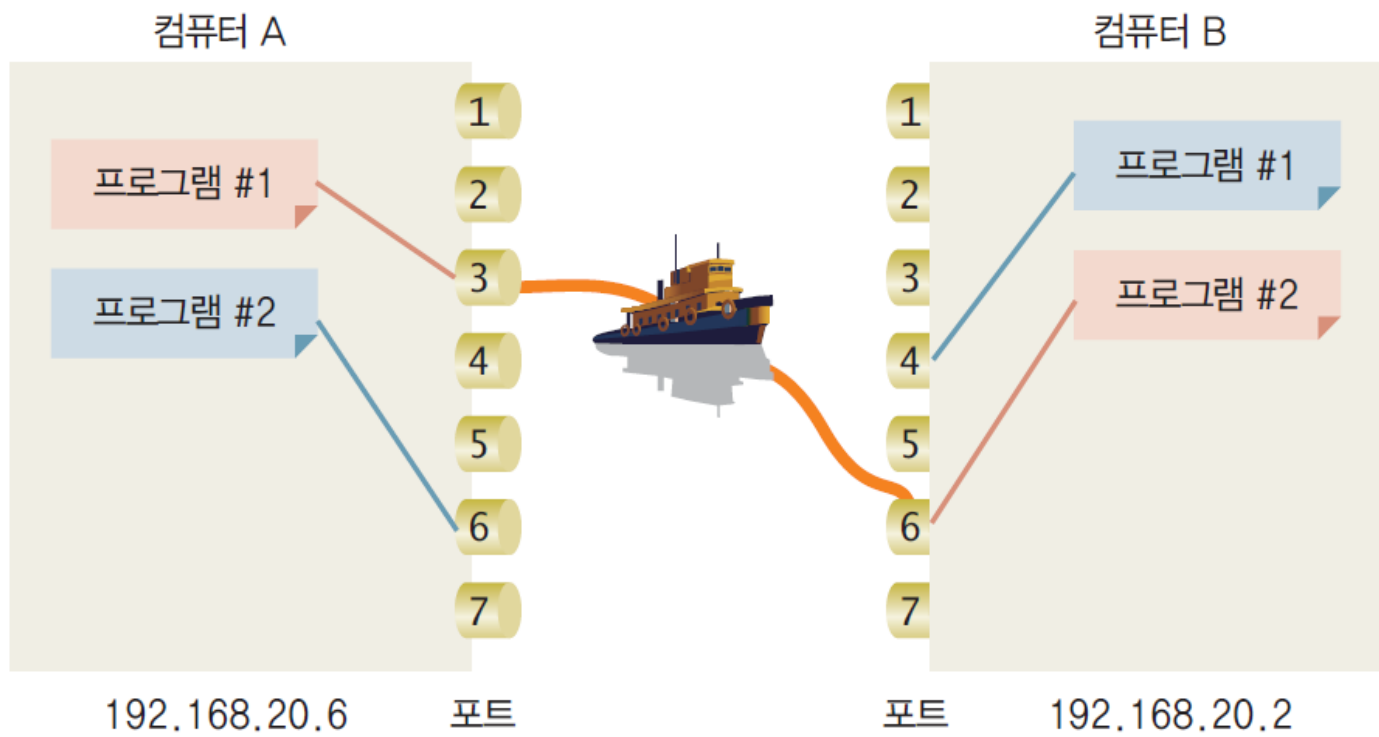
Ethernet adapter 로컬 영역 연결:

    Connection-specific DNS Suffix  . : 
    IP Address. . . . .               : 192.168.11.2
    Subnet Mask . . . . .             : 255.255.255.0
    Default Gateway . . . . .         : 192.168.11.1

c:\WINDOWS\system32>
```

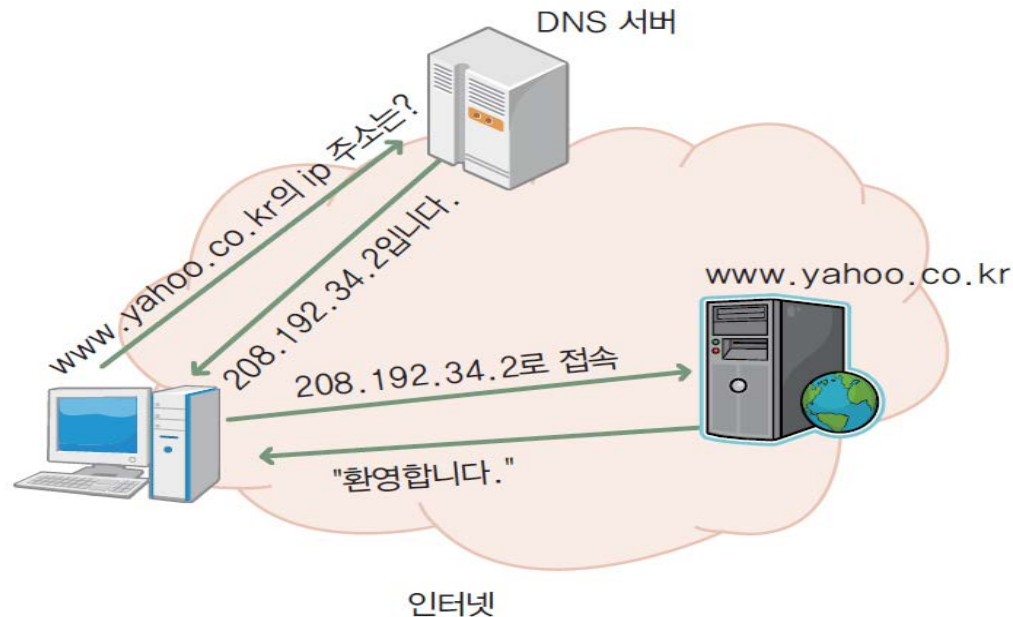
포트

- 포트(port): 가상적인 통신 선로



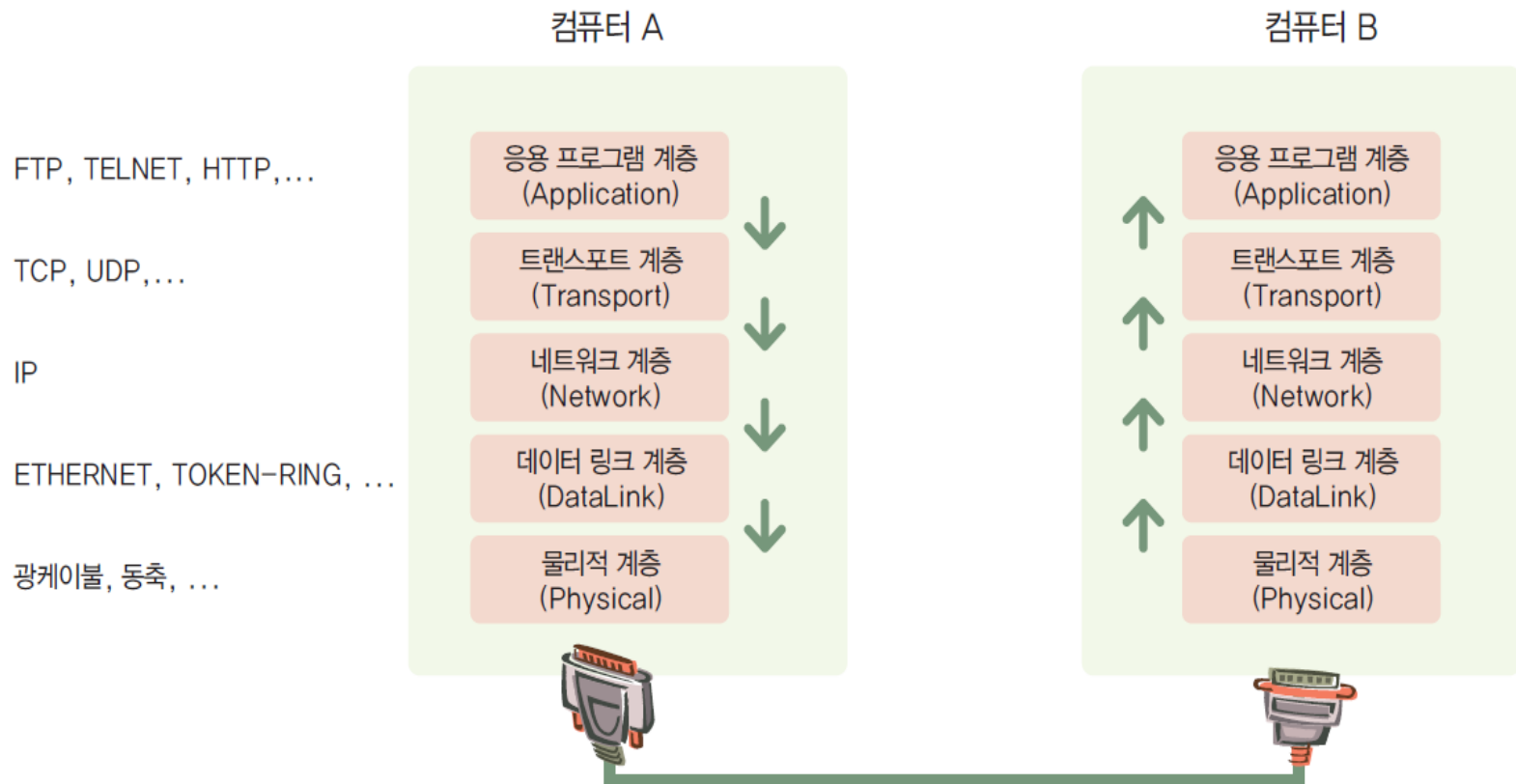
호스트 이름, DNS, URL

- **DNS(Domain Name System):** 숫자 대신 기호를 사용하는 주소
- **DNS 서버:** 기호 주소를 숫자 주소가 변환해주는 서버
- **URL(Uniform Resource Locator):** 인터넷 상의 자원을 나타내는 약속



프로토콜

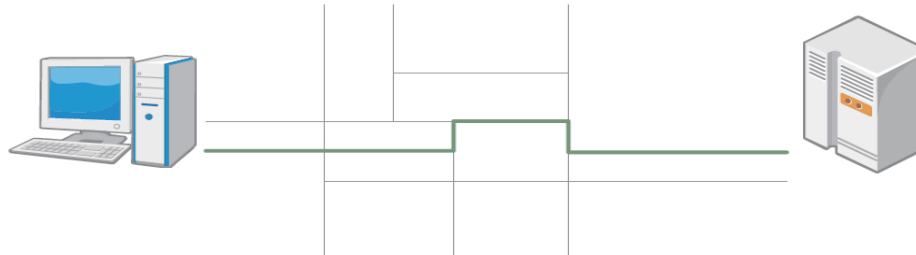
- 프로토콜(protocol): 통신을 하기 위한 약속



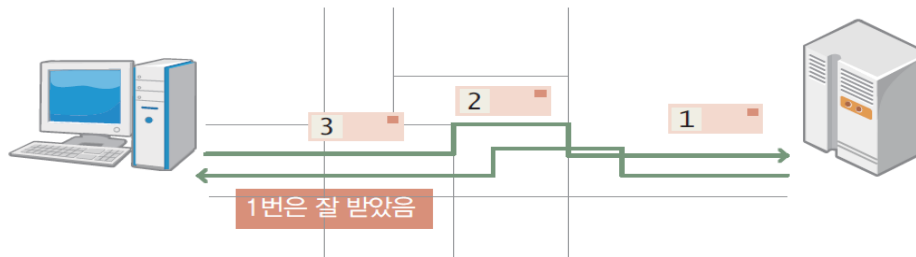
TCP

- **TCP(Transmission Control Protocol)**는 신뢰성 있게 통신하기 위하여 먼저 서로 간에 연결을 설정한 후에 데이터를 보내고 받는 방식

(1) 먼저 가능한 경로 중에서 하나가 결정된다.



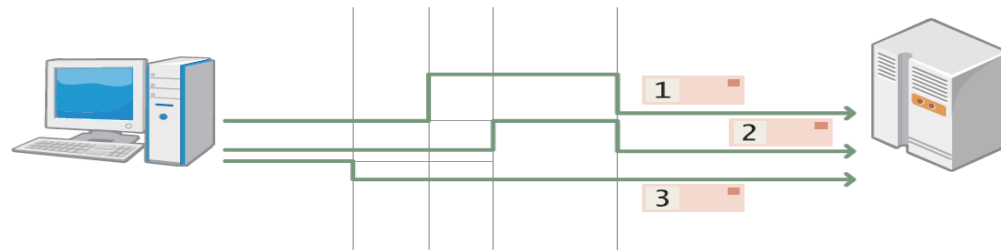
(2) 데이터는 패킷으로 나누어지고 패킷에 주소를 붙여서 전송한다.



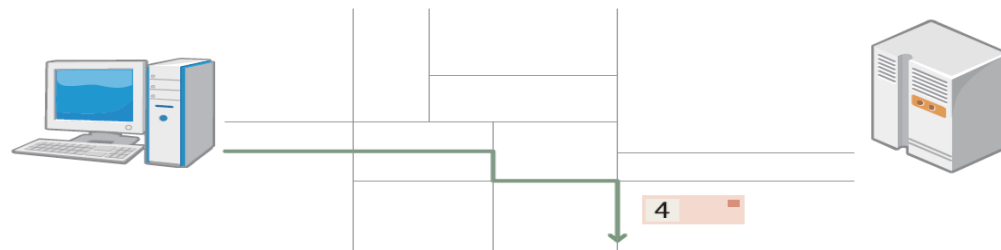
UDP

- **UDP(User Datagram Protocol)**는 데이터를 몇 개의 고정 길이의 패킷(다이어그램이라고 불린다)으로 분할하여 전송

(1) 데이터를 패킷으로 나누어서 패킷에 주소를 붙이고 전송한다.



(2) 패킷의 순서가 지켜지지 않으며 패킷이 분실될 수도 있다.





자바와 네트워크

- 네트워크 프로그래밍을 위한 패키지는 java.net
- TCP를 위한 클래스
 - URL
 - URLConnection
 - Socket
 - ServerSocket
- UDP를 위한 클래스
 - DatagramPacket
 - DatagramSocket
 - MulticastSocket

중간 점검 문제



중간점검

1. IP 주소와 도메인 이름은 어떻게 다른가?
2. 전화와 비슷한 전송 프로토콜은 _____이고 편지와 비슷한 프로토콜은 _____이다.
3. TCP/IP에서 자신을 가리키는 주소는?

URL 클래스

- URL java = **new** URL("http://java.sun.com"); // 절대 경로
- URL reference = **new** URL(java, "reference.html"); // 상대 경로

// www.yahoo.co.kr/index.html:80

↑
호스트 이름

↑
파일 이름

↑
포트번호

예제

ParseURLExample.java

```
01 import java.net.*;
02 import java.io.*;
03
04 public class ParseURLExample {
05     public static void main(String[] args) throws Exception {
06         URL myURL = new URL("http://java.sun.com:80/docs/books/tutorial"
07                             + "/index.html?name=database#TOP");
08         System.out.println("protocol = " + myURL.getProtocol());
09         System.out.println("authority = " + myURL.getAuthority());
10         System.out.println("host = " + myURL.getHost());
11         System.out.println("port = " + myURL.getPort());
12         System.out.println("path = " + myURL.getPath());
13         System.out.println("query = " + myURL.getQuery());
14         System.out.println("filename = " + myURL.getFile());
15         System.out.println("ref = " + myURL.getRef());
16     }
17 }
```

←----- URL 객체 생성



예제

실행결과

```
protocol = http
authority = java.sun.com:80
host = java.sun.com
port = 80
path = /docs/books/tutorial/index.html
query = name=database
filename = /docs/books/tutorial/index.html?name=database
ref = TOP
```



URLConnection 클래스

```
try {  
    URL java = new URL("http://java.sun.com/");  
    URLConnection javac = java.openConnection();  
    javac.connect = java.openConnection();  
    // URLConnection을 이용한 통신  
} catch (MalformedURLException e) {      // new URL() 실패  
    // 예외처리  
} catch (IOException e) {                // openConnection() 실패  
    // 예외처리  
}
```


URLConnection을 이용한 읽기

URLConnectionReader.java

```
01 import java.net.*;
02 import java.io.*;
03
04 public class URLConnectionReader {
05     public static void main(String[] args) throws Exception {
06         URL yahoo = new URL("http://www.naver.com/");
07         URLConnection yc = yahoo.openConnection(); ← URL에 연결한다.
08         BufferedReader in = new BufferedReader(
09             new InputStreamReader(
10                 yc.getInputStream()));
11         String inLine;
12
13         while ((inLine = in.readLine()) != null)
14             System.out.println(inLine);
15         in.close();
16     }
17 }
```



예제

실행결과

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="ko" lang="ko">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=euc-kr" />
...
```



CGI 스크립트와 대화하기

- URL을 생성한다.
- URLConnection을 추출한다.
- 연결로부터 출력 스트림을 얻는다. 출력 스트림은 서버 쪽의 cgi-bin 스크립트의 표준 입력 스트림에 연결된다.
- 출력 스트림에 데이터를 출력한다.
- 출력 스트림을 닫는다.



CGI 스크립트와 대화하기

URLPostTest.java

```
01 import java.io.*;
02 import java.net.*;
03
04 public class URLPostTest {
05     public static void main(String[] args) throws Exception {
```

CGI 스크립트와 대화하기

```
06
07     String s = URLEncoder.encode("This is a test", "UTF-8");
08
09     URL url = new URL("http://igchun.sch.ac.kr/script/test");
10     URLConnection connection = url.openConnection();
11     connection.setDoOutput(true);
12
13     OutputStreamWriter out = new OutputStreamWriter(connection.getOutputStream());
14     out.write("string=" + s);
15     out.close();
16
17     BufferedReader in = new BufferedReader(
18         new InputStreamReader(
19             connection.getInputStream()));
20     String inputLine;
21
22     while ((inputLine = in.readLine()) != null)
23         System.out.println(inputLine);
24
25     in.close();
26 }
27 }
```

미리 서버로 연결

서버에 쓴다.

서버로부터 읽는다.



예제

실행결과

```
string=This is a test
```

중간 점검 문제

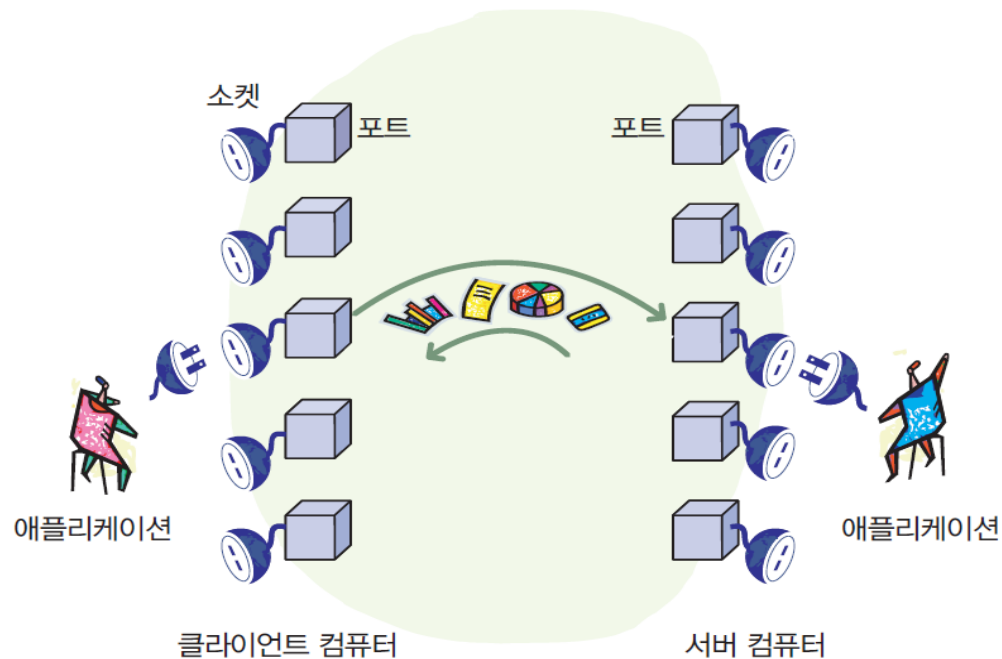


중간점검

1. URL 클래스를 이용하여서 할 수 있는 작업은?
2. `URLConnection` 클래스를 이용하여서 할 수 있는 작업은?

Socket 클래스

- 소켓(socket): TCP를 사용하여 응용 프로그램끼리 통신을 하기 위한 연결 끝점(end point)



< 소켓의 개념 >

ServerSocket과 Socket

소켓의 종류

ServerSocket 클래스: 서버를 위한 소켓



서버(server) 컴퓨터

Socket 클래스: 클라이언트를 위한 소켓



클라이언트(client) 컴퓨터



Socket 클래스

생성자	설명
<code>Socket(String host, int port)</code>	호스트 이름이 <code>host</code> 이고 포트 번호가 <code>port</code> 인 새로운 소켓을 생성한다.
<code>Socket(InetAddress address, int port)</code>	<code>InetAddress</code> 에 기술된 주소로 새로운 소켓을 생성한다.

메소드	설명
<code>InputStream getInputStream()</code>	소켓이 사용하는 입력 스트림을 반환한다.
<code>OutputStream getOutputStream()</code>	소켓이 사용하는 출력 스트림을 반환한다.
<code>InetAddress getInetAddress()</code>	소켓이 연결되어 있는 인터넷 주소를 반환한다.
<code>public int getLocalPort()</code>	소켓이 연결되어 있는 포트 번호를 반환한다.
<code>public int getPort()</code>	원격 컴퓨터의 포트 번호를 반환한다.
<code>public InetAddress getLocalAddress()</code>	소켓이 연결되어 있는 인터넷 주소를 반환한다.

ServerSocket 클래스

생성자	설명
<code>public ServerSocket(int port) throws IOException</code> <code>public ServerSocket(int port, int queue)</code>	포트 번호 <code>port</code> 에 대해 <code>ServerSocket</code> 의 새로운 인스턴스를 만든다. 포트 번호 <code>0</code> 는 비어있는 포트 번호를 사용한다는 의미이다. <code>queue</code> 는 서버가 받을 수 있는 입력 연결의 개수를 의미한다.(디폴트는 50 연결이다.) <code>addr</code> 는 컴퓨터의 인터넷 주소를 나타낸다.
<code>public ServerSocket(int port, int queue, InetAddress addr)</code>	
메소드	설명
<code>public Socket accept()</code>	접속 요청을 받는다.
<code>public void close()</code>	<code>ServerSocket</code> 을 닫는다.
<code>public InetAddress getInetAddress()</code>	소켓이 연결되어 있는 인터넷 주소를 반환한다.
<code>public int getSoTimeout()</code>	소켓에 대한 타임아웃 값을 밀리 초로 반환하거나 설정한다.

중간 점검 문제



중간점검

1. 소켓은 _____과 _____을 연결하는 역할을 한다.
2. 서버가 사용하는 소켓 클래스는?



스트림 소켓을 이용한 서버 제작

1. ServerSocket 객체 생성

- `ServerSocket server = new ServerSocket(portNumber, queueLength);`

2. accept() 메소드 호출

- `Socket clientSocket = server.accept();`

3. 소켓으로부터 스트림 객체를 얻는다.

- `InputStream input = clientSocket.getInputStream();`
- `OutputStream output = clientSocket.getOutputStream();`

4. 상호 대화 단계

- `read()`와 `write()` 사용

5. 종료

- `close()` 사용

TCP 예제: 퀴즈 서버와 클라이언트



퀴즈 클라이언트:
퀴즈에 대한 답을
보낸다.



퀴즈 서버:
퀴즈를 출제한다.

QuizServer 클래스

QuizServer.java

```
01 import java.net.*;
02 import java.io.*;
03
04
05 public class QuizServer {
06     public static void main(String[] args) throws IOException {
07
08         ServerSocket serverSocket = null;
09         try {
10             serverSocket = new ServerSocket(5555);
11         } catch (IOException e) {
12             System.err.println("다음의 포트 번호에 연결할 수 없습니다: 5555");
13             System.exit(1);
14         }
15
16         Socket clientSocket = null;
17         try {
18             clientSocket = serverSocket.accept();
19         } catch (IOException e) {
20             System.err.println("accept() 실패");
21             System.exit(1);
22         }
```

서버로 접속 요청을 받는다.

QuizServer 클래스

```
23
24     PrintWriter out = new PrintWriter(clientSocket.getOutputStream(), true);
25     BufferedReader in = new BufferedReader(
26         new InputStreamReader(
27             clientSocket.getInputStream()));
28     String inputLine, outputLine;
29     QuizProtocol qp = new QuizProtocol();
30
31     outputLine = qp.process(null);
32     out.println(outputLine);
33
34     while ((inputLine = in.readLine()) != null) {
35         outputLine = qp.process(inputLine);
36         out.println(outputLine);
37         if (outputLine.equals("quit"))
38             break;
39     }
40     out.close();
41     in.close();
42     clientSocket.close();
43     serverSocket.close();
44 }
45 }
```

소켓에 입력 스트림과
출력 스트림을 연결한다

클라이언트로부터 요청을 읽어서
퀴즈를 내고 답을 읽는다.

QuizProtocol 클래스

QuizProtocol.java

```
01 class QuizProtocol {
02     private static final int WAITING = 0;
03     private static final int PROBLEM = 1;
04     private static final int ANSWER = 2;
05
06     private static final int NUMPROBLEMS = 3;
07
08     private int state = WAITING;
09     private int currentProblem = 0;
10
11     private String[] problems = { "네트워크 처리 패키지는?", "자바의 최신버전은?",
12                                   "인터넷에서 컴퓨터를 식별하는 주소는?" };
13     private String[] answers = { "java.io",
14                                   "1.7",
15                                   "IP 주소" };

```

프로토콜이 가질 수 있는 상태

문제

답

QuizProtocol 클래스

```
16
17     public String process(String theInput) {
18         String theOutput = null;
19
20         if (state == WAITING) {
21             theOutput = "퀴즈를 시작합니다(y/n)";
22             state = PROBLEM;
23         } else if (state == PROBLEM) {
24             if (theInput.equalsIgnoreCase("y")) {
25                 theOutput = problems[currentProblem];
26                 state = ANSWER;
27             } else {
28                 state = WAITING;
29                 theOutput = "quit";
30             }
        }
```

상태가 WAITING이면
시작 메시지를 보낸다.

상태가 PROBLEM이
면 문제를 보낸다.

QuizProtocol 클래스

```
31     } else if (state == ANSWER) {
32         if (theInput.equalsIgnoreCase(answers[currentProblem])) {
33             theOutput = "정답입니다. 계속하시겠습니까? (y/n)";
34             state = PROBLEM;
35         } else {
36             state = PROBLEM;
37             theOutput = "오답입니다. 계속하시겠습니까? (y/n)";
38         }
39         currentProblem = (currentProblem+1)% NUMPROBLEMS;
40     }
41     return theOutput;
42 }
43 }
```

상태가 ANSWER이면
답을 확인한다.

중간 점검 문제



중간점검

1. TCP 통신에서 새로운 연결이 만들어지는 과정을 설명하라.
2. `accept()` 메소드가 반환하는 값은 무엇인가?
3. 소켓에서 입력 스트림 객체를 얻는 메소드 이름은 무엇인가?

스트림 소켓을 이용한 클라이언트 제작

QuizClient.java

```
01 import java.io.*;
02 import java.net.*;
03
04 public class QuizClient {
05     public static void main(String[] args) throws IOException {
06
07         Socket quizSocket = null;
08         PrintWriter out = null;
09         BufferedReader in = null;
10
11         try {
12             quizSocket = new Socket("localhost", 5555);
13             out = new PrintWriter(quizSocket.getOutputStream(), true);
14             in = new BufferedReader(new InputStreamReader(quizSocket
15                                     .getInputStream()));
16         } catch (UnknownHostException e) {
17             System.err.println("localhost에 접근할 수 없습니다.");
18             System.exit(1);
19         } catch (IOException e) {
20             System.err.println("입출력 오류");
21             System.exit(1);
22         }
23     }
```

소켓을 생성하고 소켓에
입력 스트림과 출력
스트림을 붙인다.

스트림 소켓을 이용한 클라이언트 제작

서버로부터 문자열을
읽고 사용자의 입력을
서버로 보낸다.

```
24      BufferedReader user = new BufferedReader(new InputStreamReader(  
25                                                  System.in));  
26      String fromServer;  
27      String fromUser;  
28  
29      while ((fromServer = in.readLine()) != null) {  
30          System.out.println("서버: " + fromServer);  
31          if (fromServer.equals("quit"))  
32              break;  
33  
34          fromUser = user.readLine();  
35          if (fromUser != null) {  
36              System.out.println("클라이언트: " + fromUser);  
37              out.println(fromUser);  
38          }  
39      }  
40  
41      out.close();  
42      in.close();  
43      quizSocket.close();  
44  }  
45 }
```

서버와 클라이언트 프로그램 수행

- 두 개의 프로그램을 동시에 실행하여야 한다.

```
C> java QuizServer
```

```
...
```

```
C> java QuizClient
```

```
...
```

실행결과

서버: 퀴즈를 시작합니다(y/n)

y

클라이언트: y

서버: 네트워크 처리 패키지는?

java.io

클라이언트: java.io

서버: 정답입니다. 계속하시겠습니까? (y/n)

n

클라이언트: n

서버: quit



다중 클라이언트를 지원

- 각각의 클라이언트를 별도의 스레드로 처리하여야 한다.

```
while(true){  
    연결 요청을 수락한다;  
    클라이언트를 대응하는 쓰레드를 만든다;  
}
```


UDP를 이용한 통신

- DatagramSocket 클래스
 - DatagramSocket()은 UDP 프로토콜을 사용하는 소켓을 생성
- DatagramPacket 클래스
 - DatagramPacket()은 UDP 패킷을 생성한다.



TCP 프로토콜



UDP 프로토콜

UDP를 사용해서 데이터 보내고 받기

Sender.java

```
01 import java.net.*;
02 import java.io.*;
03
04 public class Sender {
05     public static void main(String[] args) throws IOException {
06
07         DatagramSocket socket = null;
08         socket = new DatagramSocket();
09         String s = "우리는 여전히 우리 운명의 주인이다.";
10         byte[] buf = s.getBytes();
11
12         // "address"의 "port"에 있는 클라이언트에게 데이터를 보낸다.
13         InetAddress address = InetAddress.getByName("127.0.0.1"); // 로컬 호스트
14         DatagramPacket packet = new DatagramPacket(buf, buf.length, address, 5000);
15         socket.send(packet);
16         socket.close();
17     }
18 }
```

데이터그램 소켓을
생성한다.

주소와 데이터를
데이터그램에
저장해서 송신한다.

UDP를 사용해서 데이터 보내고 받기

포트 5000을 통하여
데이터그램을 수신한다.

Receiver.java

```
01 import java.io.*;
02 import java.net.*;
03
04 public class Receiver {
05     public static void main(String[] args) throws IOException {
06
07         byte[] buf = new byte[256];
08
09         DatagramSocket socket = new DatagramSocket(5000); // 포트 번호: 5000
10         DatagramPacket packet = new DatagramPacket(buf, buf.length);
11         socket.receive(packet);
12         System.out.println(new String(buf));
13     }
14 }
```



데이터그램을 이용한 서버와 클라이언트 작성하기

MessengerA.java

```
01 import java.io.*;
02 import java.net.*;
03
04 import java.awt.*;
05 import java.awt.event.*;
06 import javax.swing.*;
07
08 public class MessengerA {
09     protected JTextField textField;
10     protected JTextArea textArea;
11     DatagramSocket socket;
12     DatagramPacket packet;
13     InetAddress address = null;
14     final int myPort = 5000;      // 수신용 포트 번호
15     final int otherPort = 6000;  // 송신용 포트 번호
16
17     public MessengerA() throws IOException {
18         MyFrame f = new MyFrame();
19         address = InetAddress.getByName("127.0.0.1");
20         socket = new DatagramSocket(myPort);
21     }
```



데이터그램을 이용한 서버와 클라이언트 작성하기

```
22
23     // 패킷을 받아서 텍스트 영역에 표시한다.
24     public void process() {
25         while (true) {
26             try
27             {
28                 byte[] buf = new byte[256];
29                 packet = new DatagramPacket(buf, buf.length);
30                 socket.receive(packet); // 패킷을 받는다.
31                 // 받은 패킷을 텍스트 영역에 표시한다.
32                 textArea.append("RECIEVED: " + new String(buf) + "\n");
33             }
34             catch (IOException ioException) {
35                 ioException.printStackTrace();
36             }
37         }
38     }
39
```

데이터그램을 이용한 서버와 클라이언트 작성하기

```
40 // 내부 클래스 정의
41 class MyFrame extends JFrame implements ActionListener {
42
43     public MyFrame() {
44         super("MessengerA");
45         setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
46
47         textField = new JTextField(30);
48         textField.addActionListener(this);
49
50         textArea = new JTextArea(10, 30);
51         textArea.setEditable(false);
52
53         add(textField, BorderLayout.PAGE_END);
54         add(textArea, BorderLayout.CENTER);
55         pack();
56         setVisible(true);
57     }
58
59     public void actionPerformed(ActionEvent evt) {
60         String s = textField.getText();
61         byte[] buffer = s.getBytes();
62         DatagramPacket packet;
63
```

메신저 사용자
인터페이스

데이터그램을 이용한 서버와 클라이언트 작성하기

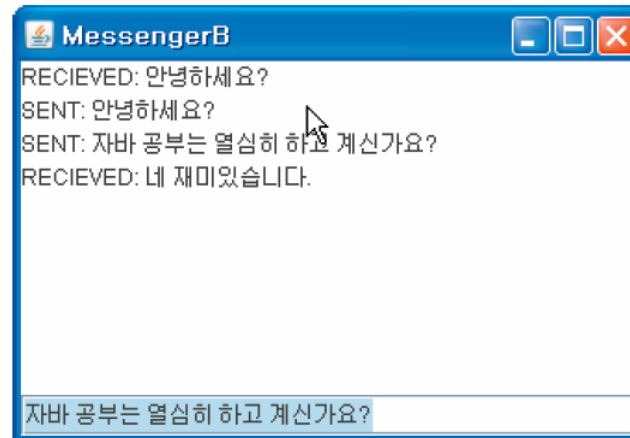
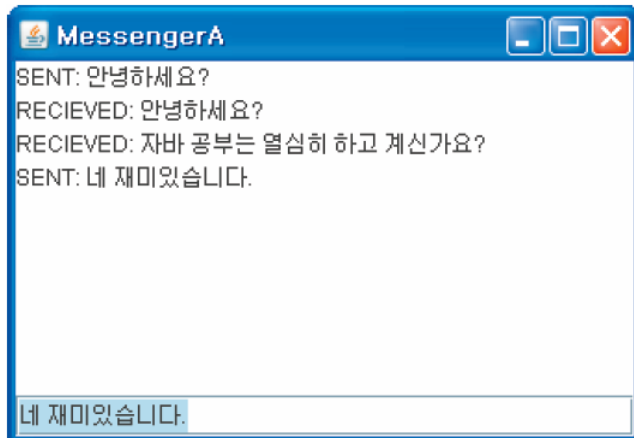
```
64      // 패킷을 생성한다.
65      packet = new DatagramPacket(buffer, buffer.length, address, otherPort);
66      try {
67          socket.send(packet);      // 패킷을 보낸다.
68      } catch (IOException e) {
69          e.printStackTrace();
70      }
71      textArea.append("SENT: " + s + "\n");
72      textField.selectAll();
73      textArea.setCaretPosition(textArea.getDocument().getLength());
74  }
75  }
76
77  public static void main(String[] args) throws IOException {
78      MessengerA m = new MessengerA();
79      m.process();
80  }
81  }
```

데이터그램을 이용한 서버와 클라이언트 작성하기

MessengerB.java

```
01 // 다음의 몇 개의 문장만 제외하고 MessengerA와 동일
02 ....
03 public class MessengerB {
04     ...
05     final int myPort = 6000;
06     final int otherPort = 5000;
07
08     public MessengerB() throws IOException {
09         ...
10     }
11     public static void main(String[] args) throws IOException {
12         MessengerB m = new MessengerB();
13         m.process();
14     }
15 }
```


예제



중간 점검 문제



중간점검

1. UDP의 장점과 단점은 무엇인가?
2. UDP에서는 패킷을 받을 상대방을 어떻게 지정하는가?
3. `DatagramSocket` 클래스에서 패킷을 보내고 받는 메소드 이름은?

Q & A

