o 각 code block 의 구조 및 기능설명

< 도메인 접속 프로그램 >

- 서버, 클라이언트 띄움 - 클라이언트가 서버에 IP로 접속하거나 도메인명으로 접속할수 있음 - 접속되지 않을 경우 찾을수 없음 메세지 출력

클라이언트 -> 서버 IP/도메인이 일치 O : 접속 가능 메시지 IP/도메인 출력
클라이언트 -> 서버 IP/도메인이 일치 X : 찾을 수 없음 메시지 출력

도메인 입력 -> IP주소로 변환

IP주소 입력 -> 텍스트파일에 해당 DNS정보가 있으면 도메인 주소로 반환

현재 local DNS.txt파일에는 127.0.0.1 www.superman.com / 127.0.0.2 www.test.com 두개의 IP -> 도메인 변환 주소가 들어있다.

(1) LocalDNS_Server_김영민_20162820.java

<Code Block (1)>

```
public static void main(String[] args) {
        // TODO Auto-generated method stub
        ServerSocket listener = null;
        Socket socket = null;

        try {
                listener = new ServerSocket(9999);

                while(true) {
                        System.out.println("연결을 기다리고 있습니다....");
                        try {
                                socket = listener.accept();
                                System.out.println(socket.getInetAddress() + "와 연결되었습니다.");
                        }
                        catch(Exception e){
                                break;
                        }
                }
```

반복해서 클라이언트의 접속을 대기 하고 클라이언트와 접속이 성공했으면 연결되었습니다. 출력하는 과정

<Code Block (2)>

```
            }catch(IOException e) {
                        System.out.println(e.getMessage());
            }finally {
                        try{
                                    socket.close();
                                    listener.close();
                        }catch(IOException e) {
                                    System.out.println("클라이언트와 통신 중 오류가 발생했습니다.");
                        }

            }
}
```

연결이 실패하는 에러를 찾으면 클라이언트 통신중 오류발생 출력하는 과정

(2) Simple_FTP_Client_김영민_20162820

<Code Block 2-(1)>

```
public static String FindDomain(String ip) {
        try{
                File file = new File(".₩₩LocalDNS.txt");

                FileReader reader = new FileReader(file);

                BufferedReader bufReader = new BufferedReader(reader);

                String line = "";

                while((line = bufReader.readLine()) != null){
                        // 읽은 문자열을 스페이스 ' '를 기준으로 쪼갠다.
                        StringTokenizer tokenizer = new StringTokenizer(line, " ");
                        if( tokenizer == null || tokenizer.countTokens() <= 0 )
                                continue;

                        if( tokenizer.nextToken().equals(ip) == true )
                                return tokenizer.nextToken();
                }

                bufReader.close();
                reader.close();
        }catch (Exception e) {
        }

        return ip; // 찾을 수 없는 경우 받았던 ip를 그대로 반환 한다.
}
```

파일로 부터 ip 에 해당하는 도메인을 찾아 반환을 위해 public static String FindDomain(String ip)을 구성시켰다.

파일 객체를 생성하고, 입력 스트림생성 후 입력 버퍼를 생성시킨다. 그리고 while((line = bufReader.readLine()) != null) 을 통해 파일로 부터 문자열을 한 줄씩 읽어들인다.

if(tokenizer.nextToken().equals(ip) == true) 을 통해 샘플데이터 "1.2.3.4 www.domain.com" 을 입력해두고, 입력 받은 ip 가 1.2.3.4 라면,

www.domain.com 을 반환 하게 된다. txt 파일에 있는 아이피에 해당하는 도메인이 있다면, 반환 한다.

<Code Block 2-(2)>

```
try {
        scanner = new Scanner(System.in);

        InetAddress addr[] = null;
        System.out.print("원격지의 도메인이나 아이피 입력 : ");
        String remote = scanner.nextLine();

        boolean connected = false;
        try {
                String domain = "";

                addr = InetAddress.getAllByName(remote);
                if( addr[0].getHostAddress().equals(remote) == true ) {
                        domain = FindDomain(remote);
                }else {
                        domain = addr[0].getHostAddress();
                }

                System.out.println( "원격지 " + domain + "(으)로 접속 시도" );
```

String remote = scanner.nextLine(); 으로 콘솔창에서 domain 또는 IP 주소를 읽어 온다. 그리고 접속을 시도한다.

if(addr[0].getHostAddress().equals(remote) == true) : 만약 ip 라면, getHostAddress 값도 ip 일 것이므로 ip 인 경우에만 아래 domain 코드가 실행 된다. domain = FindDomain(remote); 파일로 부터 ip 에 해당하는 도메인을 찾아 반환 받는다.

<Code Block 2-(3)>

타임아웃을 걸기 위해 Socket 객체 생성과 동시에 접속시키지는 기능은 사용 하지 않는다. 기본 접속 타임아웃은 상대적으로 길어서 결과를 빨리빨리 확인하지 못한다. socket.setSoTimeout(1000); // 데이터 읽을 때 타임아웃 1 초로 설정해두고 socket.connect(address, 1000); 접속에 대한 타임아웃 1 초로 설정해둔다. 그리고 catch 를 통해 도메인은 존재 하지만 접속을 하지 못한 경우, 도메인에서 어드레스를 가져오지 못한 경우를 에는 connect =false 로 설정한다.

<Code Block 2-(4)>

```
            if( connected == true ) {
                        System.out.println("원격지에 접속 가능. " + addr[0]);
            }
            else {
                        System.out.println("찾을 수 없음");
            }
}finally {
            try {
                        if(socket != null)
                                    socket.close();

                        scanner.close();
            }catch(IOException e) {
                        System.out.println("서버와 통신 중 오류가 발생했습니다.");
            }
}
```

마지막 단계로 connected를 확인하여 true이면 원격지에 접속이 가능하다. false이면 찾을 수없음 이라고 출력 설정. 그리고 FTP와 마찬가지로 try catch문을통해 서버와 통신오류 발생시 문구 출력