응용 애플리케이션 개발을 위한 자바 프로그래밍

네트워크 프로그래밍

한국기술교육대학교
온라인평생교육원

학습내용

- 네크워크 관련 클래스
- 소켓 프로그래밍

학습목표

- 네트워크 통신의 개념을 이해하고, 인터넷 관련 클래스를 활용하여 프로그램을 작성할 수 있다.
- 소켓(Socket)을 이용한 서버/클라이언트 프로그램을 작성할 수 있다.

네트워크 관련 클래스

1 네트워크의 개요

① 네트워크 관련 용어

1 IP(Internet Protocol) 주소

- 인터넷에 연결된 장치를 식별하는 유일한 번호
- IPv4는 4개의 숫자가 점(.)으로 구분된 형태
- 각 숫자는 0~255사이의 값을 가짐

2 포트(Port) 번호

- 상호 통신을 하기 위해 사용하는 논리적인 통신 연결 번호
- 0~65,535번 중에 1024번 이후를 사용하여 애플리케이션 작성

1 네트워크의 개요

① 네트워크 관련 용어

3 프로토콜(Protocol)

- 상호 통신을 하기 위한 통신 규약
- 전송 계층 프로토콜 : TCP, UDP등
- 응용 계층 프로토콜 : HTTP, FTP등

4 패킷(Packet)

- 데이터 전송 단위
- 송수신 데이터를 패킷 단위로 분리하여 전송

네트워크 관련 클래스

1 네트워크의 개요

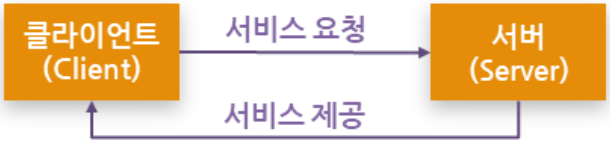
〈2〉 네트워크 통신 방식

1 인터넷의 기본 통신 방식

서버(Server)/클라이언트(Client) 방식

서버 : 서비스를 제공하는 측의 컴퓨터 또는 애플리케이션

클라이언트 : 서비스를 제공받는 측의 컴퓨터 또는 애플리케이션



1 네트워크의 개요

〈2〉 네트워크 통신 방식

1 인터넷의 기본 통신 방식

IP주소와 포트 번호를 사용하는 TCP 또는 UDP 프로토콜을 이용하여 패킷 단위로 통신

2 상호 통신을 위한 필수 정보

송신측의 IP 주소와 포트 번호, 수신측의 IP 주소와 포트 번호

네트워크 관련 클래스

1 네트워크의 개요

〈3〉 URL(Uniform Resource Locator)

1 인터넷상에 있는 웹 문서 또는 파일들의 위치를 표시하는 표준 방식

프로토콜://호스트:[포트번호]/[폴더또는 파일]/[질의]#[섹션]

프로토콜
http, ftp등

1 네트워크의 개요

〈3〉 URL(Uniform Resource Locator)

1 인터넷상에 있는 웹 문서 또는 파일들의 위치를 표시하는 표준 방식

프로토콜://호스트:[포트번호]/[폴더또는 파일]/[질의]#[섹션]

호스트 이름 :
IP 주소 또는 도메인(Domain) 이름

네트워크 관련 클래스

1 네트워크의 개요

〈3〉 URL(Uniform Resource Locator)

1 인터넷상에 있는 웹 문서 또는 파일들의 위치를 표시하는 표준 방식

프로토콜://호스트:[포트번호]/[폴더또는 파일]/[질의]#[섹션]

포트 번호(생략 가능)

1 네트워크의 개요

〈3〉 URL(Uniform Resource Locator)

1 인터넷상에 있는 웹 문서 또는 파일들의 위치를 표시하는 표준 방식

프로토콜://호스트:[포트번호]/[폴더또는 파일]/[질의]#[섹션]

웹 문서 또는 파일의 이름
(폴더 포함, 생략 가능)

네트워크 관련 클래스

1 네트워크의 개요

〈3〉 URL(Uniform Resource Locator)

1 인터넷상에 있는 웹 문서 또는 파일들의 위치를 표시하는 표준 방식

프로토콜://호스트:[포트번호]/[폴더또는 파일]/[질의]#[섹션]

질의 및 값 전달 부분
(생략 가능)

1 네트워크의 개요

〈3〉 URL(Uniform Resource Locator)

1 인터넷상에 있는 웹 문서 또는 파일들의 위치를 표시하는 표준 방식

프로토콜://호스트:[포트번호]/[폴더또는 파일]/[질의]#[섹션]

문서 내부에 표시된 위치
(생략 가능)

네트워크 관련 클래스

2 인터넷 관련 클래스

⟨1⟩ InetAddress 클래스

1 IP 주소를 표현한 클래스

2 패키지 : java.net.*

주요 메소드	static InetAddress getByName(String host)
설 명	• 문자열로 호스트(도메인) 이름을 지정하여 InetAddress 객체를 반환 • 빈 문자열을 지정하면 로컬 호스트의 InetAddress 객체를 반환 • UnKnownHostException 발생

주요 메소드	String getHostAddress()
설 명	IP 주소를 문자열로 반환

2 인터넷 관련 클래스

⟨1⟩ InetAddress 클래스

1 IP 주소를 표현한 클래스

2 패키지 : java.net.*

주요 메소드	String getHostAddress()
설 명	IP 주소를 문자열로 반환

네트워크 관련 클래스

2 인터넷 관련 클래스

〈2〉 URL 클래스

1 URL을 표현한 클래스

2 패키지 : java.net.*

주요 생성자 메소드	설 명
URL(String spec)	• URL을 문자열로 지정하여 객체 생성 • MalformedURLException 예외 발생

2 인터넷 관련 클래스

〈2〉 URL 클래스

주요 메소드	설 명
String getFile()	URL의 파일 이름을 반환
String getHost()	URL의 호스트 이름을 반환
String getPath()	URL의 폴더 경로 이름을 반환
int getPort()	URL의 포트 번호를 반환
String getProtocol()	URL의 프로토콜을 반환
String getQuery()	URL의 질의 부분을 반환

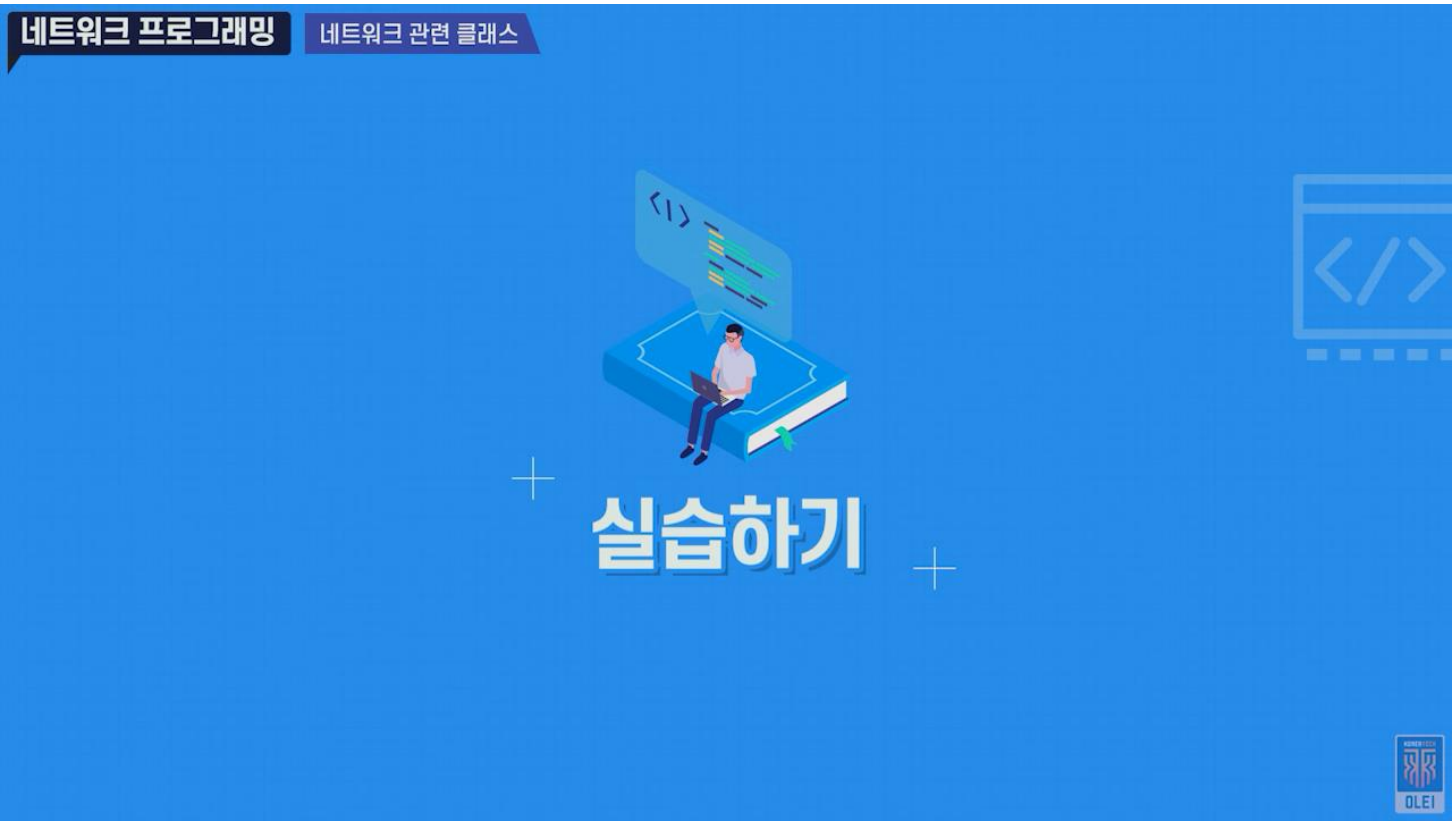
네트워크 관련 클래스

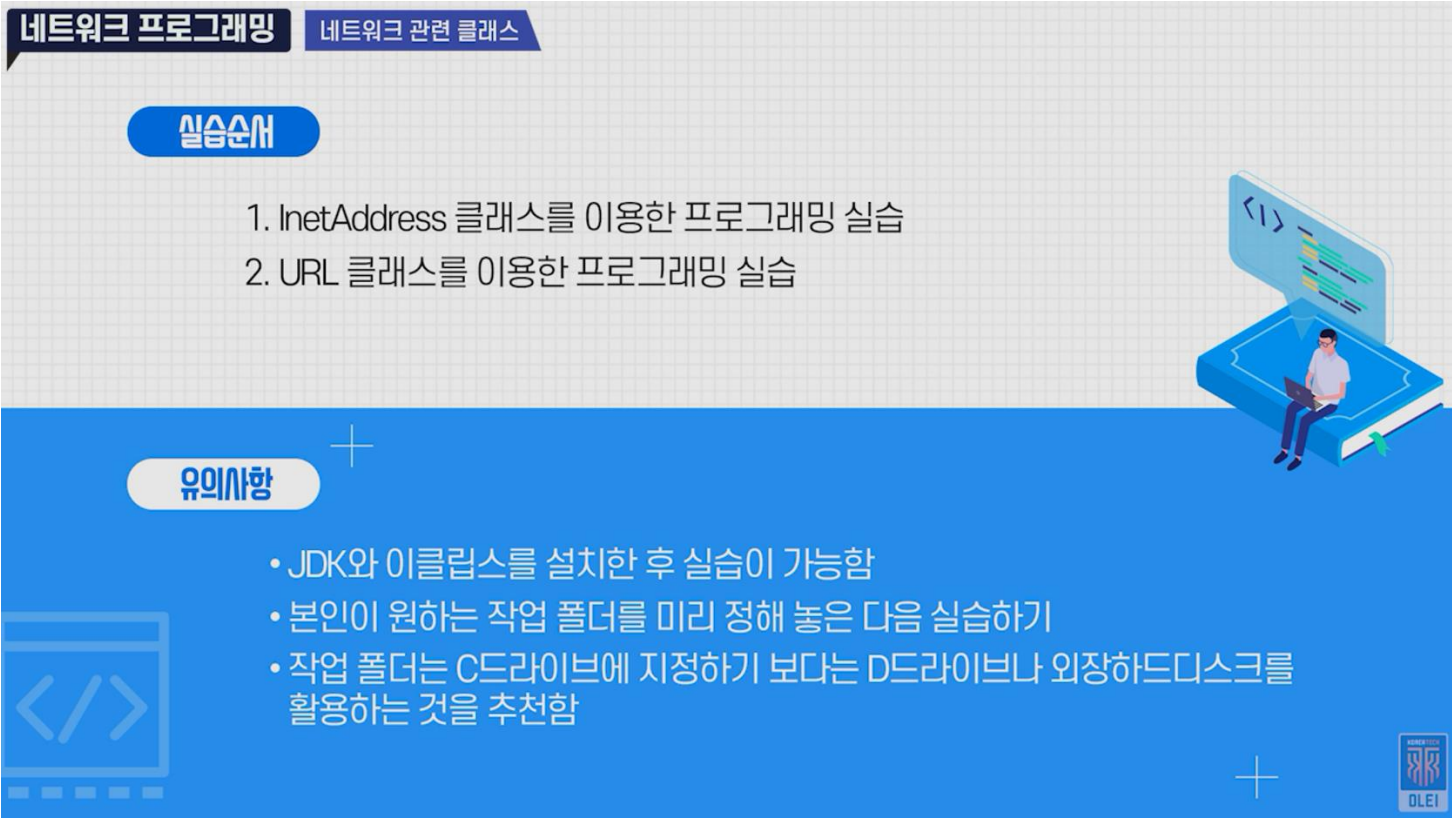
2 인터넷 관련 클래스

〈2〉 URL 클래스

주요 메소드	설 명
String getRef( )	URL의 섹션을 반환
URLConnection openConnection( )	URL에 대한 연결, URLConnection 객체 반환 - IOException 예외 발생
InputStream openStream( )	URL에 대한 연결, InputStream 객체를 반환 - IOException 예외 발생

네트워크 관련 클래스





※ 제공되는 실습 코드를 다운받아 실습해보시기 바랍니다.

소켓 프로그래밍

1 소켓 프로그래밍의 개요

⟨1⟩ 소켓(Socket)

1. TCP/IP 프로토콜을 사용하는 네트워크 프로그램을 작성하는데 표준으로 사용
2. 하위의 복잡한 프로토콜과 상관없이 프로그래밍 가능
3. 클라이언트가 서버에 접속할 때 생성
4. 서버와 클라이언트 사이의 데이터 전송로 역할

1 소켓 프로그래밍의 개요

⟨2⟩ TCP(Transmission Control Protocol) 소켓 프로그래밍

1. 서버와 클라이언트의 연결을 유지한 상태에서 데이터 송수신
2. 관련 클래스

서버 소켓 : java.net.ServerSocket

클라이언트 소켓 : java.net.Socket

소켓 프로그래밍

1 소켓 프로그래밍의 개요

〈3〉 UDP(User Datagram Protocol) 소켓 프로그래밍

1 서버와 클라이언트의 연결을 유지하지 않은 상태에서 데이터 송수신

2 관련 클래스

- java.net.DatagramSocket
- java.net.DatagramPacket

2 TCP 소켓 프로그래밍

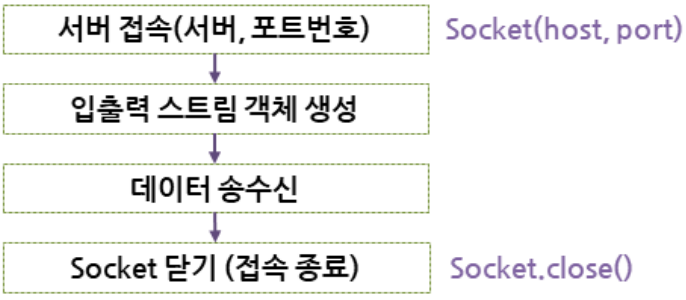
〈1〉 TCP 서버 프로그램 작성 순서

서버 시작(포트번호 지정)	ServerSocket(port)
↓	
클라이언트 접속 대기	ServerSocket.accept()
↓	
Socket 객체 생성(접속 연결)	
↓	
입출력 스트림 객체 생성	
↓	
데이터 송수신	
↓	
Socket 닫기 (접속 종료)	Socket.close()
↓	
ServerSocket 닫기 (서버 종료)	ServerSocket.close()

소켓 프로그래밍

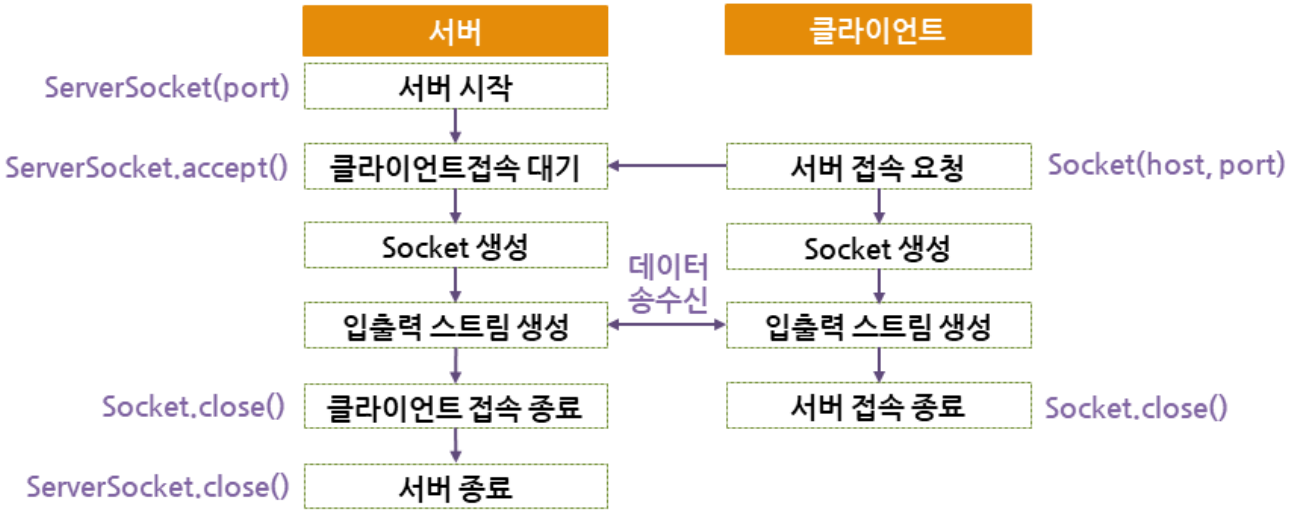
2 TCP 소켓 프로그래밍

〈2〉 TCP 클라이언트 프로그램 작성 순서



2 TCP 소켓 프로그래밍

〈3〉 TCP 소켓 프로그램 동작



소켓 프로그래밍

2 TCP 소켓 프로그래밍

④ ServerSocket 클래스

1. TCP 서버 소켓을 구현한 클래스
2. 네트워크를 통해 들어오는 요청을 대기
3. Socket 객체를 반환

주요 생성자 메소드	설 명
ServerSocket(int port)	지정한 포트 번호로 서버 소켓을 생성 (서버 시작) - IOException 예외 발생

2 TCP 소켓 프로그래밍

④ ServerSocket 클래스

1. TCP 서버 소켓을 구현한 클래스
2. 네트워크를 통해 들어오는 요청을 대기
3. Socket 객체를 반환

주요 메소드	설 명
Socket accept( )	클라이언트 접속 요청을 대기하고 있다가 요청이 들어오면 Socket 객체를 반환
void close( )	서버 소켓 닫기(서버 종료)

소켓 프로그래밍

2 TCP 소켓 프로그래밍

⑤ Socket 클래스

1 TCP 클라이언트 소켓을 표현한 클래스

2 송수신을 위한 데이터 전송로

주요 생성자 메소드	설 명
Socket(String host, int port)	지정한 서버 주소와 포트 번호로 객체를 생성 - 서버에 접속을 요청함 - IOException 예외 발생

2 TCP 소켓 프로그래밍

⑤ Socket 클래스

1 TCP 클라이언트 소켓을 구현한 클래스

2 송수신을 위한 데이터 전송로

주요 메소드	설 명
void close( )	소켓 닫기(접속 종료)
InputStream getInputStream( )	Socket으로 부터 InputStream 객체를 반환 서버의 데이터를 읽는 스트림
OutputStream getOuputStream( )	Socket으로 부터 OutputStream 객체를 반환 서버로 데이터를 전송하는 스트림

소켓 프로그래밍

2 TCP 소켓 프로그래밍

⑥ 단일 접속 소켓 프로그램

1 서버와 클라이언트가 1:1로 동작

2 서버에서 단일 접속만 처리

서버

```
ServerSocket server = new ServerSocket(8080);   ← 서버 시작
Socket client = server.accept();

InputStream is = client.getInputStream();
BufferedReader br = new BufferedReader(new InputStreamReader(is));

String receive = null;
while( (receive = br.readLine() ) == null );

output.append("읽음 : "+receive+"₩n");

client.close();
server.close();
```

클라이언트

```
Socket server = new Socket("서버주소", 8080);

OutputStream os = server.getOutputStream();
PrintWriter pw = new PrintWriter(os);

pw.println(text.getText());
pw.flush();

server.close();
```

2 TCP 소켓 프로그래밍

⑥ 단일 접속 소켓 프로그램

1 서버와 클라이언트가 1:1로 동작

2 서버에서 단일 접속만 처리

서버

```
ServerSocket server = new ServerSocket(8080);
Socket client = server.accept();   ← 접속대기

InputStream is = client.getInputStream();
BufferedReader br = new BufferedReader(new InputStreamReader(is));

String receive = null;
while( (receive = br.readLine() ) == null );

output.append("읽음 : "+receive+"₩n");

client.close();
server.close();
```

클라이언트

```
Socket server = new Socket("서버주소", 8080);

OutputStream os = server.getOutputStream();
PrintWriter pw = new PrintWriter(os);

pw.println(text.getText());
pw.flush();

server.close();
```

소켓 프로그래밍

2 TCP 소켓 프로그래밍

⑥ 단일 접속 소켓 프로그램

1 서버와 클라이언트가 1:1로 동작

2 서버에서 단일 접속만 처리

서버

```
ServerSocket server = new ServerSocket(8080);
Socket client = server.accept();

InputStream is = client.getInputStream();
BufferedReader br = new BufferedReader(new InputStreamReader(is));

String receive = null;
while( (receive = br.readLine() ) == null );

output.append("읽음 : "+receive+"₩n");

client.close();
server.close();
```

접속

클라이언트

```
Socket server = new Socket("서버주소", 8080);

OutputStream os = server.getOutputStream();
PrintWriter pw = new PrintWriter(os);

pw.println(text.getText());
pw.flush();

server.close();
```

2 TCP 소켓 프로그래밍

⑥ 단일 접속 소켓 프로그램

1 서버와 클라이언트가 1:1로 동작

2 서버에서 단일 접속만 처리

서버

```
ServerSocket server = new ServerSocket(8080);
Socket client = server.accept();

InputStream is = client.getInputStream();
BufferedReader br = new BufferedReader(new InputStreamReader(is));

String receive = null;
while( (receive = br.readLine() ) == null );

output.append("읽음 : "+receive+"₩n");

client.close();
server.close();
```

입력 스트림 생성

클라이언트

```
Socket server = new Socket("서버주소", 8080);

OutputStream os = server.getOutputStream();
PrintWriter pw = new PrintWriter(os);

pw.println(text.getText());
pw.flush();

server.close();
```

소켓 프로그래밍

2 TCP 소켓 프로그래밍

⑥ 단일 접속 소켓 프로그램

1 서버와 클라이언트가 1:1로 동작

2 서버에서 단일 접속만 처리

서버

```
ServerSocket server = new ServerSocket(8080);
Socket client = server.accept();

InputStream is = client.getInputStream();
BufferedReader br = new BufferedReader(new InputStreamReader(is));

String receive = null;
while( (receive = br.readLine() ) == null );

output.append("읽음 : "+receive+"₩n");

client.close();
server.close();
```

클라이언트

```
Socket server = new Socket("서버주소", 8080);

OutputStream os = server.getOutputStream();
PrintWriter pw = new PrintWriter(os);

pw.println(text.getText());
pw.flush();

server.close();
```

출력 스트림 생성

2 TCP 소켓 프로그래밍

⑥ 단일 접속 소켓 프로그램

1 서버와 클라이언트가 1:1로 동작

2 서버에서 단일 접속만 처리

서버

```
ServerSocket server = new ServerSocket(8080);
Socket client = server.accept();

InputStream is = client.getInputStream();
BufferedReader br = new BufferedReader(new InputStreamReader(is));

String receive = null;
while( (receive = br.readLine() ) == null );

output.append("읽음 : "+receive+"₩n");

client.close();
server.close();
```

데이터 읽기

클라이언트

```
Socket server = new Socket("서버주소", 8080);

OutputStream os = server.getOutputStream();
PrintWriter pw = new PrintWriter(os);

pw.println(text.getText());
pw.flush();

server.close();
```

소켓 프로그래밍

2 TCP 소켓 프로그래밍

⑥ 단일 접속 소켓 프로그램

1 서버와 클라이언트가 1:1로 동작

2 서버에서 단일 접속만 처리

서버

```
ServerSocket server = new ServerSocket(8080);
Socket client = server.accept();

InputStream is = client.getInputStream();
BufferedReader br = new BufferedReader(new InputStreamReader(is));

String receive = null;
while( (receive = br.readLine() ) == null );

output.append("읽음 : "+receive+"₩n");

client.close();
server.close();
```

클라이언트

```
Socket server = new Socket("서버주소", 8080);

OutputStream os = server.getOutputStream();
PrintWriter pw = new PrintWriter(os);

pw.println(text.getText());
pw.flush();

server.close();
```

← 데이터 전송

2 TCP 소켓 프로그래밍

⑥ 단일 접속 소켓 프로그램

1 서버와 클라이언트가 1:1로 동작

2 서버에서 단일 접속만 처리

서버

```
ServerSocket server = new ServerSocket(8080);
Socket client = server.accept();

InputStream is = client.getInputStream();
BufferedReader br = new BufferedReader(new InputStreamReader(is));

String receive = null;
while( (receive = br.readLine() ) == null );

output.append("읽음 : "+receive+"₩n");

client.close();
server.close();
```

← 읽은 데이터 출력

클라이언트

```
Socket server = new Socket("서버주소", 8080);

OutputStream os = server.getOutputStream();
PrintWriter pw = new PrintWriter(os);

pw.println(text.getText());
pw.flush();

server.close();
```

소켓 프로그래밍

2 TCP 소켓 프로그래밍

⑥ 단일 접속 소켓 프로그램

1 서버와 클라이언트가 1:1로 동작

2 서버에서 단일 접속만 처리

서버

```
ServerSocket server = new ServerSocket(8080);
Socket client = server.accept();

InputStream is = client.getInputStream();
BufferedReader br = new BufferedReader(new InputStreamReader(is));

String receive = null;
while( (receive = br.readLine() ) == null );

output.append("읽음 : "+receive+"₩n");

client.close();
server.close();
```

← 접속 종료 및 서버 종료

클라이언트

```
Socket server = new Socket("서버주소", 8080);

OutputStream os = server.getOutputStream();
PrintWriter pw = new PrintWriter(os);

pw.println(text.getText());
pw.flush();

server.close();
```

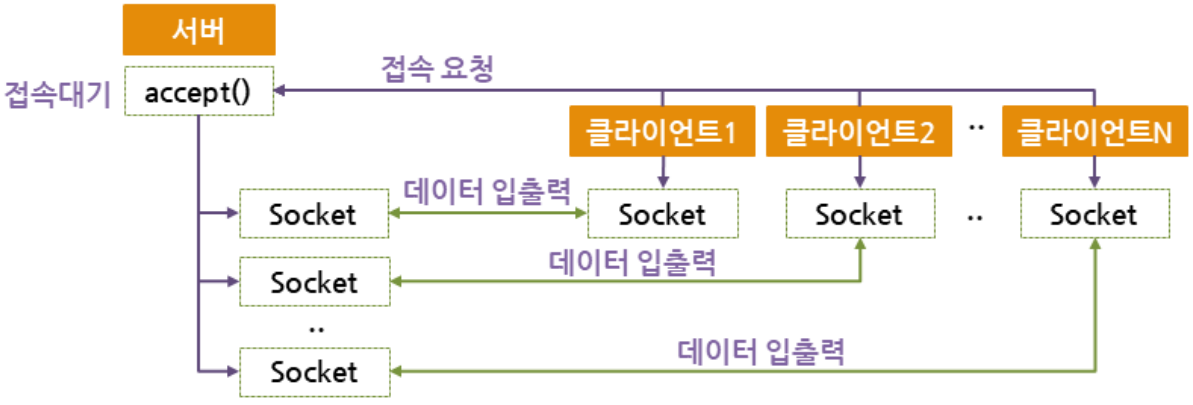
← 접속 종료

2 TCP 소켓 프로그래밍

⑦ 다중 접속 소켓 프로그램

1 서버와 클라이언트가 1:N로 동작

2 접속되는 각 클라이언트는 스레드로 처리



소켓 프로그래밍

네트워크 프로그래밍 | 소켓 프로그래밍

실습하기

네트워크 프로그래밍 | 소켓 프로그래밍

실습순서

1. TCP 소켓 프로그래밍 실습
   1) TCP 단일 접속 프로그래밍
   2) TCP 다중 접속 프로그래밍

유의사항

- JDK와 이클립스를 설치한 후 실습이 가능함
- 본인이 원하는 작업 폴더를 미리 정해 놓은 다음 실습하기
- 작업 폴더는 C드라이브에 지정하기 보다는 D드라이브나 외장하드디스크를 활용하는 것을 추천함

※ 제공되는 실습 코드를 다운받아 실습해보시기 바랍니다.

소켓 프로그래밍

3 UDP 소켓 프로그래밍

〈1〉 UDP 소켓 프로그래밍의 특징

1 TCP와 UDP 비교

구 분	TCP	UDP
연결	연결 지향 프로토콜 (Connection-oriented protocol)	비연결 지향 프로토콜 (Connection-less protocol)
신뢰성	패킷의 전송 순서 보장됨	패킷의 전송 순서 보장 안됨
전송속도	늦음	빠름
용도	HTTP, Email, FTP등	실시간 멀티미디어 통신, DNS등

3 UDP 소켓 프로그래밍

〈1〉 UDP 소켓 프로그래밍의 특징

2 UDP 소켓 프로그래밍 방법

- 서버와 클라이언트 소켓 구분이 없음
- DatagramPacket : 목적지 주소, 포트 번호, 데이터를 가짐
- DatagramPacket을 DatagramSocket을 이용하여 전송
- DatagramSocket은 접속 설정 없음

소켓 프로그래밍

3 UDP 소켓 프로그래밍

〈2〉 DatagramPacket 클래스

1 UDP 소켓 프로그래밍 방법

2 전송 데이터는 바이트 배열로 지정

3 목적지 주소는 InetAddress를 사용

주요 생성자 메소드	설 명
DatagramPacket(byte[] buf, int length, InetAddress address, int port)	• buf : 전송 데이터의 배열 • length : 전송 데이터의 크기 • address : 목적지 주소의 InetAddress 객체 • port : 포트 번호 * 송신용 패킷 생성

3 UDP 소켓 프로그래밍

〈2〉 DatagramPacket 클래스

1 UDP 소켓 프로그래밍 방법

2 전송 데이터는 바이트 배열로 지정

3 목적지 주소는 InetAddress를 사용

주요 생성자 메소드	설 명
DatagramPacket(byte[] buf, int length)	전송할 데이터와 크기를 지정하여 객체 생성 * 수신용 패킷 생성

소켓 프로그래밍

3 UDP 소켓 프로그래밍

〈2〉 DatagramPacket 클래스

주요 메소드	설 명
InetAddress getAddress() void setAddress(InetAddress iaddr)	패킷의 주소를 InetAddress 객체로 반환하거나 지정
byte[] getData() void setData(byte[] buf)	패킷의 데이터를 바이트 배열로 반환하거나 지정
int getLength() void setLength(int length)	패킷의 데이터 크기를 반환하거나 지정
int getPort() void setPort(int iport)	패킷의 포트 번호를 반환하거나 지정

3 UDP 소켓 프로그래밍

〈3〉 DatagramSocket 클래스

1 데이터그램 패킷을 송수신하기 위한 소켓을 표현한 클래스

주요 생성자 메소드	설 명
DatagramSocket( )	송신용 소켓 생성에 사용 • SocketException 발생
DatagramSocket(int port)	포트 번호를 지정하여 객체 생성 • 수신용 소켓 생성에 사용 • SocketException 발생

소켓 프로그래밍

3 UDP 소켓 프로그래밍

〈3〉 DatagramSocket 클래스

1 데이터그램 패킷을 송수신하기 위한 소켓을 표현한 클래스

주요 메소드	설 명
void close( )	소켓 닫기
void send(DatagramPacket p)	DatagramPacket을 전송
void receive(DatagramPacket p)	DatagramPacket을 수신

소켓 프로그래밍

네트워크 프로그래밍 | 소켓 프로그래밍

실습하기

네트워크 프로그래밍 | 소켓 프로그래밍

실습순서

1. UDP 소켓 프로그래밍 실습

유의사항

- JDK와 이클립스를 설치한 후 실습이 가능함
- 본인이 원하는 작업 폴더를 미리 정해 놓은 다음 실습하기
- 작업 폴더는 C드라이브에 지정하기 보다는 D드라이브나 외장하드디스크를 활용하는 것을 추천함

※ 제공되는 실습 코드를 다운받아 실습해보시기 바랍니다.

응용문제

네트워크 프로그래밍 응용문제

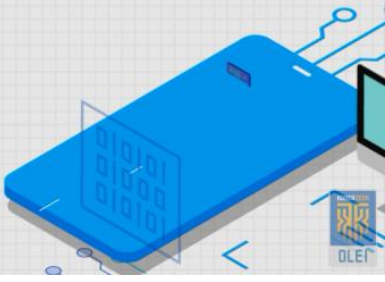
다음 실행화면과 조건에 맞게 프로그램을 작성하시오.

조건

1. 다중 접속 TCP 소켓 프로그램 작성
2. 서버 : TCP 소켓 프로그램
   - 포트 번호를 입력 받아 서버 실행
3. 클라이언트 : TCP 소켓 프로그램
   - 호스트 주소와 포트 번호를 입력 받아 서버에 접속

클래스명 : NetworkServer, NetworkClient

제공되는 실습 소스코드를 다운받아 실습해보시기 바랍니다.



네트워크 프로그래밍 응용문제

실행화면