네트워크

여러대의 컴퓨터를 통신 회선으로 연결한 것

서버

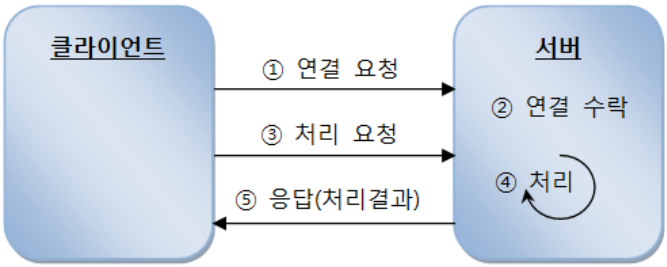
서비스를 제공하는 프로그램

예) 웹서버, FTP서버, 메신저 서버

클라이언트

서비스를 받는 프로그램

예) 웹 브라우저, FTP 클라이언트, 메신저



IP주소와 포트

IP(Internet Protocol) 주소

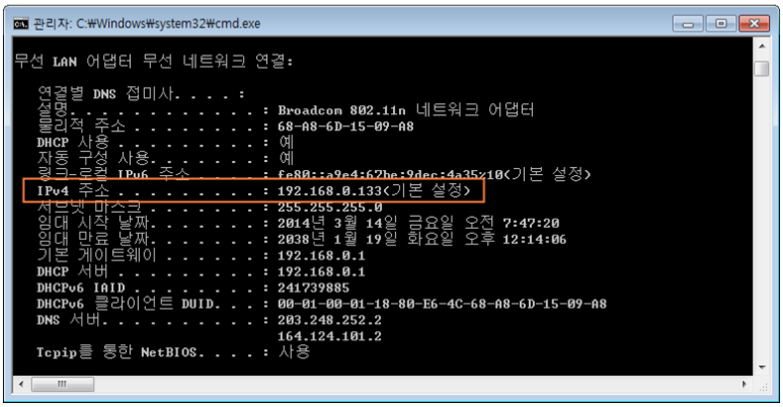
네트워크상에서 컴퓨터를 식별하는 번호

네트워크 어댑터(랜 (Lan) 카드) 마다 할당

IP 주소 확인 법 – 명령 프롬프트 (cmd.exe) 사용

xxx.xxx.xxx.xxx 형식으로 표현 (xxx는 0~255 사이의 정수)

```
C:₩>ipconfig /all
```



포트

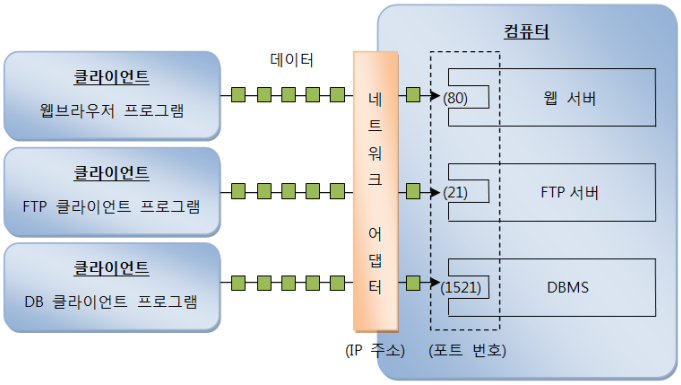
같은 컴퓨터 내에서 프로그램을 식별하는 번호

클라이언트는 서버 연결 요청 시 IP주소와 Port 번호를 같이 제공

0 ~ 65535 범위를 가짐

포트의 범위

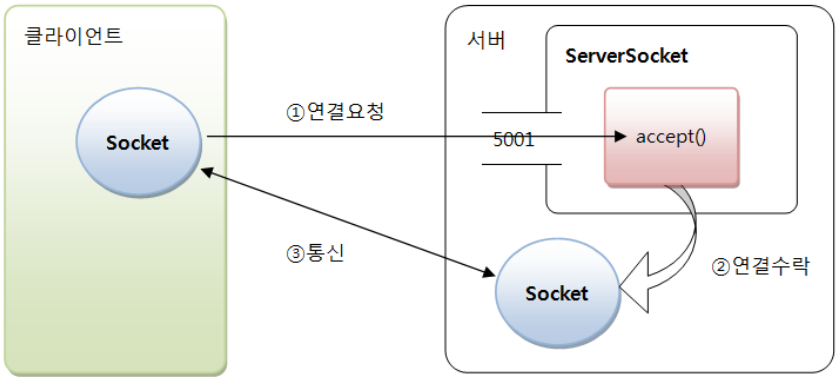
구분명	범위	설명
Well Know Port Numbers	0~1023	국제인터넷주소관리기구(ICANN)가 특정 애플리케이션용으로 미리 예약한 포트
Registered Port Numbers	1024~49151	회사에서 등록해서 사용할 수 있는 포트
Dynamic Or Private Port Numbers	49152~65535	운영체제가 부여하는 동적 포트 또는 개인적인 목적으로 사용할 수 있는 포트



TCP

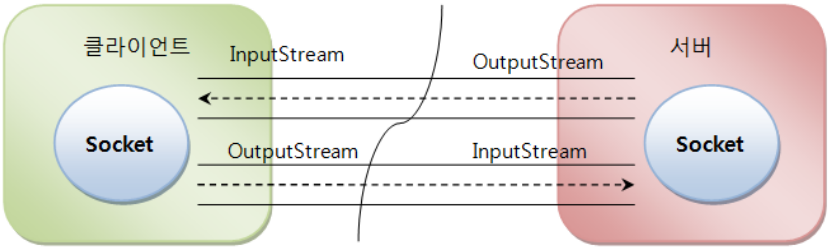
java.net API

ServerSocket, Socket 의 용도



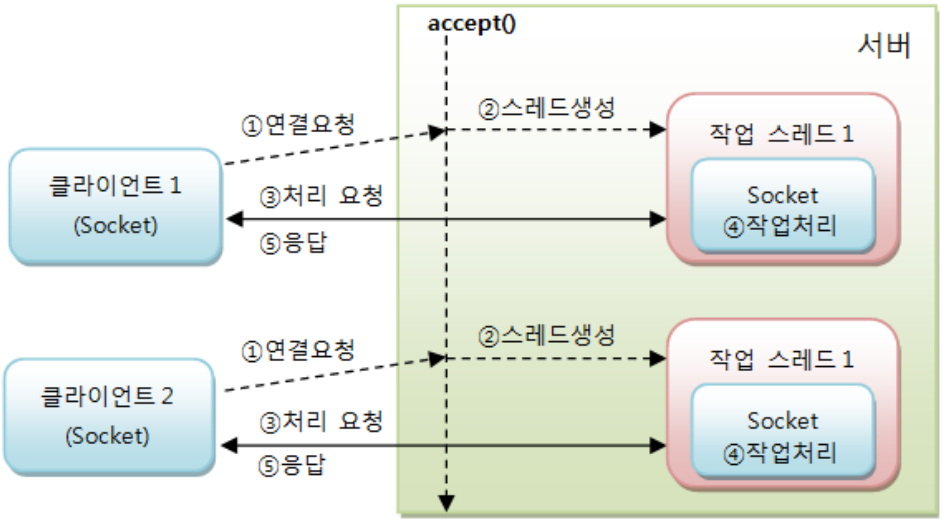
소켓 데이터 통신

Socket 객체로부터 입출력 스트림 얻기



쓰레드 병렬 처리

accept(), connect(), read(), write()는 별도의 작업 쓰레드에서 생성



클라이언트 서버 접속 프로그램 (실습해 보세요)

```
< Server.java >
import java.io.IOException;
import java.io.InputStream;
import java.net.InetSocketAddress;
import java.net.ServerSocket;
import java.net.Socket;

public class TcpServer1 {
	public static void main(String[] args) throws IOException {
		ServerSocket ss = new ServerSocket();
		ss.bind(new InetSocketAddress(40005)); // listener를 띄우는 것
		Socket socket = ss.accept(); // 클라이언트 접속 요청이 있을때 까지 멈춰있다

		System.out.println("클라이언트 접속: " + socket.getRemoteSocketAddress());

		byte[] buf = new byte[1024];
		InputStream in = socket.getInputStream();
		int byteReadCount = in.read(buf);
```

```
                String msg = new String(buf, 0, byteReadCount, "UTF-8"); // UTF-8 Decoding 방식
                System.out.println("전송 메시지: " + msg);

                socket.close();
                ss.close();
        }
}
```

1대1 통신 방식 (클라이언트 1개, 서버 1개)

```
< Client.java >
import java.io.IOException;
import java.io.OutputStream;
import java.net.Socket;

public class TcpClient1 {
        public static void main(String[] args) throws IOException {
                Socket socket = new Socket("127.0.0.1", 40005);

                String msg = "안녕하세요?";
                byte[] bytes = msg.getBytes("UTF-8");
                OutputStream out = socket.getOutputStream();
                out.write(bytes);
                out.flush();

                socket.close();
        }
}
```

```
클라이언트 접속: /127.0.0.1:53948
전송 메시지: 안녕하세요?
```

1대 다 통신 방식 (클라이언트 여러개, 서버 1개)

```
< Server.java >
import java.io.IOException;
import java.net.InetSocketAddress;
import java.net.ServerSocket;
import java.net.Socket;

public class TcpServer1 {
        public static void main(String[] args) throws IOException {
                ServerSocket ss = new ServerSocket();
                ss.bind(new InetSocketAddress(40005)); // listener를 띄우는 것

                System.out.println("서버가 시작되었습니다");

                try {
                        while(true) {
                                Socket socket = ss.accept(); // 클라이언트 접속 요청이 있을때 까지 멈춰있다
                                System.out.println("클라이언트 접속: " + socket.getRemoteSocketAddress());
```

```
                    ClientService service = new ClientService(socket);
                    service.setDaemon(true);   // 메인 쓰레드가 종료되면 Client 쓰레드도 모두 종료 시킴
                    service.start();
                }
            } catch(Exception e) {}
                    ss.close();
            }
        }
}
```

```
< Client.java >
import java.io.IOException;
import java.io.InputStream;
import java.io.OutputStream;
import java.net.Socket;
import java.util.Scanner;

public class TcpClient1 {
    public static void main(String[] args) throws IOException {
            Socket socket = new Socket("127.0.0.1", 40005);

            try {
                    while(true) {
                            Scanner scanner = new Scanner(System.in);
                            if(scanner.hasNext()) {
                                    String msg = scanner.nextLine();
                                    scanner.close();

                                    byte[] bytes = msg.getBytes("UTF-8");
                                    OutputStream out = socket.getOutputStream();
                                    out.write(bytes);
                                    out.flush();
                            }
                    }
            } catch(Exception e) {
            }
            socket.close();
    }
}

class ClientService extends Thread {
            private Socket socket;

            ClientService(Socket socket) {
                    this.socket = socket;
            }

            public void run() {
                try {
                    while(true) {
                            byte[] buf = new byte[1024];
                            InputStream in = socket.getInputStream();
                            int byteReadCount = in.read(buf);

                            if(byteReadCount > 0) {
                                    String msg = new String(buf, 0, byteReadCount, "UTF-8"); // UTF-8
Decoding 방식

                                    System.out.println("전송 메시지: " + msg);
                            }
                    }
                } catch(IOException e) {
```

```
            }
        }
}
```

1대 1 통신

```
1. EchoServer.java
public class EchoServer {

 private void startServer(){
  try{
   // 서버소켓 생성
   ServerSocket ss = new ServerSocket(2008);
   System.out.println("서버 구동 완료.");

   // 클라이언트 접속 대기
   Socket s = ss.accept();  // 소켓객체를 반환해 준다
   System.out.println("클라이언트 접속 완료.");

   // 소켓에서 입출력 스트림 생성
   InputStream in = s.getInputStream();
   OutputStream out = s.getOutputStream();

   // 입력 스트림에서 읽은 데이터를 출력 스트림으로 전송
   int readData = 0;
   while((readData = in.read()) != -1) {
    System.out.write(readData);  // 디버깅용
    out.write(readData);
   }

  }catch(Exception e){
   e.printStackTrace();
  }
 }

 public static void main(String[] args) {
  new EchoServer().startServer();
 }
}
```

```
2. EchoClient.java
public class EchoClient {

 private void startClient(){
  try{
   // 소켓 생성(서버 접속 요청)
   Socket s = new Socket("localhost", 2008);
   System.out.println("서버 접속 완료.");

   // 키보드에서 입력한 문자를 꺼내기 위한 입력 스트림 생성
   BufferedReader key = new BufferedReader(new InputStreamReader(System.in)); // 문자열로 바꾸기 위해

   // 서버로 전송할 출력 스트림 생성
```

```
   PrintWriter out = new PrintWriter(s.getOutputStream(), true); // 한바이트가 아닌 수십바이트를 한번에
읽어와서 보내기 위해 2 차 스트림 사용

   // 서버에서 보내는 데이터를 받기 위한 입력 스트림 생성
   BufferedReader in = new BufferedReader(new InputStreamReader(s.getInputStream()));
   // 키보드에서 입력한 데이터를 출력 스트림으로 전송
   String readData = "";
   while((readData = key.readLine()) != null) {
    out.println(readData);
    //out.flush();

    // 서버에서 받은 데이터를 화면에 출력
    String recvData = "";
    recvData = in.readLine();
    System.out.println("에코 메세지: " + recvData);
   }

  }catch(Exception e){
   e.printStackTrace();
  }
 }

 public static void main(String[] args) {
  new EchoClient().startClient();
 }
}
```

서버1개, 클라이언트 여러 개

```
1. EchoCleint.java
public class EchoClient {

 private void startClient(){
  try{
   // 소켓 생성(서버 접속 요청)
   Socket s = new Socket("localhost", 2008);
   System.out.println("서버 접속 완료.");

   // 키보드에서 입력한 문자를 꺼내기 위한 입력 스트림 생성
   BufferedReader key = new BufferedReader(new InputStreamReader(System.in)); // 문자열로 바꾸기 위해

   // 서버로 전송할 출력 스트림 생성
   PrintWriter out = new PrintWriter(s.getOutputStream(), true); // 한바이트가 아닌 수십바이트를 한번에
읽어와서 보내기 위해 2 차 스트림 사용

   // 서버에서 보내는 데이터를 받기 위한 입력 스트림 생성
   BufferedReader in = new BufferedReader(new InputStreamReader(s.getInputStream()));
   // 키보드에서 입력한 데이터를 출력 스트림으로 전송
   String readData = "";
   while((readData = key.readLine()) != null) {
    out.println(readData);
    //out.flush();
```

```
    // 서버에서 받은 데이터를 화면에 출력
    String recvData = "";
    recvData = in.readLine();
    System.out.println("에코 메세지: " + recvData);
   }
  }catch(Exception e){
   e.printStackTrace();
  }
 }

 public static void main(String[] args) {
  new EchoClient().startClient();
 }
}
```

```
2. EchoServer.java
public class EchoServer {

 private void startServer(){
  try{
   // 서버소켓 생성
   ServerSocket ss = new ServerSocket(2008);
   System.out.println("서버 구동 완료.");

   while(true) {
    // 클라이언트 접속 대기
    Socket s = ss.accept();  // 소켓객체를 반환해 준다
    System.out.println("클라이언트 접속 완료.");
    EchoServerThread et = new EchoServerThread(s);
    et.start();
   }
  }catch(Exception e){
   e.printStackTrace();
  }
 }

 public static void main(String[] args) {
  new EchoServer().startServer();
 }
}
3. EchoServerThread.java
public class EchoServerThread extends Thread{
 private Socket s;

 public EchoServerThread(Socket s) {
  this.s = s;
 }

 public void run(){
  try{
   // 소켓에서 입출력 스트림 생성
   InputStream in = s.getInputStream();
   OutputStream out = s.getOutputStream();
   // 입력 스트림에서 읽은 데이터를 출력 스트림으로 전송
   int readData = 0;
   while((readData = in.read()) != -1) {
```

```
    System.out.write(readData);   // 디버깅용
    out.write(readData);
   }
  }catch(Exception e){
   e.printStackTrace();
  }
 }
}
```

클라이언트간 통신

```
1. ChatServer.java
public class ChatServer {

 private Set<ChatServerThread> clients
      = new HashSet<>();

 private void startServer(){
  try{
   // 서버소켓 생성
   ServerSocket ss = new ServerSocket(2008);
   System.out.println("서버 구동 완료.");

   while(true){
    // 클라이언트 접속 대기
    Socket s = ss.accept();
    System.out.println("클라이언트 접속 완료.");
    ChatServerThread et = new ChatServerThread(s, clients);
    et.start();
    clients.add(et);
   }
  }catch(Exception e){
   e.printStackTrace();
  }
 }

 public static void main(String[] args) {
  new ChatServer().startServer();
 }
}
```

```
2. ChatClient.java
public class ChatClient {

 private void startClient(){
  try{
   // 소켓 생성(서버 접속 요청)
   Socket s = new Socket("localhost", 2008);
   System.out.println("서버 접속 완료.");
   // 키보드에서 입력한 문자를 꺼내기 위한 입력 스트림 생성
   BufferedReader key = new BufferedReader(
        new InputStreamReader(
         System.in));
   // 서버로 전송할 출력 스트림 생성
   PrintWriter out = new PrintWriter(s.getOutputStream(), true); // auto flush
```

```
  // 서버에서 보내는 데이터를 받기 위한 입력 스트림 생성
  final BufferedReader in = new BufferedReader(
        new InputStreamReader(
         s.getInputStream()));


  new Thread(){
   public void run(){
    try{
     // 서버에서 받은 데이터를 화면에 출력
     String recvData = "";
     while((recvData = in.readLine()) != null){
      System.out.println(recvData);
     }
    }catch(Exception e){
     e.printStackTrace();
    }
   }
  }.start();

  // 키보드에서 입력한 데이터를 출력 스트림으로 전송
  String readData = "";
  while((readData = key.readLine()) != null){
   out.println(readData);
  }
 }catch(Exception e){
  e.printStackTrace();
 }
}


 public static void main(String[] args) {
  new ChatClient().startClient();
 }
}
```

```
3. ChatServerThread.java
public class ChatServerThread extends Thread{
 private Socket s;
 private BufferedReader in;
 private PrintWriter out;
 private String nickname; //  대화명
 private Set<ChatServerThread> clients;

 public ChatServerThread(Socket s, Set<ChatServerThread> clients) {
  this.s = s;
  this.clients = clients;
 }

 public void run(){
  try{
   // 소켓에서 입출력 스트림 생성
   in = new BufferedReader(
     new InputStreamReader(
      s.getInputStream()));
   out = new PrintWriter(s.getOutputStream(), true); // auto flush
   // 입력 스트림에서 읽은 데이터를 출력 스트림으로 전송
   String readData = "";
   while((readData = in.readLine()) != null){
    broadcastMsg(readData);
```

```
   }
  }catch(Exception e){
   e.printStackTrace();
  }
 }

 /**
  * 지정한 대화명을 저장한다.
  * @param nickname 대화명
  */
 private void login(String nickname) {

 }

 /**
  * 로그아웃
  */
 private void logout(){

  try{ if(out != null) out.close(); }catch(Exception e){}
  try{ if(in != null) in.close(); }catch(Exception e){}
  try{ if(s != null) s.close(); }catch(Exception e){}
 }

 /**
  * 자신의 담당 클라이언트에 지정한 메세지를 전송한다.
  * @param msg 전송메세지
  */
 private void sendMsg(String msg){
  out.println(msg);
 }

 /**
  * 접속되어 있는 모든 클라이언트에 지정한 메세지를 전송한다.
  * @param msg 전송메세지
  */
 private void broadcastMsg(String msg){
  for(ChatServerThread ct : clients){
   ct.sendMsg(msg);
  }

//  for(int i=0; i<clients.size(); i++){
//   ChatServerThread ct = clients.get(i);
//   ct.sendMsg(msg);
//  }
  }
 }
```

HTTP protocol :

웹브라우저와 웹서버가 통신하는것

< 요청정보(HTTP) >

3 가지 영역으로 나눈다

요청방식 URL HTTP 버젼 (시작라인)

GET / HTTP/1.1

헤더명: 헤더값 (헤더)

바디영역 (바디)

* 크롬에서 F12 를 누르면 개발자 모드로 전환되고 Reflush 버튼 누르고 Headers 정보 보면 Request/Response 정보 보인다.

HTTP 프로토콜 (실습하기)

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

public class HttpClient {
  public static void main(String[] args) {
    try {
      Socket s = new Socket("www.naver.com", 80); // 웹서버에 접속
      BufferedReader key = new BufferedReader(
         new InputStreamReader(
            System.in));
      // 서버로 전송할 출력 스트림 생성
      PrintWriter out = new PrintWriter(s.getOutputStream(), true); // auto flush
      // 서버에서 보내는 데이터를 받기 위한 입력 스트림 생성
      final BufferedReader in = new BufferedReader(
                    new InputStreamReader(
                  s.getInputStream(), "UTF-8")); // 인코딩 방식
      out.println("GET /index.html HTTP/1.1");
      out.println("Host: www.naver.com");
      out.println();

      String readData = "";
      while((readData = in.readLine()) != null) {
        System.out.println(readData);
      }
    } catch(Exception e) {
      e.printStackTrace();

    }
  }
}
```

```
HTTP/1.1 200 OK
Server: nginx
Date: Sun, 09 Oct 2016 06:24:09 GMT
Content-Type: text/html; charset=UTF-8
Transfer-Encoding: chunked
Connection: close
Cache-Control: no-cache, no-store, must-revalidate
Pragma: no-cache
P3P: CP="CAO DSP CURa
……
```

수고했습니다. 이제 GUI를 만들어서 채팅 프로그램을 만들어 보세요. 시간은 2~3주 정도 소요됩니다.