[디지털 컨버전스]
스마트 콘텐츠와 웹 융합 응용SW 개발자 양성과정

강사 : 이상훈
학생 : 임초롱

23회차

네트워크 - Server

링크 https://github.com/limcholong/LectureContents/blob/main/java/CholongLim/Day23/src/SocketServerTest.java

```java
/*
 서버, 클라이언트 개념은 무엇일까 ?
 서비스는 뭐지?
 이거 내가 하기 너무 힘드니까 돈 줄테니깐 좀 해줘

 이 개념 하에서 서버란 것은 무엇인가 ?
  서비스 제공자
  클라이언트는 ?
  서비스 이용자
 */

import java.io.*;
import java.net.ServerSocket;
import java.net.Socket;
import java.util.Date;

public class SocketServerTest {
    // 누군가 접속하면 접속 시간을 알려주는 서비스
    public static void main(String[] args) {
        // 포트의 역할 : 서비스 번호
        // 결국 우리가 어떤 서비스에 접근하기 위해서는 무엇을 알아야 한다 ?
        // 아이피(192.168...)와 포트(80 / 443 ...)
        int port = Integer.parseInt( s: "33333");

        try{
            // 소켓이란 ? 돼지코 구멍(콘센트)
            // 전기 분야에서 소켓에 전원 코드를 연결하면 거기 제품들이 구동 가능한것과 마찬가지로
            // 프로그래밍 분야에서 소켓이란 다른 컴퓨터와 내 컴퓨터를 연결하는 역할을 한다.
            // 소켓이 없으면 내 컴퓨터와 다른 컴퓨터의 통신 안된다.
            // 그러니까 통신을 수행할 수 있도록 내 소켓을 만들었음

            ServerSocket servSock = new ServerSocket(port);
            System.out.println("Server : Listening - " + port);
```

Server : 서비스 제공자
Clinet : 서비스 이용자

IP : 컴퓨터들이 갖고 있는 고유번호,
사설IP의 경우 192.168.…으로 시작하는 것을 IP라고 한다.

포트(port) :
서비스 번호로 우리가 서비스에 접근하기 위해 알아야 할 번호이다.

Socket :
- 콘센트의 돼지코 구멍과 같은 것이다.
- 전기 분야에서 소켓에 전원 코드를 연결하면 제품들이 구동 가능한 것과 마찬가지로 프로그래밍 분야에서 소켓이란 다른 컴퓨터와 내 컴퓨터를 연결하는 역할을 한다.
- 소켓이 없으면 내 컴퓨터와 다른 컴퓨터의 통신이 안된다.
- 통신을 수행할 수 있도록 소켓을 만들어야 한다.

```
ServerSocket servSock = new ServerSocket(port)
```

네트워크 - Server

링크 https://github.com/limcholong/LectureContents/blob/main/java/CholongLim/Day23/src/SocketServerTest.java

```
35          while(true){
36              // accept() 부분에서 서버는 Blocking(블록킹) 연산을 수행하고 있음
37              // 블록킹 : 막는 것 = 무엇을 막는것인가?
38              // accept = Thread spinlock같은 느낌이다.
39              // 나올때 까지 화장실 문을 두드림
40              // 니(클라이언트)가 준비될때까지 난 계속 기다린다. (문 두드리면서)
41              // 클라이언트가 소켓을 요청할때까지 계속 문 두들긴다.
42
43              // Blocking의 반대 개념도 있지 않을까 ?
44              // Non - Blocking 이라고 하며 비동기 처리와 관계가 깊음
45
46              // (여기에 있는 sock은 접속한 사용자 소켓임)
47              Socket sock = servSock.accept();
48              // 접속이 완료되었으면 접속한 클라이언트 IP를 확인한다.
49
50              System.out.println("[" + sock.getInetAddress() + "] client connected");
51
52              // 클라이언트를 향해 출력할 객체를 생성함
53              OutputStream out = sock.getOutputStream();
54              // PrintWriter에 송신용 객체를 배치함으로써
55              // writer.println 으로 구동시키는 것이 전송되게 만들었음
56              PrintWriter writer = new PrintWriter(out, autoFlush: true);
57
58              writer.println(new Date().toString());
59
60              // 입력이 들어올 때 까지 대기 (InputStream)
61              InputStream in = sock.getInputStream();
62              BufferedReader reader = new BufferedReader(new InputStreamReader(in));
63
64              System.out.println("msg : " + reader.readLine());
65          }
66      }catch (IOException e){
67          System.out.println("Server Exception : "+ e.getMessage());
68          e.printStackTrace();
```

Socket sock = servSock.accept( );

accept( ) 부분에서 서버는 Blocking(블록킹) 연산을 수행하고 있다.
Blocking이란 막는 것을 뜻하는 여기서 무엇을 막는 것인가?
accept = Thread의 Spinlock과 비슷하다.

Spinlock 화장실 예에서, 사용자가 있어도 나올 때까지 문을 두드리는 것과 같은 일을 수행한다고 했었다. 이처럼 accept도 나올 때까지 화장실 문을 두드린다. (클라이언트가 소켓을 요청할 때 까지 계속 문을 두들김)

클라이언트 접속이 완료되었으면 접속한 클라이언트 IP를 확인한다.

outputStream out = sock.getOutputStream( );

클라이언트를 향해 출력할 객체를 생성한다.

InputStream in = sock.getInputStream( );

입력이 들어올 때까지 대기한다.

네트워크 - Client

링크 https://github.com/limcholong/LectureContents/blob/main/java/CholongLim/Day23/src/SocketClientTest.java

```java
import ...

public class SocketClientTest {
    // 접속 시간에 대한 정보를 획득하고자 하는 서비스 이용자
    public static void main(String[] args) {
        String hostname = "192.168.123.101";
        int port = 33333;

        for(int i =0; i < 10; i++){
            try{
                // 클라이언트 자신의 소켓을 생성한다.
                // 생성할 때 나는 서버의 ip 주소(hostname)에 서비스(port)에 접속하고 싶어!
                // 라고 요청하면서 소켓을 만든다.
                Socket sock = new Socket(hostname, port);

                // 서버에게 전송하기 위한 객체를 준비함
                OutputStream out = sock.getOutputStream();
                // 이 내용을 서버에게 송신함
                String str = "Hello Network Programming";

                out.write(str.getBytes());

                // 서버에서 날아온 수신 정보
                // 서버가 아웃풋으로 보낸 시간 정보
                // 입력이 들어올 때 까지 대기 (InputStream)
                InputStream in = sock.getInputStream();
                BufferedReader reader = new BufferedReader(new InputStreamReader(in));

                String time = reader.readLine();
                System.out.println(time);
                // 주소가 잘못됐을때, 동작에러 시를 위한 코드
            } catch (UnexpectedException e){
                System.out.println("Server Not Found : " + e.getMessage());
            } catch (IOException e){
                System.out.println("I/O Error : " + e.getMessage());
            }
        }
    }
```