■ 소켓 프로그래밍

소켓 프로그래밍이란 소켓을 이용한 통신 프로그래밍을 말한다.

소켓이란 (Socket) 프로세스 통신 간에 사용되는 양쪽 끝단을 의미한다. 즉, 소프트웨어 차원에서 원격에 존재하는 두 호스트를 연결해주는 매개체이다.

대표적으로 TCP 와 UDP이용이 있다. 소켓프로그램은 주로 서버-클라이언트 2개의 프로그램이 쌍을 이룬다. 이런 흐름은 포트에 대기하다가 클라이언트의 연결을 기다리고 (listen), 클라이언트가 접근을 요청하면 받아들여서(accept) 서버-클라이언트 연결을 설정하고, 클라이언트의 여러 명령을 받아서 서비스를 하게 되는 것이다.

비유적으로 생각해보자.

서버소켓은 먼저 소켓을 생성한다. (전화기 구입)

그리고 주소와 포트를 할당한다. (전화번호 개통)

요청대기 (전화를 기다림)

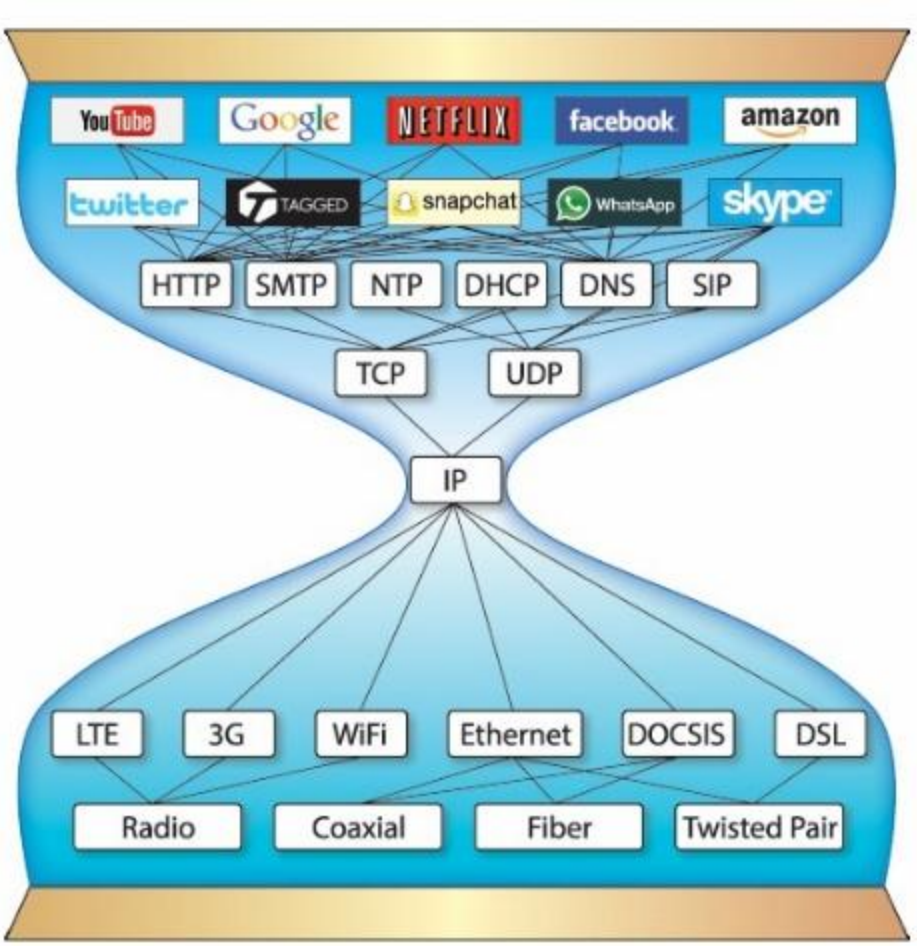
연결수락 (전화를 받는다)

클라이언트 소켓은 똑같이 소켓을 생성한다 (전화기 구입)

연결을 요청한다 (전화걸기)

■ TCP/IP

TCP/IP는 인터넷 프로토콜 표준이 제정될 때 부터 지금처럼 두 개로 분리되어 제안되었다. 인터넷의 설계 기준과 관계를 보면 이렇게 IP가 허리인 형태, 즉 모래시계 형태가 된다. 이러한 디자인으로 어떤 통신 관련 하드웨어 기술이라도 IP만 있으면 되고 어떤 소프트웨어라도 IP에서 동작하게 할 수 있으면 된다라는 의미이다.



이 때문에 하드웨어와 소프트웨어의 분리 발전이 가능해졌다. 예를들어 스마트폰 기기를 바꿨다고 해서 예전에 쓰던 어플을 사용못하지 않는것, 새 어플을 깔기 위해서 하드웨어를 교체하지 않는 것 이런것이다. (물론 매우 예전기종이어서 더 이상 운영체제의 업데이트가 되지 않는 벽돌수준의 기계 이런것들은 제외다.)

어찌됐건 ip 만 지원하면 뭐든지 된다라는 사실에서 매우 빠르게 기술이 발전해왔다.

Internet은 inter-(상호의) net(네트워크)가 합쳐진 말이다. 독립적인 net을 interaction할 수 있게 만드는 것이 인터넷이다. 그리고 이렇게 서로 독립적인 net들을 연결하기 위한 프로토콜이 Internet Protocol 즉 , IP이다.

IP는 서로 다른 네트워크를 연결하는데 각 네트워크의 관리자들이 있을 것이다. 관리자가 막아버린다면 통신을 하지 못할 것이다. 이렇게 프로토콜이 통신의 안정성을 보장하지 못한다면 안된다. 그래서 안정적으로 통신을 할 수 있게 layer를 까는 것이 TCP이다.

■ TCP

Transmissoin Control Protocol 의 약자로 전송 제어 프로토콜이라고 부른다. 안정적으로 패킷이 가지 못하였을 때 이것을 바로잡아서 보내는 역할을 한다. 그것을 하는 것이 ACK이다. ACK는 Acknowledge의 준말로 인식이라는 뜻을 가지고 있다. TCP는 패킷을 받은 쪽이 보낸 쪽으로 별도의 ACK 패킷을 만들어서 답장을 보낸다. 즉, 내가 보낸것이 저쪽에 잘 도착했는지를 보낸 쪽이 알

수 있게 하는것이다. 메세지가 오지 않는다면? 제대로 가지 못한 것이다. ACK가 일정동안 오지 않으면 다시 보낸다.

잘 생각해 보면 2가지 경우다. 진짜 못갔거나, 갔는데 답장이 오는 도중 날라갔거나. 하지만 이 둘을 구분하기에는 까다롭기 때문에 그냥 보낸 쪽에서 패킷을 한 번 더 보낸다. 이렇게 TCP는 이것 저것 할일이 많다. 그래서 느린 특성을 가지고 있지만 신뢰성은 좋겠다.

전화라고 생각하면 편하다. 데이터를 전송하기 전에 먼저 상대편과의 연결을 확인한 후 데이터를 전송하고 전송되었는지 확인한다.

연결기반(connection-oriented) 통신, 1:1 통신

UDP보다 전송속도가 느림

사용되는 클래스 : Socket, ServerSocket

■ UDP

User Datagram Protocol의 약자로 단방향 통신을 의미한다. 데이터의 유실이 생길수 있어 신뢰성을 보장하지 못한다. 실시간 동영상 서비스에 많이 이용된다.

편지, 소포라고 생각하자. 연결하지 않고 데이터를 전송하고 전송 되었는지 확인하지 않는다. 데이터를 보낸 순서대로 가는것도 아니다.

신뢰성이 없는 데이터를 전송

TCP보다 전송속도가 빠름

사용되는 클래스 : DatagramSocket, DatagramPacket

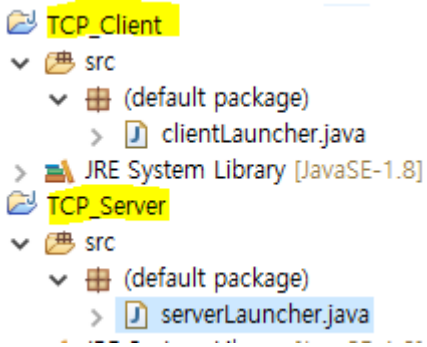
둘의 공통점? 포트번호를 이용하여 주소를 지정한다.

■ TCP와 UDP 통신 실습

이클립스 실습)

TCP 실습

이런식으로 간단하게 프로젝트를 2개 만든다. 먼저 TCP부터 실습을 하고 각 프로젝트에 파일 하나씩을 생성한다.



serverLauncher.java

```cs
public class serverLauncher {

    public static void main(String[] args) {
        Socket client = null;
        ServerSocket server = null;

        int port = 10789;


        try {
            server = new ServerSocket(port);
            client = server.accept(); // call set up

            if(client != null) {
                System.out.println("client : "+client.toString());
            }

            client.close();
            server.close();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }

}
```

Colored by Color Scripter

통신을 하기 위해서는 서버에서 소켓을 준비해야 한다. 서버 쪽에서는 반드시 두 개의 소켓이 필요하다. 하나는 서버용 소켓이고 하나는 클라이언트의 연결을 기다리는 소켓이다. 이 소켓은 ServerSockect으로 만들 수 있다.

Socket 클래스

Socket클래스는 java.net패키지에 있는 클래스로 서버와 통신하기 위해 클라이언트에서 사용하는 소켓이다.

ServerSocket 클래스

ServerSocket클래스는 java.net패키지에 있는 클래스로 클라이언트 측으로부터 연결 요청을 기다리는 모니터링을 위해 사용되며 서버가 클라이언트의 요청을 수락하면 Socket객체가 새로 생성된다. 얘는 클라이언트와의 데이터 송수신에 사용되지 않는다. 이것은 TCP Server Socket을 의미하는것과 같다. ServerSocket Object는 클라이언트쪽에서 TCP 요청이 있을 때 까지 Blocking된다.

서버소켓은 클라이언트의 연결요청이 올 때 까지 실행을 멈추고 계속 기다린다. 그 이후 소켓과 소켓서버를 다 닫아준다.

Socket에서는 두 가지 예외가 터질 수 있다.

1. HOST를 찾을 수 없을 때, 즉 Server의 Port가 열려 있지 않은 경우 UnknownHostException이 발생한다.

2. Network의 실패, 방화벽 때문에 Server에 접근할 수 없는 경우 IOException이 발생한다.

clientLauncher.java

```
1  public class clientLauncher {
2
3      public static void main(String[] args) {
4          Socket server = null;
5          String serIp = "127.0.0.1"; // 아이피
6          int port = 10789; // 포트번호
7
8          try {
9              server = new Socket(serIp, port);
10
11             if (server != null) {
12                 System.out.println("client : " + server.toString());
13             }
14
```

```
15                server.close();
16
17            } catch (UnknownHostException e) {
18                e.printStackTrace();
19            } catch (IOException e) {
20                e.printStackTrace();
21            }
22
23        }
24    }
```

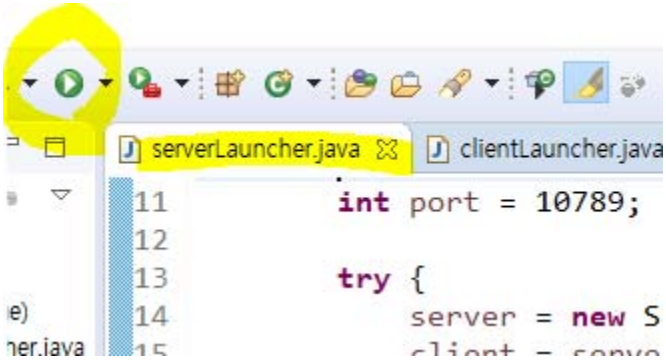
Colored by Color Scripter cs

소켓을 만들고 인자로 아이피와 포트번호를 넣었다.

Socket(InetAddress address, int port) throws IOException

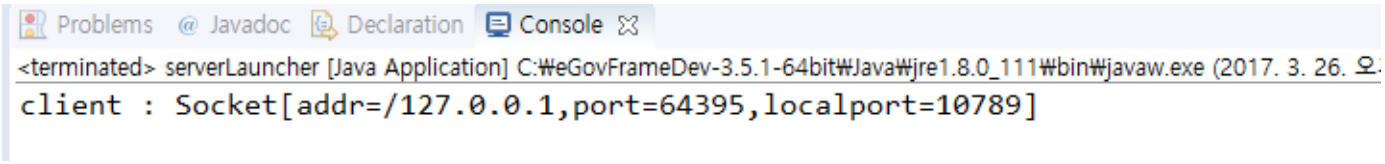
소켓의 스트림을 형성하고 인자로 넣은 아이피의 포트번호에 연결한다. 서버가 뚫려있다면 서버의 내용을 출력한다.

그럼이제 serverLauncher를 실행시켜보자. 실행시키면 console에는 아무것도 찍히지 않는다. 왜냐하면 클라이언트의 요청이 올 때까지 기다리고 있기 때문이다.



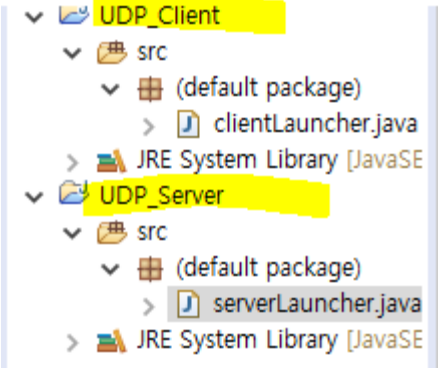
이제 clientLauncher를 실행시키자. 그러면 console창에 서버정보가 뜰 것이다.

Socket에서의 toString은 소켓을 스트링으로 변환시키는 것이다.



UDP 실습

TCP할때와 같이 두 프로젝트를 만들고 파일을 생성한다.



UDP는 비연결 지향이고 Data전송할 때 잘 도착했는지 알 수가 없다.

serverLauncher.java

```
1  public class serverLauncher {
2
3      public static void main(String[] args) {
4          // udp 는 accept없이 패킷을 바로 보낼 수 있다.
5          DatagramSocket socket;
6          DatagramPacket packet;
7          int port = 10789;
8          byte[] buf = new byte[1024];
9
10         try {
11             socket = new DatagramSocket(port);
12             packet = new DatagramPacket(buf, buf.length);
13             socket.receive(packet);
14
15             String msg = new String(packet.getData());
16             System.out.println("received Msg: "+msg);
17
```

```
18              socket.close();
19          } catch (SocketException e) {
20              e.printStackTrace();
21          } catch (IOException e) {
22              e.printStackTrace();
23          }
24
25      }
26
27  }
```

Colored by Color Scripter cs

DatagramSocket

DatagramSocket 클래스는 데이터 그램을 송수신 하기 위한 소켓이다.

```
public DatagraSocket(int port) throws SocketException
```

데이터 그램 소켓을 구축해, 로컬 호스트 머신상의 지정된 포트에 바인드한다.

예외터지는 상황은 SocektException인데 이것은 소켓을 열 수 가없는 경우 혹은 포트지정을 할 수 없는 경우에 터진다.

DatagramPacket

UDP Datagram은 Data를 송신하기 위한 기능과 수신을 하기 위한 기능이 있다. TCP의 경우 스트림을 이용해 데이터를 주고 받지만 UDP의 경우 데이터그램을 이용해 데이터를 송수신한다

clientLauncher.java

```
1   public class clientLauncher {
2
3       public static void main(String[] args) {
4           DatagramSocket socket;
5           DatagramPacket packet;
6           String text = "hello socketProgramming";
```

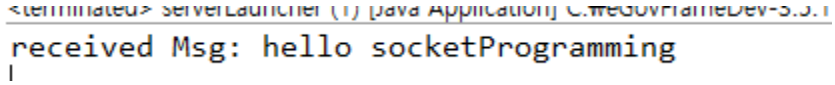
cs

```
7           byte[] buf = text.getBytes();
8           String serverIp = "127.0.0.1";
9           int port = 10789;
10
11          try {
12              socket = new DatagramSocket();
13              packet = new DatagramPacket(buf, buf.length,
14                      InetAddress.getByName(serverIp), port);
15
16              socket.send(packet);
17
18              System.out.println("send Msg : "+text);
19
20              socket.close();
21          } catch (SocketException e) {
22              e.printStackTrace();
23          } catch (UnknownHostException e) {
24              e.printStackTrace();
25          } catch (IOException e) {
26              e.printStackTrace();
27          }
28
29      }
30  }
```

Colored by Color Scripter

String을 getByte로 넣었다.getByte는 string을 일련의 바이트들로 나타내는 것이다.

그리고 소켓에다가 패킷을 설정한 것을 넣었다. 파리미터로 buf는 패킷 데이터이고 buf.length는 패킷의 길이이다. buf의 길이와 같거나 작아야 한다. 그리고 ip주소와 포트번호가 들어간다. 그리고 socket에 send로 날린다.



그럼 서버측에서 메세지를 받아서 보여주게 된다.