Network

Network

- 용어
  - ✓ Network: 장비 또는 사용자 들 간에 하드웨어와 소프트웨어를 공유할 수 있도록 서로 연결된 노드 들의 모임
  - ✓ Protocol: 장비 사이의 데이터 송수신을 위한 규칙-TCP (연결형), UDP(비연결형)
  - ✓ TCP/IP: 인터넷상에서 호스트들을 서로 연결시키는데 필요한 프로토콜
  - ✓ HTTP: Hyper Text Transfer Protocol
  - ✓ HTTPS: Hyper Text Transfer Protocol over Secure Socket Layer
  - ✓ IP 주소: 인터넷에서 단말기를 구별하기 위한 주소(IPv4-32비트, IPv6-128비트)
    - ❑ IPv4: 32비트 주소 체계로 8개의 비트를 하나의 그룹으로 묶어서 .으로 구분하고 10진수로 표현(0.0.0.0 – 255.255.255.255)
    - ❑ IPv6: 128비트 주소 체계로 16비트를 하나의 그룹으로 묶어서 :: 으로 구분하고 16진수로 표현하며 연속된 0은 생략이 가능
      (0000::0000::0000::0000::0000::0000::0000::0000 – FFFF:: FFFF:: FFFF:: FFFF:: FFFF:: FFFF:: FFFF:: FFFF)
    - ❑ Loopback 주소: 자신의 장비 주소를 나타내는 것으로 IPv4에서는 127.0.0.1 이며 IPv6에서는 0::0::0::0::0::0::0::1
  - ✓ PORT: 하나의 ip에서 프로세스를 구분하기 위한 번호로 0 ~ 65535 (0 ~ 1024번은 시스템이 사용하는 영역)으로 생성되며 하나의 프로세스가 하나 이상의 포트 사용 가능
    - ❑ 시스템 예약: http(80), FTP(20, 21), SSH(22), Telnet(23), SMTP(25), DNS(53)
    - ❑ 응용프로그램: Tomcat(8080), Oracle(1521), MySql(3306), MSSQL(1443)..

Network

- 용어
  - ✓ Domain: ip 주소에 매핑시킨 문자로 된 주소
  - ✓ URI(Uniform Resource Identifier, 통합 자원 식별자)
    - ❑ 인터넷에 있는 자원을 나타내는 유일한 주소
    - ❑ URI의 존재는 인터넷에서 요구되는 기본조건으로서 인터넷 프로토콜에 항상 붙어 다님
    - ❑ URI의 하위개념으로 URL, URN 이 있음
  - ✓ URL(Uniform Resource Locator 또는 web address)
    - ❑ 네트워크 상에서 자원이 어디 있는지를 알려주기 위한 규약
    - ❑ 컴퓨터 네트워크와 검색 메커니즘에서의 위치를 지정하는 웹 리소스에 대한 참조
    - ❑ 웹 흔히 웹 사이트 주소로 알고 있지만 URL은 웹 사이트 주소 뿐만 아니라 컴퓨터 네트워크상의 자원을 모두 나타낼 수 있음
    - ❑ 그 주소에 접속하려면 해당 URL에 맞는 프로토콜을 알아야 하고 그와 동일한 프로토콜로 접속해야 함
  - ✓ URN(Uniform Resource Name, 통합 자원 이름)
    - ❑ urn:scheme 을 사용하는 URI를 위한 이름
    - ❑ URN은 영속적이고 위치에 독립적인 자원을 위한 지시자로 사용하기 위해 1997년도 RFC 2141 문서에서 정의

Network

- 용어
  - ✓ 전송 방식에 따른 통신 분류
    - ❑ Unicast: 데이터를 특정 대상에게만 송신하는 형태
    - ❑ Broadcast: 데이터를 동시에 전송하여 동일 그룹 내의 모든 대상에게 송신하는 형태
    - ❑ Multicast: 특정 그룹에게만 송신하는 형태
    - ❑ Anycast: 트래픽 요청을 처리 할 수 있는 가장 가까운 데이터 센터로 라우팅
  - ✓ Subnet mask: 동일 네트워크의 규모를 나타내기 위한 주소
  - ✓ Gateway: 외부 네트워크로 나가기 위한 내부 네트워크의 종단점
  - ✓ Routing: 최적의 경로를 찾는 방법
  - ✓ Proxy: 클라이언트가 자신을 통해서 다른 네트워크 서비스에 간접적으로 접속할 수 있게 해주는 컴퓨터 시스템이나 응용 프로그램
  - ✓ Firewall: 미리 정의된 보안 규칙에 기반한, 들어오고 나가는 네트워크 트래픽을 모니터링하고 제어하는 네트워크 보안 시스템으로 서로 다른 네트워크를 지나는 데이터를 허용하거나 거부하거나 검열, 수정하는 하드웨어나 소프트웨어 장치
  - ✓ RPC(Remote Procedure Call)
    - ❑ 원격 제어를 코딩 없이 다른 주소 공간에서 함수나 프로시저를 실행할 수 있게 하는 프로세스 간 통신 기술
    - ❑ 원격 프로시저 호출을 이용하면 프로그래머는 함수가 실행 프로그램에 로컬 위치에 있든 원격 위치에 있든 동일한 코드를 이용할 수 있음

네트워크 용어

- 용어
  - ✓ IP 정보 확인
    - ❑ ipconfig(MS-Windows)
    - ❑ ifconfig(MS-Windows 가 아닌 경우)
  - ✓ ping: IP 네트워크를 통해 특정한 호스트가 도달할 수 있는지의 여부를 테스트하는 데 쓰이는 컴퓨터 네트워크 도구
  - ✓ LoopBack IP
    - ❑ IPv4 및 IPv6에서, 자기 자신을 가리키기 위한 목적으로 쓰기 위해 예약된 IP 주소
    - ❑ IPv4의 경우 127.0.0.0부터 127.255.255.255(127.0.0.0/8) 까지 있으며, 보통 127.0.0.1을 사용
    - ❑ IPv6은 ::1/128 한 개의 주소만 사용

네트워크 용어

- 용어
  - ✓ 오픈 API(Open Application Programming Interface, Open API, 공개 API)
    - ❑ 누구나 사용할 수 있도록 공개된 API를 말하며 개발자에게 사유 응용 소프트웨어나 웹 서비스에 프로그래밍적인 권한을 제공
  - ✓ REST(Representational State Transfer): 월드 와이드 웹과 같은 분산 하이퍼미디어 시스템을 위한 소프트웨어 아키텍처의 한 형식
  - ✓ 오픈 API 에서 많이 사용되는 데이터 포맷
    - ❑ CSV: 구분자로 구분된 문자열로 일반적으로 파일의 형태로 제공
    - ❑ XML: 태그 형식으로 데이터를 표현하는 포맷
    - ❑ JSON: 자바스크립트 데이터 표현법으로 데이터를 표현하는 포맷
    - ❑ YAML: XML, C, 파이썬, 펄, RFC2822에서 정의된 e-mail 양식에서 개념을 얻어 만들어진 '사람이 쉽게 읽을 수 있는' 데이터 직렬화 양식

InetAddress

❖컴퓨터의 IP정보를 저장하는 클래스

- ✓ static 메서드 인 getLocalHost(), getByName(호스트이름), getAllByName(호스트이름)으로 객체를 생성
- ✓ 객체 생성 시 UnknownHostException 예외처리가 필요
- ✓ getHostName(): 호스트 명을 리턴
- ✓ getAddress(): ip 주소를 바이트 배열로 리턴
- ✓ getHostAddress(): ip주소를 문자열로 리턴
- ✓ toString(): ip 주소를 문자열로 변환하여 리턴

실습(IPDisplay.java)

```
import java.net.*;
public class IPDisplay{
        public static void main(String[] ar) {
                InetAddress ia = null;  //주소를 얻기 위한  클래스
                InetAddress []ia2 = null;
                try {
                        ia = InetAddress.getByName("www.daum.net");
                System.out.println("server HostAddress : " + ia.getHostAddress());
                        System.out.println("server HostName : " + ia.getHostName());
                        System.out.println("--------------");
                        ia = InetAddress.getLocalHost();
                System.out.println("local HostAddress : " + ia.getHostAddress());
                        System.out.println("local HostName : " + ia.getHostName());
                        System.out.println("--------------");
                        ia2 = InetAddress.getAllByName("www.daum.net");
                        for(InetAddress aa:ia2){
                                System.out.println(aa.getHostAddress());
                                System.out.println(aa.getHostName());
                        }
                }
                catch (UnknownHostException ee){
                        System.out.println("해당 사이트는 유효하지 않습니다.");
                }
        }
}
```

Socket

- Socket
  - ✓ 네트워크 어댑터를 추상화한 것
  - ✓ Stream Socket과 Datagram Socket을 제공
  - ✓ Socket은 프로세스간의 통신을 담당하며 InputStream과 OutputStream을 가지고 있어서 이 2개의 Stream을 이용해서 통신
  - ✓ Stream Socket은 TCP 프로토콜을 이용하고 Datagram Socket은 UDP 프로토콜을 사용
  - ✓ TCP는 연결형 프로토콜이고 UDP는 비연결형 프로토콜
  - ✓ 생성자
    - ❑ Socket(): 기본 생성자
    - ❑ Socket(InetAddress addr, int port): addr의 port에 접속을 시도
    - ❑ Socket(String addr, int port): addr의 port에 접속을 시도
    - ❑ Socket(InetAddress addr, int port, InetAddress localAddr, int port): addr의 port에 접속을 시도하고 자신의 주소와 포트를 명시
  - ✓ 주소가 잘못되면 NullPointerException 이 발생하고 port 번호가 잘못되면 IllegalArgumentException이 발생

Socket

- Socket
  - 메서드
    - void close()
    - InetAddress getInetAddress()
    - int getPort()
    - InetAddress getLocalAddress()
    - int getLocalPort()
    - InputStream getInputStream()
    - OutputStream getOutputStream()
    - boolean getKeepAlive(), void setKeepAlive(boolean b)
    - int getReceiveBufferSize(), void setReceiveBufferSize(int a)
    - int getSendBufferSize(), void setSendBufferSize(int a)
    - int getSoTimeout(), void setSoTimeout(int a)
    - boolean isBound(), isClosed(), isConnected(), isInputShutdown(), isOutputShutdown()
    - void shutdownInput(), shutdownOutput()

Stream Socket

- Stream Socket
  - ✓ TCP 프로토콜을 이용하는 Socket
  - ✓ 상대방에게 연결을 요청하고 상대방이 허락하면 데이터를 전송하고 데이터의 수신 여부를 송신하는 측에 알려주는 방식
  - ✓ 연결을 한 상태에서 통신을 하므로 안정적이지만 연결을 유지하는 트래픽 비용이 발생한다는 단점이 있음
  - ✓ 클라이언트 Socket 생성 및 사용방법

    Socket socket = new Socket("접속할 컴퓨터의 ip", 포트번호);
  - ✓ Socket 닫기

    socket.close();

Stream Socket

❖ Socket을 이용한 데이터 입출력(바이트 단위)

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

InputStream in = socket.getInputStream();
int data = in.read()
```

❖ Socket을 이용한 데이터 입출력(문자 단위)

```
PrintWriter out = new PrintWriter(new BufferedWriter(new
OutputStreamWriter(socket.getOutputStream())));
out.println("메시지");

BufferedReader in = new BufferedReader(new
InputStreamReader(socket.getInputStream()));
String str = in.readLine();
```

실습(WebDownload.java)

```
import java.io.*;
import java.net.*;

public class WebDownload {
        public static void main(String[] ar) {
                InetAddress ia = null;
                Socket sock = null;
                PrintWriter out = null;
                BufferedReader in = null;

                try {
                        ia = InetAddress.getByName("www.daum.net");
                        sock = new Socket(ia, 80);
                        out = new PrintWriter(new BufferedWriter(new
OutputStreamWriter(sock.getOutputStream())));
                        out.println("GET http://www.daum.net");
                        out.flush();
                        in = new BufferedReader(new
InputStreamReader(sock.getInputStream()));
```

실습(WebDownload.java)

```
                    while (true) {
                        String str = in.readLine();

                        if (str == null)
                            break;
                        System.out.println(str);
                    }
                    in.close();
                    out.close();
                    sock.close();
                } catch (IOException e) {
                    System.out.println("데이터 가져오기 실패 : " + e.getMessage());
                }
            }
}
```

Stream Socket

- 클라이언트 – 서버 모델
  - ✓ TCP 포트를 개방해서 그곳으로 요청을 받았을 때 사용하는 클래스가 ServerSocket
  - ✓ ServerSocket의 생성자
    - ❑ ServerSocket()
    - ❑ ServerSocket(int port):포트 개방
    - ❑ ServerSocket(int port, int backlog):포트를 개방하고 클라이언트 개수를 설정하는 Socket
  - ✓ ServerSocket의 메서드
    - ❑ Socket accept(): 바인딩 된 포트로부터 Socket정보를 가져오는데 클라이언트가 접속할 때 까지 블럭
    - ❑ void close()
    - ❑ InetAddress getInetAddress()
    - ❑ int getLocalPort()
    - ❑ int getReceiveBufferSize(), void setReceiveBufferSize(int n)
    - ❑ int getSoTimeout(), void setSoTimeout(int n)
    - ❑ boolean isBound(), isClosed()

Stream Socket

- 서버 프로그램과 클라이언트 프로그램 간의 통신 과정
  - ✓ 서버 측에서 서버 Socket을 생성해서 서버 컴퓨터의 특정 포트에서 클라이언트의 연결요청을 처리할 준비
  - ✓ 클라이언트 프로그램은 접속할 서버의 IP 와 포트 정보를 가지고 Socket을 생성해서 서버에 연결을 요청
  - ✓ 서버 Socket은 클라이언트의 연결 요청을 받으면 서버에 새로운 Socket을 생성해서 클라이언트의 Socket과 연결
  - ✓ 서버와 클라이언트 일대일 통신 가능

- 서버 Socket 생성 및 사용방법

```
ServerSocket serverSocket = new ServerSocket(9000);
Socket socket = serverSocket.accept();
```

실습(TCPServer.java)

```
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.net.ServerSocket;
import java.net.Socket;

public class TCPServer {
        public static void main(String[] args) {
                ServerSocket ss = null;
                Socket sock = null;
```

```
서버 대기중...
접속자 정보 : Socket[addr=/127.0.0.1,port=51311,localport=9999]
전송된 내용 : 안녕하세요
접속자 정보 : Socket[addr=/127.0.0.1,port=51312,localport=9999]
```

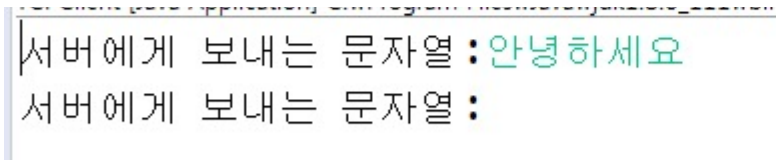
실습(TCPServer.java)

```
                try {
                        ss = new ServerSocket(9999);
                        System.out.println("서버 대기중...");
                        while (true) {
                                sock = ss.accept();
                                System.out.println("접속자 정보 : " + sock.toString());
                                BufferedReader in = new BufferedReader(new
InputStreamReader(sock.getInputStream()));
                                String str = in.readLine();
                                System.out.println("전송된 내용 : " + str);
                                in.close();
                                sock.close();
                        }
                }
                catch (IOException e) {
                        System.out.println("해당 포트 사용중!!!");
                        try {
                                ss.close();
                        } catch (Exception ex) {
                        }
                }
        }
}
```

실습(TCPClient.java)

```
import java.io.IOException;
import java.io.PrintWriter;
import java.net.InetAddress;
import java.net.Socket;
import java.util.Scanner;

public class TCPClient {
        public static void main(String[] args) {
                InetAddress ia = null;
                Socket sock = null;
                PrintWriter out = null;
                Scanner sc = new Scanner(System.in);
```



실습(TCPClient.java)

```
                try {
                        while (true) {
                                ia = InetAddress.getByName("127.0.0.1");
                                sock = new Socket(ia, 9999);
                                out = new PrintWriter(sock.getOutputStream(), true);
                                System.out.print("서버에게 보내는 문자열:");
                                String msg = sc.nextLine();
                                out.println(msg + "₩n");
                                out.flush();
                                out.close();
                        }
                } catch (IOException e) {
                        System.out.println("접속 오류 : " + e.toString());
                }finally {
                        sc.close();
                }
        }
}
```

Datagram Socket

❖UDP(Datagram) 전송

- ✓ 세션 설정 없이 IP 주소만으로 데이터를 전송하는 방식
- ✓ 수신 상태를 확인할 수 없어서 데이터의 전송여부를 확인할 수 없는 방식
- ✓ DatagramPacket 클래스와 DatagramSocket 클래스를 이용
- ✓ DatagramSocket 클래스의 생성자
  - ❑ DatagramSocket()
  - ❑ DatagramSocket(int port)
  - ❑ DatagramSocket(int port, InetAddress addr)
- ✓ DatagramSocket 클래스의 메서드
  - ❑ void close()
  - ❑ void connect(InetAddress addr, int port)
  - ❑ void disconnect()
  - ❑ void receive(DatagramPacket Dp): 데이터 받기
  - ❑ void send(DatagramPacket Dp): 데이터 보내기

Datagram Socket

- UDP(Datagram) 전송
  - ✓ DatagramPacket 클래스의 생성자
    - ❑ DatagramPacket(byte[] buf, int length): buf에 length 만큼 전송받기 위한 생성자
    - ❑ DatagramPacket(byte[] buf, int length, InetAddress addr, int port): 전송하기 위한 생성자
  - ✓ DatagramPacket의 메서드
    - ❑ byte[] getData: 데이터 내용
    - ❑ int getLength(): 데이터 길이

실습(UDPServer.java)

```
import java.net.DatagramPacket;
import java.net.DatagramSocket;

public class UDPServer {
	public static void main(String[] args) {
		try (DatagramSocket dsoc = new DatagramSocket(7777);) {
			byte[] data = new byte[65536];

			DatagramPacket dp = new DatagramPacket(data, data.length);
			System.out.println("서버 서비스 시작...");
			while (true) {
				dsoc.receive(dp);
				System.out.println("------보낸 곳 주소 : " +
dp.getAddress().getHostAddress());
				System.out.println("자료 크기 : " + dp.getLength());
				String message = new String(dp.getData()).trim();
				System.out.println("내용 : " + message);
			}
		} catch (Exception e) {
			System.out.println("오류 : " + e);
		}
	}
}
```

실습(UDPClient.java)

```
import java.net.*;
import java.io.*;

public class UDPClient {
        public static void main(String[] args) throws Exception {
                BufferedReader in = new BufferedReader(new
InputStreamReader(System.in));
                System.out.print("보낼 자료 입력 : ");
                String msg = in.readLine();

                DatagramSocket dsoc = new DatagramSocket();

                InetAddress ia = InetAddress.getByName("127.0.0.1");
                DatagramPacket dp = new DatagramPacket(msg.getBytes(),
                                msg.getBytes().length, ia, 7777);
                dsoc.send(dp);
                System.out.println("전송완료");
                dsoc.close();
        }
}
```

Datagram Socket

- Multicast
  - ✓ 네트워크에 접속한 모든 클라이언트들에게 데이터를 전송하는 방식
  - ✓ IPv4에서는 224.0.0.0 – 239.255.255.255까지의 D 클래스 주소를 이용(IPv6 – ff00::/8)
  - ✓ 화상회의 등에서 주로 이용
  - ✓ MulticastSocket 클래스를 이용하며 전송방식은 UDP
  - ✓ 생성자
    - ❑ MulticastSocket()
    - ❑ MulticastSocket(int port)
  - ✓ 메서드
    - ❑ joinGroup(InetAddress addr): 멀티캐스트 그룹에 참여
    - ❑ leaveGroup(InetAddress addr): 멀티캐스트 그룹에서 이탈

실습(MulticastReceiver.java)

```
import java.io.IOException;
import java.net.DatagramPacket;
import java.net.DatagramSocket;
import java.net.InetAddress;
import java.net.InetSocketAddress;
import java.net.MulticastSocket;
import java.net.NetworkInterface;
import java.net.SocketException;

public class MulticastReceiver {
        public static void main(String[] args) {
                DatagramPacket packet = null;
                MulticastSocket socket = null;
                try {
                        socket = new MulticastSocket(9006);
                        System.out.printf("멀티캐스트 수신자 생성₩n");
                        // 그룹에 조인(라우터가 보냄)
                        InetAddress address = InetAddress.getByName("224.128.1.5");
// 멀티 캐스트를 위한 아이피 설정
                        socket.joinGroup(address);
```

실습(MulticastReceiver.java)

```
                        byte[] buf = new byte[512];
                        packet = new DatagramPacket(buf, buf.length);
                        while (true) {
                                // 패킷 수신
                                socket.receive(packet);
                                String msg = new String(packet.getData(), 0,
packet.getLength());

                                System.out.printf("%s₩n", msg);
                        }
                } catch (SocketException e) {
                        e.printStackTrace();
                } catch (IOException e) {
                        e.printStackTrace();
                }
        }
}
```

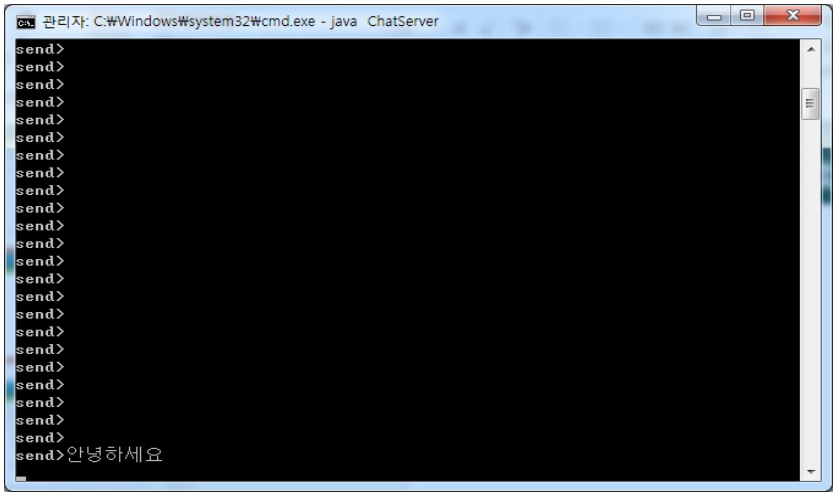
실습(MulticastSender.java)

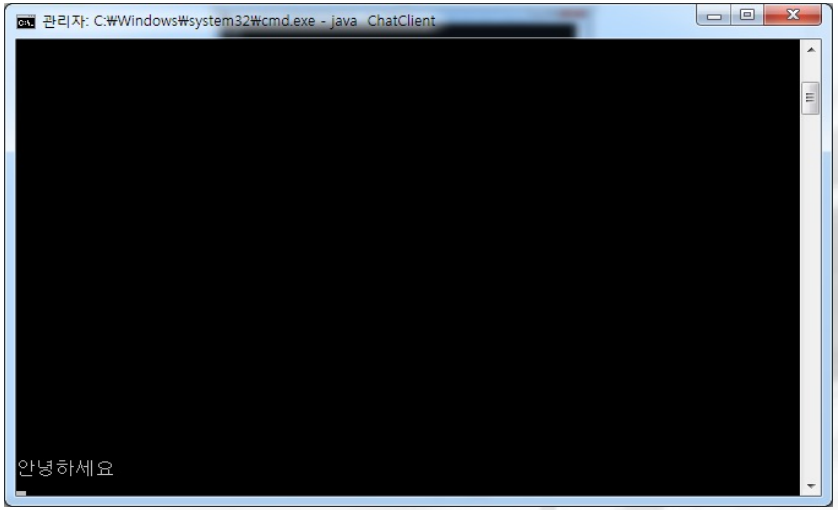
```
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.net.DatagramPacket;
import java.net.InetAddress;
import java.net.MulticastSocket;
public class MulticastSender {
		public static void main(String[] args) {
			DatagramPacket packet = null;
			MulticastSocket socket = null;
			try {
				socket = new MulticastSocket();
				System.out.println("서버 생성 성공.");
				BufferedReader reader = new BufferedReader(new
InputStreamReader(System.in));
				InetAddress address = InetAddress.getByName("224.128.1.5");
// 멀티캐스트 방식으로 서버 주소를 설정함.
				System.out.printf("대화명 입력 : ");
				String nickname = reader.readLine();
```

실습(MulticastSender.java)

```
                while (true) {
                        System.out.printf("메시지 : ");
                        String msg = reader.readLine();
                        if (msg.equals("종료")) {
                                msg = nickname + "님이 퇴장하셨습니다.";
                                packet = new
DatagramPacket(msg.getBytes(), msg.getBytes().length, address, 9006);
                                socket.send(packet);
                                break;
                        } else {
                                msg = nickname + " > " + msg;
                                packet = new
DatagramPacket(msg.getBytes(), msg.getBytes().length, address, 9006);
                                socket.send(packet);
                        }
                }
                socket.close();
        } catch (IOException e) {
                e.printStackTrace();
        }
    }
}
```

실습(1:1 채팅)


관리자: C:₩Windows₩system32₩cmd.exe - java ChatServer
send>
send>
send>
send>
send>
send>
send>
send>
send>
send>
send>
send>
send>
send>
send>
send>
send>
send>
send>
send>
send>
send>
send>
send>안녕하세요


관리자: C:₩Windows₩system32₩cmd.exe - java ChatClient
안녕하세요

실습(1:1 채팅 – 받는 스레드)

```
import java.io.BufferedReader;
import java.io.InputStreamReader;
import java.net.Socket;

class ReceiveThread extends Thread {
    Socket socket;
    public ReceiveThread(Socket socket) {
        this.socket = socket;
    }
    public void run() {
        try {
            BufferedReader reader = new BufferedReader(
                new InputStreamReader(socket.getInputStream()));
            while (true) {
                String str = reader.readLine();
                if (str == null)
                    break;
                System.out.println("수신>" + str);
            }
        }
        catch (Exception e) {
            System.out.println(e.getMessage());
        }
    }
}
```

실습(1:1 채팅 – 보내는 스레드)

```
import java.io.*;
import java.net.*;
import java.util.Scanner;

class SendThread extends Thread {
        Socket socket;

        public SendThread(Socket socket) {
                this.socket = socket;
        }

        public void run() {
                Scanner scanner = new Scanner(System.in);
                try {
                        PrintWriter writer = new PrintWriter(socket.getOutputStream());
                        while (true) {
                                System.out.print("전송 메시지:");
                                String str = scanner.nextLine();
                                if (str.equals("bye"))
                                        break;
                                writer.println(str);
                                writer.flush();
                        }
                }
```

실습(1:1 채팅 – 보내는 스레드)

```
                    catch (Exception e) {
                            System.out.println(e.getMessage());
                    } finally {
                            try {
                                    scanner.close();
                                    socket.close();
                            } catch (Exception e) {
                            }
                    }
            }
}
```

실습(1:1 채팅 – ChatServer)

```
import java.net.*;

public class ChatServer {
        public static void main(String[] args) {
                ServerSocket serverSocket = null;
                Socket socket = null;
                try {
                        serverSocket = new ServerSocket(9001);
                        socket = serverSocket.accept();
                        Thread thread1 = new SendThread(socket);
                        Thread thread2 = new ReceiveThread(socket);
                        thread1.start();
                        thread2.start();
                } catch (Exception e) {
                        System.out.println(e.getMessage());
                } finally {
                        try {
                                serverSocket.close();
                        } catch (Exception ignored) {
                        }
                }
        }
}
```

실습(1:1 채팅 – ChatClient)

```
import java.net.*;

public class ChatClient {
        public static void main(String[] args) {
            Socket socket = null;
            try {
                socket = new Socket("127.0.0.1", 9001);
                Thread thread1 = new SendThread(socket);
                Thread thread2 = new ReceiveThread(socket);
                thread1.start();
                thread2.start();
            }
            catch (Exception e) {
                System.out.println(e.getMessage());
            }
        }
}
```

URL 통신

- HTTP 요청
  - HTTP는 웹의 기본 프로토콜
  - HTTP 클라이언트는 원하는 정보를 URL로 전송
  - URL은 인터넷상의 자원이나 서비스의 유일한 한 지점을 가리키는 주소
  - RFC 1738에 의해 포맷이 정의되어 있으며 프로토콜, 호스트 이름, 포트, 경로 , 파일 등의 기본 정보와 쿼리, 사용자명, 인증 정보 등으로 구성
  - URL 클래스의 생성자
    - URL (String spec)
    - URL (String protocol, String host, int port, String file)

URL 통신

- URL 클래스의 메서드
  - 다음 메서드는 URL의 각 부분을 분리

메서드	리턴	설명
String getProtocol ()	http	통신 방법을 정의하는 프로토콜
int getDefaultPort ()	80	프로토콜이 정의하는 디폴트 포트
int getPort ()	-1	URL에 정의된 포트. 없을 경우 -1이 리턴
String getHost ()	www.chatting.com	서버 주소
String getFile ()	/showroom?girl=3	Path와 쿼리
String getPath ()	/showroom	서버내의 경로
String getQuery ()	girl=3	서버로 전달되는 쿼리 변수 값

  - 규칙에 맞지 않은 무효한 URL을 설정하면 MalformedURL.Exception 예외가 발생하며 접속할 주소가 결정되면 URL 클래스의 다음 메서드로 접속
    - URLConnection openConnection ([Proxy proxy])
  - 접속에 성공했으면 지정한 주소로 양방향 통신이 가능한 연결 객체가 리턴

URL 통신

- HTTP 요청
  - ✓ URLConnection 자체는 추상 클래스이며 프로토콜에 따라 HTTP, HTTPs, Jar 등의 연결 객체가 리턴
  - ✓ 요청한 프로토콜에 따라 연결 객체의 타입이 달라지는데 통상적인 웹 주소라면 HttpURLConnection 객체가 리턴되므로 이 타입으로 강제 형변환해서 사용

메서드	설명
setConnectTimeout (int timeout)	연결 제한 시간을 1/1000초 단위로 지정 0이면 무한 대기
setReadTimeout (int timeout)	읽기 제한 시간을 지정 0이면 무한 대기
setUseCaches (boolean newValue)	캐시 사용 여부를 지정
setDoInput (boolean newValue)	입력을 받을 것인지를 지정
setDoOutput (boolean newValue)	출력을 할 것인지를 지정
setRequestProperty (String field, String newValue)	요청 헤더에 값을 설정
addRequestProperty (String field, String newValue)	요청 헤더에 값을 추가 속성의 이름이 같더라도 덮어 쓰지는 않음

URL 통신

- HTTP 요청
  - ✓ 기본 속성을 설정한 후 각 연결 타입 별로 속성을 추가로 설정
  - ✓ HTTP 연결의 경우 요청 방식을 지정해야 하고 다음 메서드로 “GET”, “POST”등의 방법을 지정하되 별 지정이 없으면 디폴트인 “GET”방식이 적용
    - ❑ void HttpURLConnection.setRequestMethod (String method)
  - ✓ 모든 속성 설정이 완료되었으면 다음 메서드로 서버에 요청을 보내는데 이 메서드는 원격지의 서버로 명령을 보내고 응답 결과를 받음
    - ❑ int getResponseCode ()
  - ✓ 요청이 무사히 전달되었으면 HTTP_OK(200)가 리턴
  - ✓ URL이 발견되지 않으면 HTTP_NOT_FOUND(404)가 리턴되며 인증에 실패하면 HTTP_UNAUTHORIZED(401) 에러 코드가 리턴

URL 통신

- 비동기 다운로드
  - ✓ 네트워크는 불확실성이 높은데다 모바일에서는 접속 상태도 신뢰성이 없어 에러 발생 가능성이 높음
  - ✓ 네트워크 프로그램은 멀티 스레드 프로그램으로 작성해야 하며 멀티 스레드를 사용해야 할 가장 전형적인 예
  - ✓ 메인 스레드는 다운로드 시작만 명령하고 실질적인 네트워크 액세스는 분리된 작업 스레드에서 수행
  - ✓ 다운로드를 받는 작업 스레드를 별도로 작성해야 하며 일반적으로 스레드의 run 메서드에서 네트워크 액세스를 수행

스레드를 이용한 HTTP 통신

스레드를 이용한 HTTP 통신

1.다운로드에 사용할 스레드 클래스(ConnectThread.java)

```
import java.io.*;
import java.net.*;


public class ConnectThread extends Thread {
        String urlStr;

        public ConnectThread(String inStr) {
                urlStr = inStr;
        }

        public void run() {

                try {
                        final String output = request(urlStr);
                        System.out.println("가져온 데이터:" + output);
                } catch (Exception ex) {
                        ex.printStackTrace();
                }
        }
```

스레드를 이용한 HTTP 통신

```
private String request(String urlStr) {
        StringBuilder html = new StringBuilder();
        try {
                URL url = new URL(urlStr);
                HttpURLConnection conn = (HttpURLConnection)
url.openConnection();
                if (conn != null) {
                        conn.setConnectTimeout(10000);
                        conn.setUseCaches(false);
                        if (conn.getResponseCode() ==
HttpURLConnection.HTTP_OK) {
                                BufferedReader br = new
BufferedReader(new InputStreamReader(conn.getInputStream()));
                                while (true) {
                                        String line = br.readLine();
                                        if (line == null)
                                                break;
                                        html.append(line + '₩n');
                                }
                                br.close();
                        }
                        conn.disconnect();
                }
        }
```

스레드를 이용한 HTTP 통신

```
            catch (Exception e) {
                        System.out.printf("SampleHTTPClient",
                        "Exception in processing response.", e);
                        }
                        return html.toString();
            }
}
```

스레드를 이용한 HTTP 통신

2. 스레드를 시작시키는 클래스(AsyncDownloadMain.java)

```
public class AsyncDownloadMain {
        public static void main(String[] args) {
                ConnectThread mThread = new ConnectThread("https://www.naver.com");
                mThread.start();
        }
}
```

이미지 파일 다운로드

이미지 파일 다운로드

❖ 다운로드를 처리할 스레드 클래스(ImageDownloadThread.java)

```
import java.io.*;
import java.net.*;

public class ImageDownloadThread extends Thread {
        String mAddr;
        String mFile;

        ImageDownloadThread(String addr, String filename) {
                mAddr = addr;
                mFile = filename;
                System.out.println(mFile);
        }
        public void run() {
                URL imageurl;
                int read;
                try {
                        imageurl = new URL(mAddr);
                        HttpURLConnection conn=
(HttpURLConnection)imageurl.openConnection();
```

이미지 파일 다운로드

```
                    int len = conn.getContentLength();
                    byte[] raster = new byte[len];
                    InputStream is = conn.getInputStream();
                    FileOutputStream fos = new FileOutputStream(mFile);
                    while(true) {
                            read = is.read(raster);
                            System.out.println(read);
                            if (read <= 0) {
                                    break;
                            }
                            fos.write(raster,0, read);
                    }

                    is.close();
                    fos.close();
                    conn.disconnect();
            } catch (Exception e) {
                    mFile = null;
            }
        }
}
```

이미지 파일 다운로드

❖ 스레드를 시작시킬 클래스(ImageMain.java)

```
import java.io.*;

public class ImageMain {
        public static void main(String[] args) {
                String imageurl =
"http://www.onlifezone.com/files/attach/images/962811/376/321/005/2.jpg";
                int idx = imageurl.lastIndexOf('/');
                String localimage = imageurl.substring(idx + 1);
                String path = "./" + localimage;

                if (new File(path).exists()) {
                        System.out.println("image is exist");
                } else {
                        System.out.println("image is not exist");
                        (new ImageDownloadThread(imageurl, path)).start();
                }
        }
}
```

URL 통신

- 파라미터 설정
  - ✓ GET 방식: 파라미터를 URL 뒤에 ?를 추가하고 이름=값& 형태로 이어서 붙이는 방식
    - URL 뒤에 인코딩을 해서 부착하면 됨

URL 통신

- 파라미터 설정
  - ✓ POST 방식

```
//파일을 제외한 파라미터 만들기
String[] data = {파라미터 값 나열 };
String[] dataName = {파라미터 값 나열};
String lineEnd = "₩r₩n";
// boundary생성, 여기서는 고정값이지만 되도록이면 실행할 때마다 다른 값을 할당
String boundary = UUID.randomUUID().toString();
//연결 옵션 설정
con.setRequestMethod("POST");
```

URL 통신

- 파라미터 설정
  - ✓ POST 방식

```
//파일 첨부가 있는 경우 생성
con.setRequestProperty("ENCTYPE", "multipart/form-data");
con.setRequestProperty("Content-Type","multipart/form-data;boundary="+boundary);

String delimiter = "--" + boundary + lineEnd; // --androidupload₩r₩n
StringBuffer postDataBuilder = new StringBuffer();
//파라미터를 하나로 만들기
for(int i=0;i<data.length;i++){
    postDataBuilder.append(delimiter);
    postDataBuilder.append("Content-Disposition: form-data; name=₩"" + dataName[i]
+"₩""+lineEnd+lineEnd+data[i]+lineEnd);
}
```

URL 통신

- 파라미터 설정
  - ✓ POST 방식

```
// 파일 파라미터 생성
String fileName = "파일명";
if(fileName!=null){
    postDataBuilder.append(delimiter);
    postDataBuilder.append("Content-Disposition: form-data; name=₩"" + "profile" +
"₩";filename=₩"" + fileName +"₩"" + lineEnd);
}
//파라미터 전송
DataOutputStream ds = new DataOutputStream(con.getOutputStream());
ds.write(postDataBuilder.toString().getBytes());
```

URL 통신

- 파라미터 설정
  - ✓ POST 방식

```
//파일 업로드
if(fileName!=null){
    ds.writeBytes(lineEnd);
    InputStream fres = getResources().openRawResource(R.raw.ball);
    byte[] buffer = new byte[fres.available()];
    int length = -1;
    while((length=fres.read(buffer)) != -1){
            ds.write(buffer,0,length);
    }
    ds.writeBytes(lineEnd);
    ds.writeBytes(lineEnd);
    ds.writeBytes("--" + boundary + "--" + lineEnd); // requestbody end
    fres.close();
}else {
    ds.writeBytes(lineEnd);
    ds.writeBytes("--" + boundary + "--" + lineEnd); // requestbody end
}
ds.flush();
ds.close();
```

URL 통신

- URL 인코딩
  - ✓ URLEncoding 과 URLDecoding 은 java.net.URLEncoder.encode(String, String) 과 URLDecoder.decode 메서드를 이용 – 첫번째 매개변수는 변경할 문자열이고 두번째 문자열은 인코딩 방식
  - ✓ 일반 문자열은 String 의 getBytes() 와 String(byte[], String) 을 이용

Kakao Open API 사용

- KakaoAPI.java

```
import java.io.BufferedReader;
import java.io.InputStreamReader;
import java.net.HttpURLConnection;
import java.net.URL;
import java.net.URLEncoder;

public class KakaoAPI {

	public static void main(String[] args) {
		String json = null;
		try {
			String urlAddr =
"https://dapi.kakao.com/v3/search/book?target=title&query=";
			//파라미터 인코딩
			String title = "삼국지";
			urlAddr += URLEncoder.encode(title, "utf-8");
			URL url = new URL(urlAddr);
			HttpURLConnection conn = (HttpURLConnection)
url.openConnection();
			//요청 헤더 설정
			conn.setRequestProperty("Authorization", "KakaoAK
ae6b2875ee4452804ed4e01890078a7e");
```

Kakao Open API 사용

- KakaoAPI.java

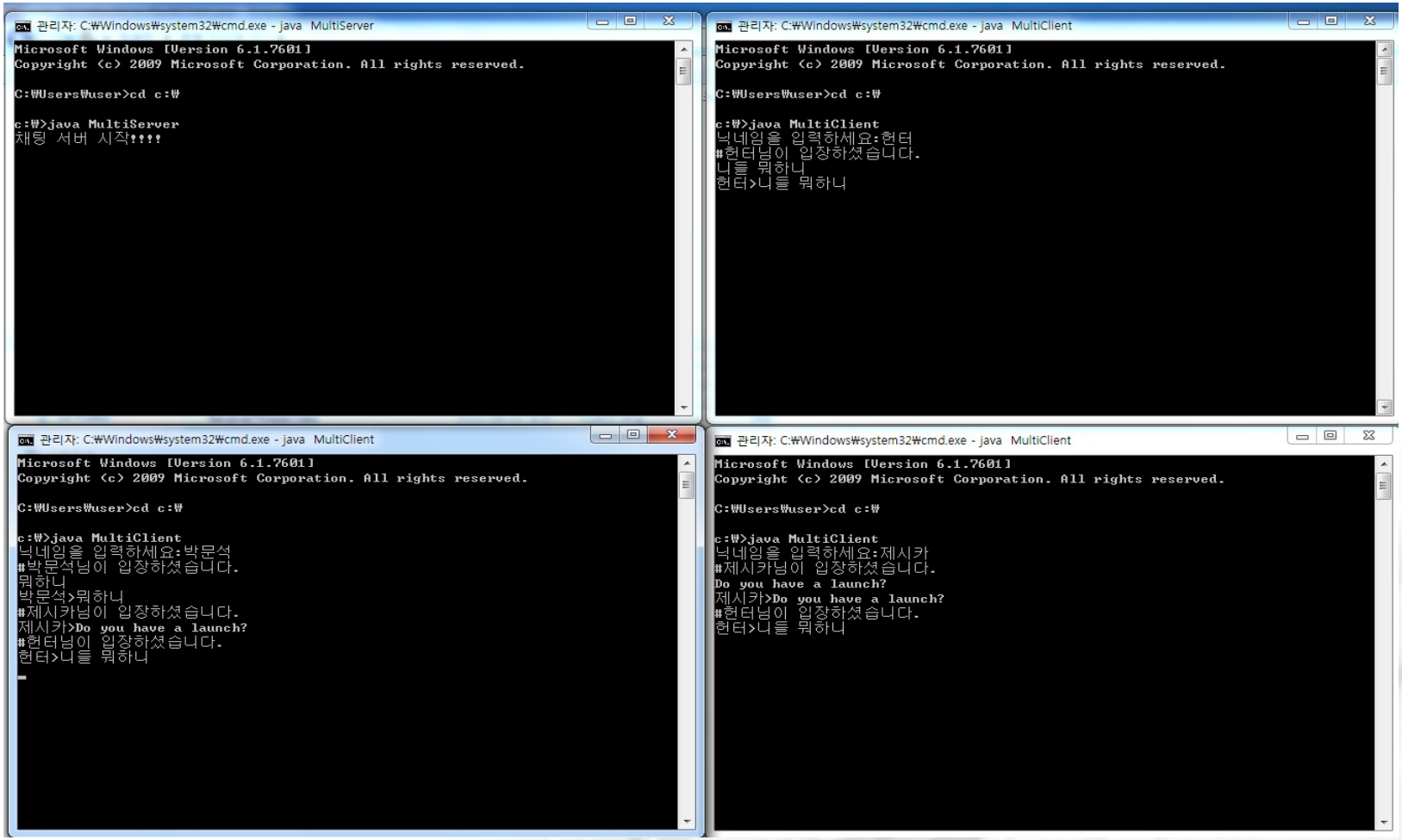
```
StringBuilder sb = new StringBuilder();
if (conn != null) {
        conn.setConnectTimeout(20000);
        conn.setUseCaches(false);
        if (conn.getResponseCode() ==
HttpURLConnection.HTTP_OK) {
                InputStreamReader isr = new
InputStreamReader(conn.getInputStream());
                BufferedReader br = new
BufferedReader(isr);
                while (true) {
                        String line = br.readLine();
                        if (line == null) {
                                break;
                        }
                        sb.append(line);
                }
                br.close();
                conn.disconnect();
        }
}
```

Kakao Open API 사용

- KakaoAPI.java

```
                        json = sb.toString();
                        System.out.printf("%s₩n", json);
                } catch (Exception e) {
                        System.out.printf("다운로드 중 에러 발생:%s₩n",
e.getMessage());
                }
        }
}
```

실습(멀티 채팅)


관리자: C:₩Windows₩system32₩cmd.exe - java MultiServer
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation. All rights reserved.
C:₩Users₩user>cd c:₩
c:₩>java MultiServer
채팅 서버 시작!!!!
관리자: C:₩Windows₩system32₩cmd.exe - java MultiClient
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation. All rights reserved.
C:₩Users₩user>cd c:₩
c:₩>java MultiClient
닉네임을 입력하세요:헌터
#헌터님이 입장하셨습니다.
니들 뭐하니
헌터>니들 뭐하니
관리자: C:₩Windows₩system32₩cmd.exe - java MultiClient
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation. All rights reserved.
C:₩Users₩user>cd c:₩
c:₩>java MultiClient
닉네임을 입력하세요:박문석
#박문석님이 입장하셨습니다.
뭐하니
박문석>뭐하니
#제시카님이 입장하셨습니다.
제시카>Do you have a launch?
#헌터님이 입장하셨습니다.
헌터>니들 뭐하니
관리자: C:₩Windows₩system32₩cmd.exe - java MultiClient
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation. All rights reserved.
C:₩Users₩user>cd c:₩
c:₩>java MultiClient
닉네임을 입력하세요:제시카
#제시카님이 입장하셨습니다.
Do you have a launch?
제시카>Do you have a launch?
#헌터님이 입장하셨습니다.
헌터>니들 뭐하니

실습(멀티 채팅 – Client)

1.클라이언트에서 문자열을 전송하는 스레드 클래스(ChatSendThread.java)

```
import java.net.*;
import java.io.*;
class ChatSendThread extends Thread {
    Socket socket;
    String name;
    ChatSendThread(Socket socket, String name) {
        this.socket = socket;
        this.name = name;
    }
    public void run() {
        try {
            BufferedReader reader = new BufferedReader(new InputStreamReader(System.in));
            PrintWriter writer = new PrintWriter(socket.getOutputStream());
            writer.println(name);
            writer.flush();
            while (true) {
                String str = reader.readLine();
                if (str.equals("bye"))
                    break;
                writer.println(str);
                writer.flush();
            }
        }
```

실습(멀티 채팅 – Client)

```
        catch (Exception e) {
            System.out.println(e.getMessage());
        }
        finally {
            try {
                socket.close();
            }
            catch (Exception ignored) {
            }
        }
    }
}
```

실습(멀티 채팅 – Client)

2.클라이언트에서 문자열을 전송받는 스레드 클래스(ChatReceiveThread.java)

```
import java.io.*;
import java.net.*;
class ChatReceiveThread extends Thread {
    Socket socket;
    ChatReceiveThread(Socket socket) {
        this.socket = socket;
    }
    public void run() {
        try {
            BufferedReader reader = new BufferedReader(new
InputStreamReader(socket.getInputStream()));
            while (true) {
                String str = reader.readLine();
                if (str == null)
                    break;
                System.out.println(str);
            }
        }
        catch (IOException e) {
            System.out.println(e.getMessage());
        }
    }
}
```

실습(멀티 채팅 – Client)

3.클라이언트 실행 클래스(ChatMultiClient.java)

```
import java.io.*;
import java.net.*;

class ChatMultiClient {
	public static void main(String[] args) {

		try {
			BufferedReader in = new BufferedReader(new
InputStreamReader(
					System.in));
			System.out.print("닉네임을 입력하세요:");
			String nick = in.readLine();
			Socket socket = new Socket("127.0.0.1", 9002);
			Thread thread1 = new ChatSendThread(socket, nick);
			Thread thread2 = new ChatReceiveThread(socket);
			thread1.start();
			thread2.start();
		} catch (Exception e) {
			System.out.println(e.getMessage());
		}
	}
}
```

실습(멀티 채팅 – Server)

1.서버에서 클라이언트를 1개를 생성하는 클래스(ChatPerClientThread.java)

```
import java.io.*;
import java.net.*;
import java.util.*;

class ChatPerClientThread extends Thread {
    static List<PrintWriter> list = Collections.synchronizedList(new ArrayList<PrintWriter>());
    Socket socket;
    PrintWriter writer;
    ChatPerClientThread(Socket socket) {
        this.socket= socket;
        try {
            writer = new PrintWriter(socket.getOutputStream());
            list.add(writer);
        }
        catch (Exception e) {
            System.out.println(e.getMessage());
        }
    }
```

실습(멀티 채팅 – Server)

```
public void run() {
        String name = null;
        try {
                BufferedReader reader = new BufferedReader(new
InputStreamReader(socket.getInputStream()));
                name = reader.readLine();
                sendAll("#" + name + "님이 입장하셨습니다.");
                while (true) {
                        String str = reader.readLine();
                        if (str == null)
                                break;
                        sendAll(name + ">" + str);
                }
        } catch (Exception e) {
                System.out.println(e.getMessage());
        } finally {
                list.remove(writer);
                sendAll("#" + name + "님이 퇴장하셨습니다.");
                try {
                        socket.close();
                } catch (Exception ignored) {
                }
        }
}
```

실습(멀티 채팅 – Server)

```
    private void sendAll(String str) {
        for (PrintWriter writer : list) {
            writer.println(str);
            writer.flush();
        }
    }
}
```

실습(멀티 채팅 – Server)

2.서버 실행 클래스(ChatMultiServer.java)

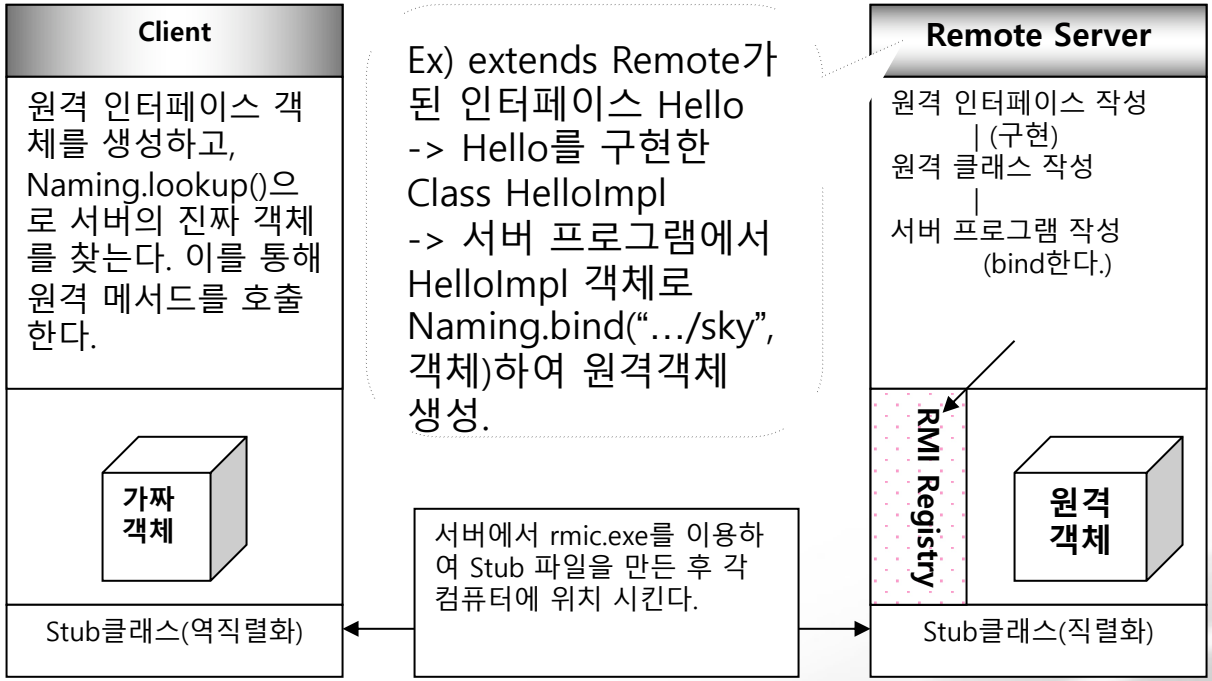
```
import java.net.*;
class ChatMultiServer {
    public static void main(String[] args) {
        ServerSocket serverSocket = null;
        try {
            serverSocket = new ServerSocket(9002);
            System.out.println("채팅 서버 시작!!!!");
            while (true) {
                Socket socket = serverSocket.accept();
                Thread thread = new ChatPerClientThread(socket);
                thread.start();
            }
        }
        catch (Exception e) {
            System.out.println(e.getMessage());
        }
    }
}
```

RMI

- RMI(Remote Method Invocation)
  - ✓ 원격 컴퓨터의 메서드(원격컴퓨터의 CPU 사용)를 호출하여 사용할 수 있는 기술
  - ✓ RMI는 소켓 통신을 내부적으로 수행 시키는 미들웨어(Middleware)
  - ✓ 내부에서 네트워크 통신방법을 제공하기 때문에 신뢰성이 보장
  - ✓ rmiregistry라는 서버 모델을 사용하면 간단하게 구현 가능
  - ✓ 여러 컴퓨터에 작업을 나눠주고 한 곳에서 작업을 처리하여 효율성을 증진시킬 수 있으며 포털 사이트에서 처럼 각 벤더들의 상품을 분산 객체 형식으로 서비스 가능

RMI

❖ RMI (Remote Method Invocation) : OOP 형태의 자바 전용 RPC 기술



❖ RMI Registry : 기본적인 서버 모델 제공 --응용모델-→ J2EE (상업용으로 WebLogic 등 다양한 제품)

RMI

- RMI
  - ✓ 작업 순서
    - Remote 인터페이스를 implements 하고 UnicastRemoteObject 클래스로부터 상속받는 클래스를 생성해서 생성자를 재정의해서 RemoteException을 처리하도록 하고 RMI로 동작할 메서드를 선언하며 이 메서드 또한 RemoteException을 처리
    - 앞에서 만든 클래스의 객체를 생성해서 호출할 수 있도록 바인딩해주는 서버 측 프로그램을 작성
    - 서버 준비
      - 자바가 제공하는 rmic를 이용하여 stub 클래스 객체를 생성
      - rmiregistry 를 실행: start rmiregistry
      - 서버 측 프로그램 실행
    - 클라이언트 준비

실습(서버 측 준비)

1. 원격 인터페이스를 생성(Hello.java)

```
import java.rmi.*;

public interface Hello extends Remote{
        public String sayHello(String name) throws RemoteException;
}
```

실습(서버 측 준비)

2. 원격 클래스를 생성(HelloImpl.java)

```
import java.rmi.*;
import java.rmi.server.*;

public class HelloImpl extends UnicastRemoteObject implements Hello {
        private static final long serialVersionUID = 1L;

        public HelloImpl() throws RemoteException {}

        public String sayHello(String name) throws RemoteException {
                System.out.println("Server Working Start");
                return name + " Welcome To MyServer";
        }
}
```

실습(서버 측 준비)

3. 서버 실행 클래스를 생성(HelloServer.java)

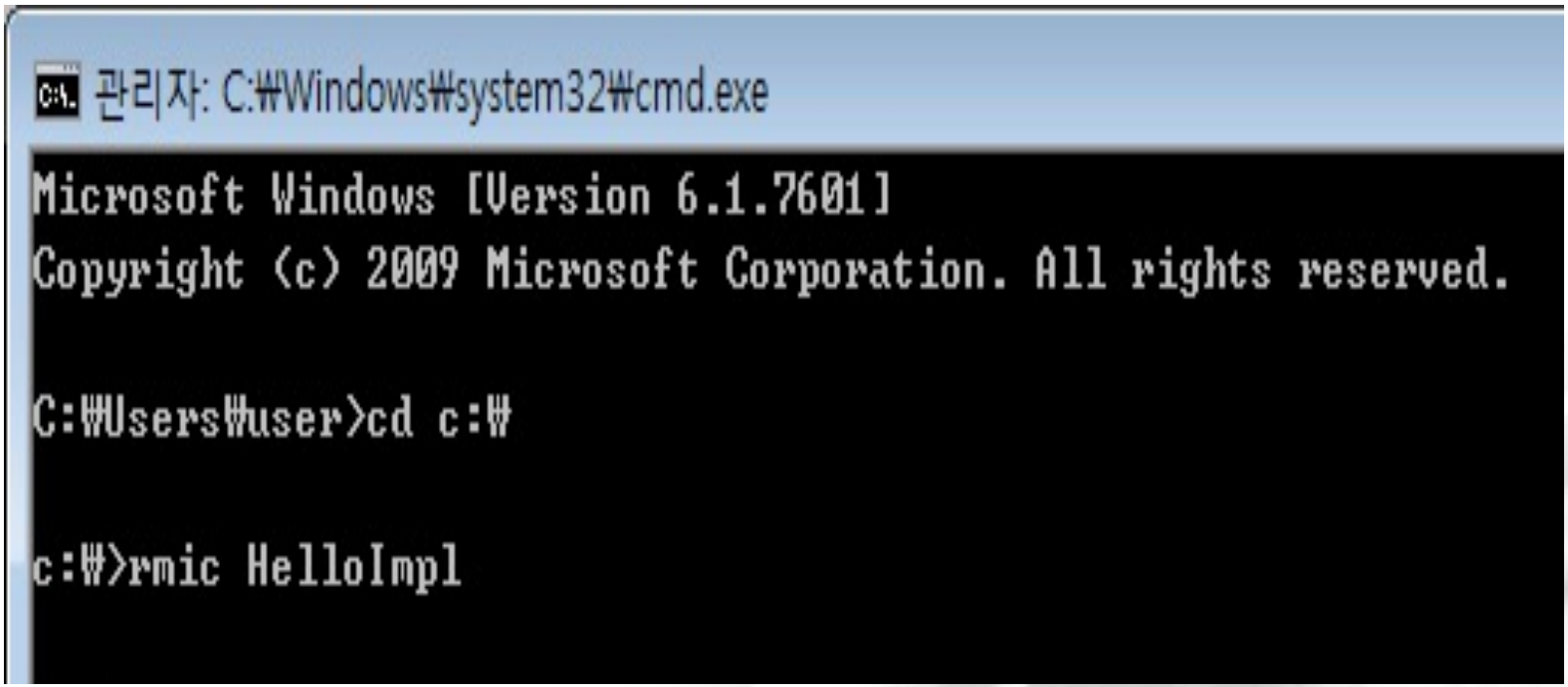
```
import java.rmi.*;
public class HelloServer {
        public static void main(String[] args) throws Exception{
                HelloImpl impl=new HelloImpl();
                Naming.rebind("rmi://127.0.0.1:1099/test", impl);
                System.out.println("Remote Object HelloImpl is binding");
        }
}
```

실습(서버 측 준비)

4. 3개의 java 파일을 컴파일

5. 자바가 제공하는 rmic를 이용하여 Stub 클래스 객체를 생성

rmic HelloImpl

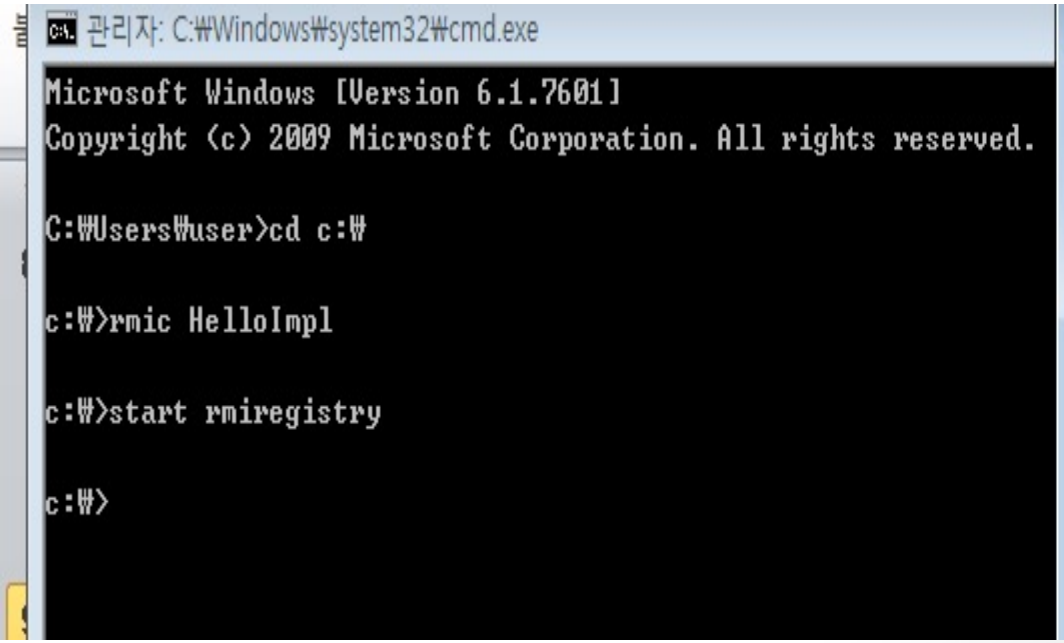


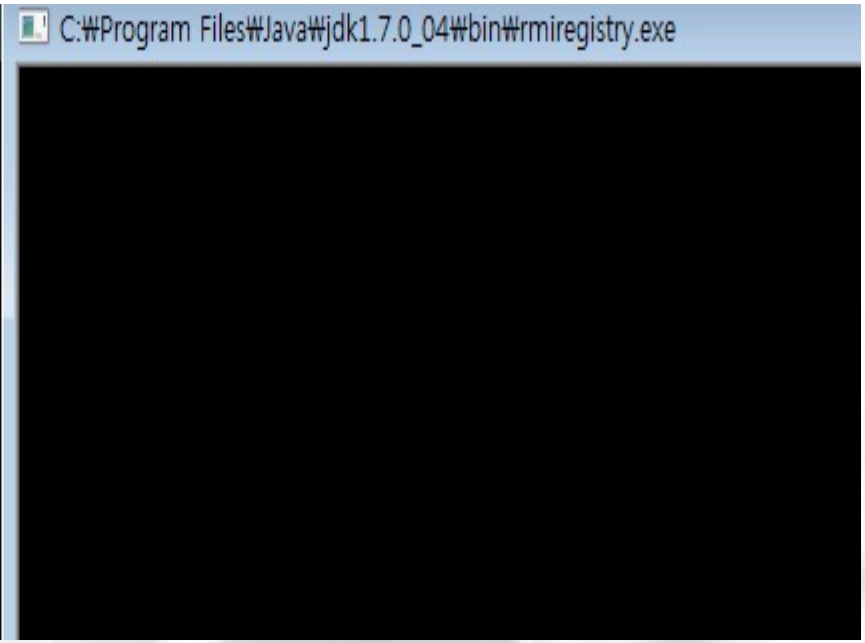
실습(서버 측 준비)

6. rmiregistry 를 실행

start rmiregistry

✓ Mac 에서는 rmiregistry

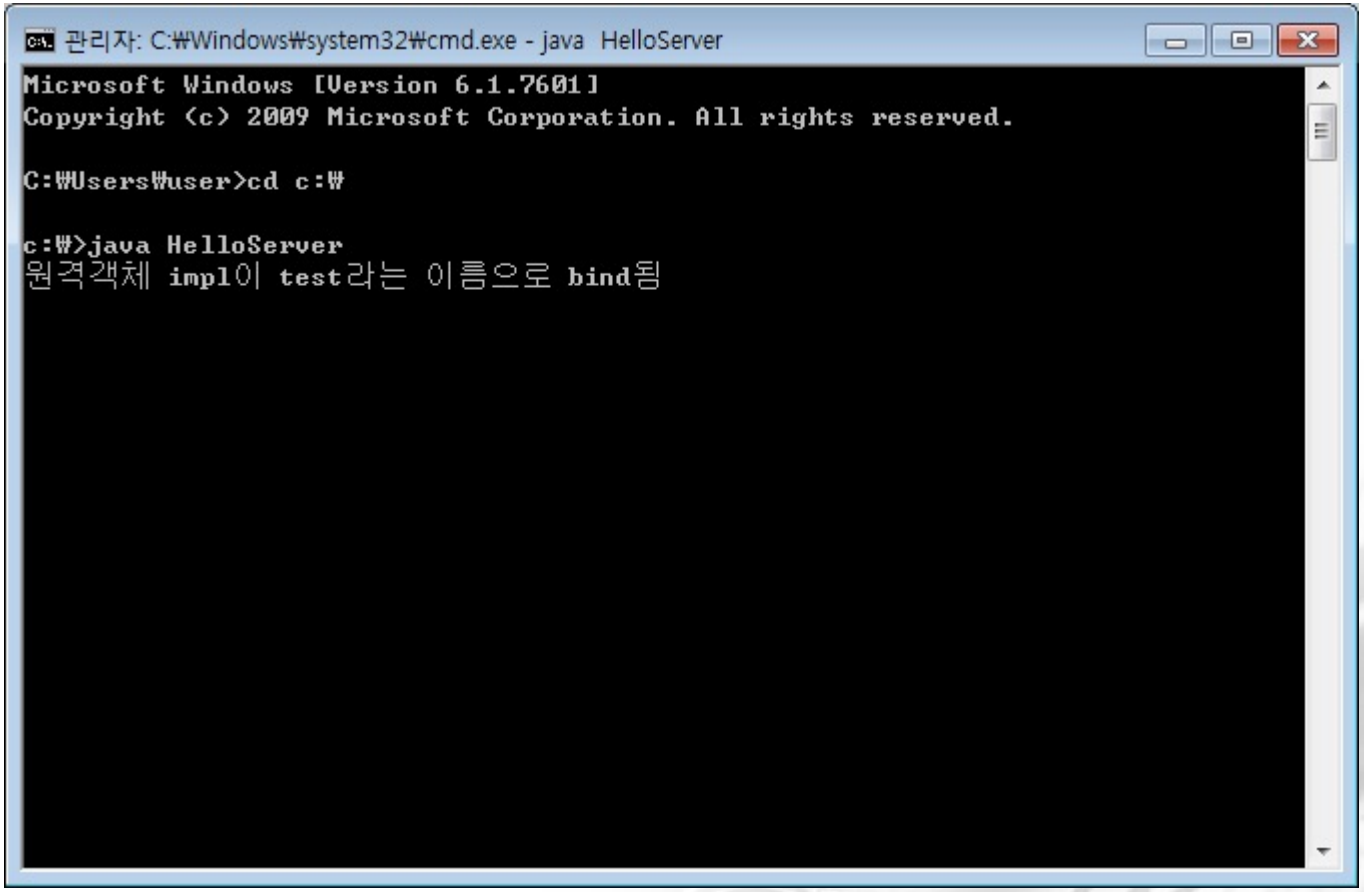




실습(서버 측 준비)

7.서버 측 클래스 파일 실행

java HelloServer



실습(클라이언트 측 준비)

1.클라이언트 측 자바 파일 생성

```
import java.rmi.*;
public class HelloClient{
public static void main(String[] args){
        try{
          Hello h=(Hello)Naming.lookup("rmi://127.0.0.1:1099/test");
          String str=h.sayHello("Jessica");
          System.out.println();
          System.out.println("result : " + str);
        }catch(Exception ex){ System.out.println(ex); }
        }
}
```

2.스터브 클래스 파일과 인터페이스 클래스 파일(Hello.class)을 복사

3.클라이언트 파일 컴파일 후 실행