KB금융그룹 | 국민의 평생 금융파트너

2024년 상반기 K-디지털 트레이닝

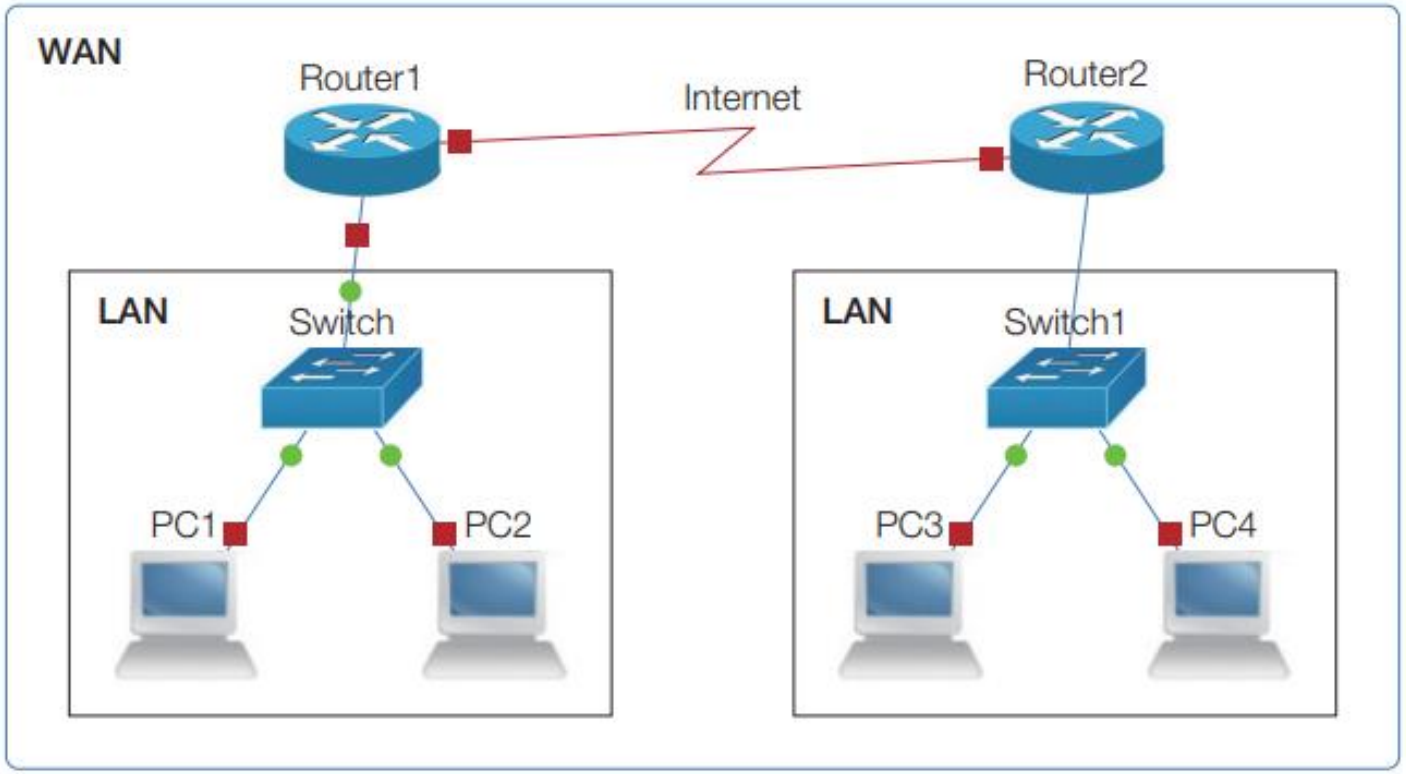
네트워크 입출력

[KB] IT's Your Life

KB 국민은행

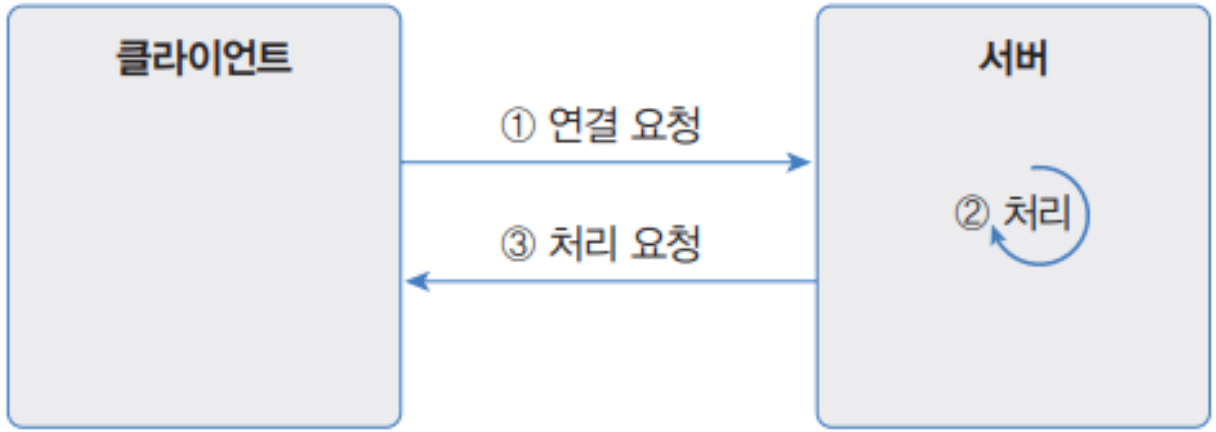
네트워크

- 네트워크: 여러 컴퓨터들을 통신 회선으로 연결한 것
- LAN: 가정, 회사, 건물, 특정 영역에 존재하는 컴퓨터를 연결한 것
- WAN: LAN을 연결한 것 = 인터넷



서버와 클라이언트

- 서버: 서비스를 제공하는 프로그램을
- 클라이언트: 서비스를 요청하는 프로그램
- 먼저 클라이언트가 서비스를 요청하고, 서버는 처리 결과를 응답으로 제공



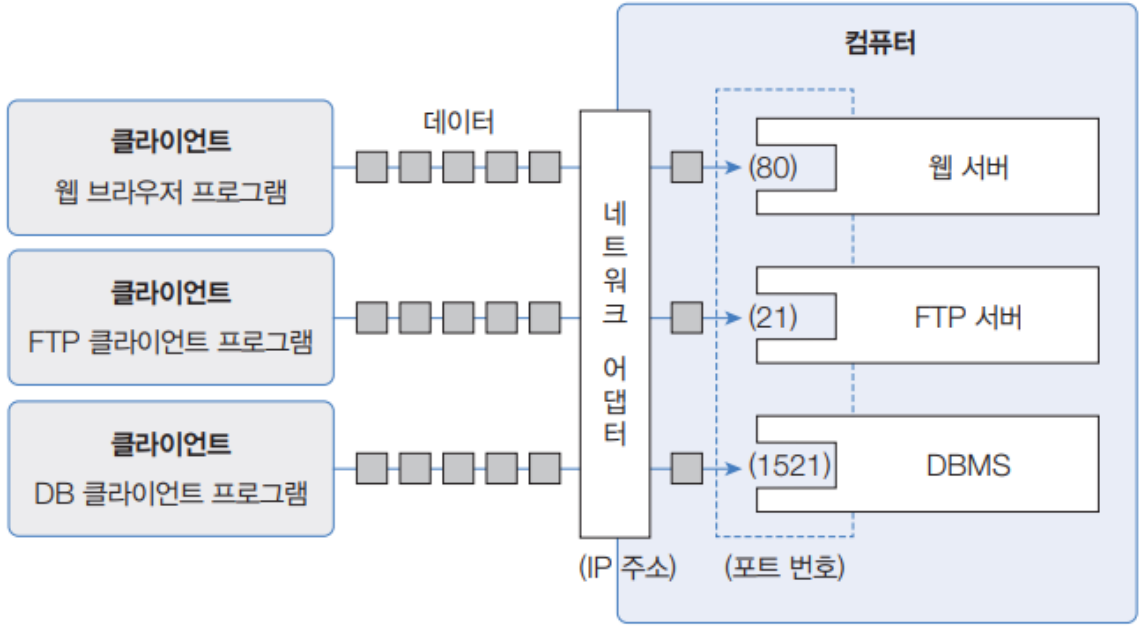
IP 주소

- IP 주소: 네트워크 어댑터(LAN 카드)마다 할당되는 컴퓨터의 고유한 주소
- ipconfig(윈도우), ifconfig(맥OS) 명령어로 네트워크 어댑터에 어떤 IP 주소가 부여되어 있는지 확인
- 프로그램은 DNS를 이용해서 컴퓨터의 IP 주소를 검색

```
>ipconfig

Windows IP 구성

이더넷 어댑터 이더넷:

   연결별 DNS 접미사. . . . : kornet
   링크-로컬 IPv6 주소 . . . . : fe80::56da:537:3ff2:68ee%7
   IPv4 주소 . . . . . . . . . . : 125.143.140.84
   서브넷 마스크 . . . . . . . . : 255.255.255.0
   기본 게이트웨이 . . . . . . . : 125.143.140.254


이더넷 어댑터 Bluetooth 네트워크 연결:

   미디어 상태 . . . . . . . . . . : 미디어 연결 끊김
   연결별 DNS 접미사. . . . :
```

Port 번호

- 운영체제가 관리하는 서버 프로그램의 연결 번호. 서버 시작 시 특정 Port 번호에 바인딩



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

InetAddress

- 자바는 IP 주소를 java.net 패키지의 InetAddress로 표현
- 로컬 컴퓨터의 InetAddress를 얻으려면 InetAddress.getLocalHost() 메소드를 호출

```
InetAddress ia = InetAddress.getLocalHost();
```

- getByName ( ) 메소드는 DNS에서 도메인 이름으로 등록된 단 하나의 IP 주소를 가져오고, getAllByName() 메소드는 등록된 모든 IP 주소를 배열로 가져옴

```
InetAddress ia = InetAddress.getByName(String domainName);
InetAddress[ ] iaArr = InetAddress.getAllByName(String domainName);
```

- InetAddress 객체에서 IP 주소를 얻으려면 getHostAddress () 메소드를 호출

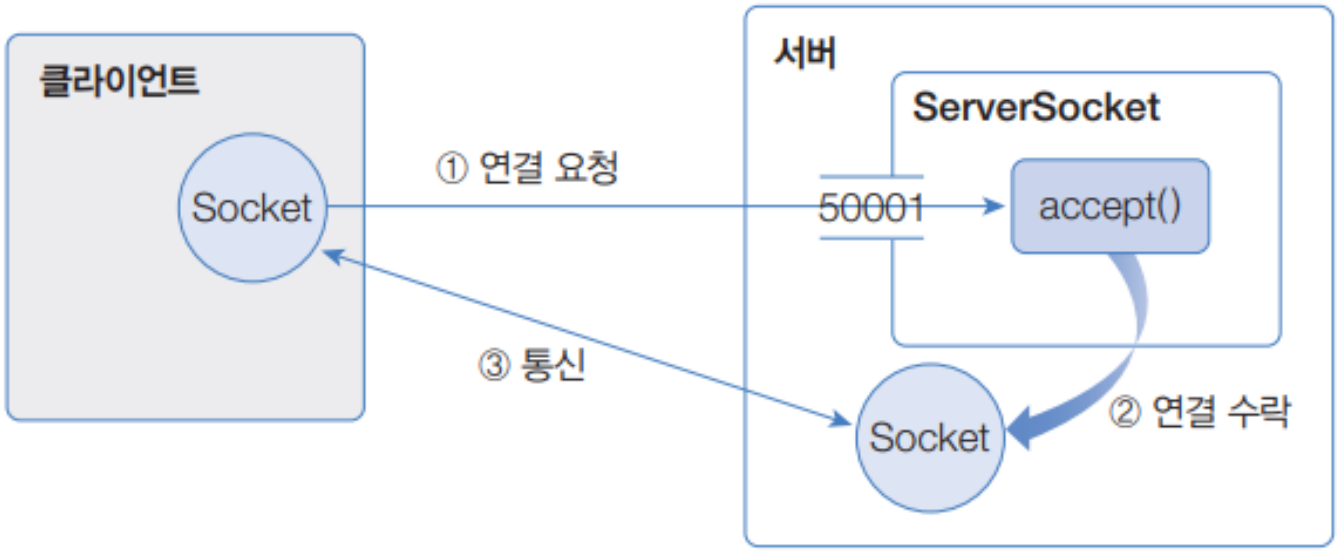
```
String ip = InetAddress.getHostAddress();
```

InetAddressExample.java

```
package ch19.sec02;

import java.net.InetAddress;
import java.net.UnknownHostException;

public class InetAddressExample {
  public static void main(String[] args) {
    try {
      InetAddress local = InetAddress.getLocalHost();
      System.out.println("내 컴퓨터 IP 주소: " + local.getHostAddress());

      InetAddress[] iaArr = InetAddress.getAllByName("www.naver.com");
      for(InetAddress remote : iaArr) {
        System.out.println("www.naver.com IP 주소: " + remote.getHostAddress());
      }
    } catch(UnknownHostException e) {
      e.printStackTrace();
    }
  }
}
```

TCP

- TCP는 연결형 프로토콜로, 상대방이 연결된 상태에서 데이터를 주고 받는 전송용 프로토콜
- 클라이언트가 연결 요청을 하고 서버가 연결을 수락하면 통신 회선이 고정되고, 데이터는 고정 회선을 통해 전달. TCP는 보낸 데이터가 순서대로 전달되며 손실이 발생하지 않음
- ServerSocket은 클라이언트의 연결을 수락하는 서버 쪽 클래스이고, Socket은 클라이언트에서 연결 요청할 때와 클라이언트와 서버 양쪽에서 데이터를 주고 받을 때 사용되는 클래스



TCP 서버

- TCP 서버 프로그램을 개발하려면 우선 ServerSocket 객체를 생성

```
ServerSocket serverSocket = new ServerSocket(50001);
```

- 기본 생성자로 객체를 생성하고 Port 바인딩을 위해 bind() 메소드를 호출해도 ServerSocket 생성

```
ServerSocket serverSocket = new ServerSocket();
serverSocket.bind(new InetSocketAddress(50001));
```

- 여러 개의 IP가 할당된 서버 컴퓨터에서 특정 IP에서만 서비스를 하려면 InetSocketAddress의 첫 번째 매개값으로 해당 IP를 줌

```
ServerSocket serverSocket = new ServerSocket();
serverSocket.bind( new InetSocketAddress("xxx.xxx.xxx.xxx", 50001) );
```

TCP 서버

- Port가 이미 다른 프로그램에서 사용 중이라면 BindException이 발생.
  - 다른 Port로 바인딩하거나 Port를 사용 중인 프로그램을 종료하고 다시 실행해야 함

- ServerSocket이 생성되면 연결 요청을 수락을 위해 accept( ) 메소드를 실행
  - accept()는 클라이언트가 연결 요청하기 전까지 블로킹(실행 멈춘 상태) 클라이언트의 연결 요청이 들어오면 블로킹이 해제되고 통신용 Socket을 리턴

```
Socket socket = serverSocket.accept();
```

TCP 서버

- 리턴된 Socket을 통해 연결된 클라이언트의 IP 주소와 Port 번호를 얻으려면 getRemoteSocketAddress ( ) 메소드를 호출해서 InetSocketAddress를 얻은 다음 getHostName()과

```
InetSocketAddress isa = (InetSocketAddress) socket.getRemoteSocketAddress();
String clientIp = isa.getHostName();
String portNo = isa.getPort();
```

-

```
serverSocket.close();
```

ServerExample.java

```
package ch19.sec03.exam01;

import java.io.IOException;
import java.net.InetSocketAddress;
import java.net.ServerSocket;
import java.net.Socket;
import java.util.Scanner;

public class ServerExample {
  private static ServerSocket serverSocket = null;

  public static void main(String[] args) {
    System.out.println("-----------------------------------------------------");
    System.out.println("서버를 종료하려면 q 또는 Q를 입력하고 Enter 키를 입력하세요.");
    System.out.println("-----------------------------------------------------");

    //TCP 서버 시작
    startServer();
```

ServerExample.java

```
    //키보드 입력
    Scanner scanner = new Scanner(System.in);
    while(true) {
      String key = scanner.nextLine();
      if(key.toLowerCase().equals("q")) {
        break;
      }
    }
    scanner.close();

    //TCP 서버 종료
    stopServer();
  }

  public static void stopServer() {
    try {
      //ServerSocket을 닫고 Port 언바인딩
      serverSocket.close();
      System.out.println("[서버] 종료됨 ");
    } catch (IOException e1) {}
  }
```

ServerExample.java

```
public static void startServer() {
  //작업 스레드 정의
  Thread thread = new Thread() {
    @Override
    public void run() {
      try {
        //ServerSocket 생성 및 Port 바인딩
        serverSocket = new ServerSocket(50001);
        System.out.println("[서버] 시작됨");

        while(true) {
          System.out.println( "\n[서버] 연결 요청을 기다림\n");
          //연결 수락
          Socket socket = serverSocket.accept();

          //연결된 클라이언트 정보 얻기
          InetSocketAddress isa = (InetSocketAddress) socket.getRemoteSocketAddress();
          System.out.println("[서버] " + isa.getHostName() + "의 연결 요청을 수락함");
```

ServerExample.java

```
            //연결 끊기
            socket.close();
            System.out.println("[서버] " + isa.getHostName() + "의 연결을 끊음");
          }
        } catch(IOException e) {
          System.out.println("[서버] " + e.getMessage());
        }
      }
    };
    //스레드 시작
    thread.start();
  }

}
```

TCP 클라이언트

- 클라이언트가 서버에 연결 요청을 하려면 Socket 객체를 생성할 때 생성자 매개값으로 서버 IP 주소와 Port 번호를 제공
- 로컬 컴퓨터에서 실행하는 서버로 연결 요청을 할 경우에는 IP 주소 대신 localhost 사용 가능

```
Socket socket = new Socket( "IP",  50001 );
```

- 도메인 이름을 사용하려면 DNS에서 IP 주소를 검색하는 생성자 매개값으로 InetSocketAddress 제공

```
Socket socket = new Socket( new InetSocketAddress("domainName", 50001) );
```

- 기본 생성자로 Socket을 생성한 후 connect() 메소드로 연결 요청 가능

```
socket = new Socket();
socket.connect( new InetSocketAddress("domainName", 50001) );
```

TCP 클라이언트

- 연결에 실패하면 예외 발생

```
try {
  Socket socket = new Socket("IP", 50001);
} catch (UnknownHostException e) {
  //IP 표기 방법이 잘못되었을 경우
} catch (IOException e) {
  //IP와 Port로 서버에 연결할 수 없는 경우
}
```

- 연결 끊기

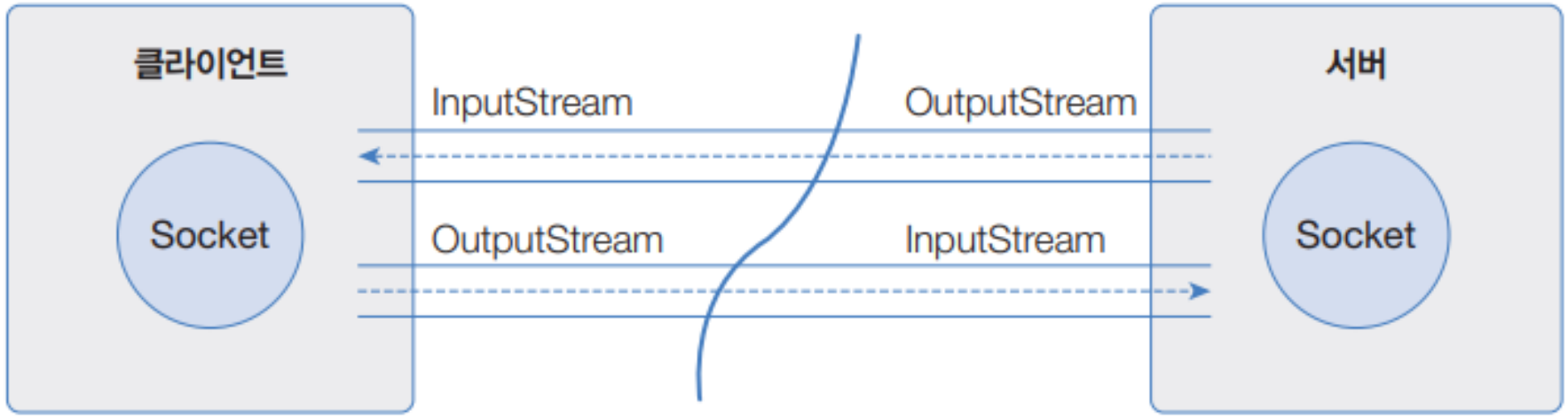
```
socket.close();
```

ClientExample.java

```
package ch19.sec03.exam01;

import java.io.IOException;
import java.net.Socket;
import java.net.UnknownHostException;

public class ClientExample {
  public static void main(String[] args) {
    try {
      //Socket 생성과 동시에 localhost의 50001 Port로 연결 요청;
      Socket socket = new Socket("localhost", 50001);

      System.out.println( "[클라이언트] 연결 성공");

      socket.close();       //Socket 닫기
      System.out.println("[클라이언트] 연결 끊음");
    } catch (UnknownHostException e) {
      //IP 표기 방법이 잘못되었을 경우
    } catch (IOException e) {
      //해당 포트의 서버에 연결할 수 없는 경우
    }
  }
}
```

입출력 스트림으로 데이터 주고 받기

- 클라이언트가 연결 요청(connect())을 하고 서버가 연결 수락(accept())했다면, 양쪽의 Socket 객체로부터 각각 InputStream과 OutputStream을 얻을 수 있다.




```
InputStream is = socket.getInputStream();
OutputStream os = socket.getOutputStream();
```

- 상대방에게 데이터를 보낼 때에는 보낼 데이터를 byte[ ] 배열로 생성하고, 이것을 매개값으로 해서 OutputStream의 write() 메소드를 호출

입출력 스트림으로 데이터 주고 받기

- 문자열을 좀 더 간편하게 보내고 싶다면 보조 스트림인 DataOutputStream을 연결해서 사용

```
String data = "보낼 데이터";
byte[ ] bytes = data.getBytes("UTF-8");
OutputStream os = socket.getOutputStream();
os.write(bytes);
os.flush();
```

```
String data = "보낼 데이터";
DataOutputStream dos = new DataOutputStream(socket.getOutputStream());
dos.writeUTF(data);
dos.flush();
```

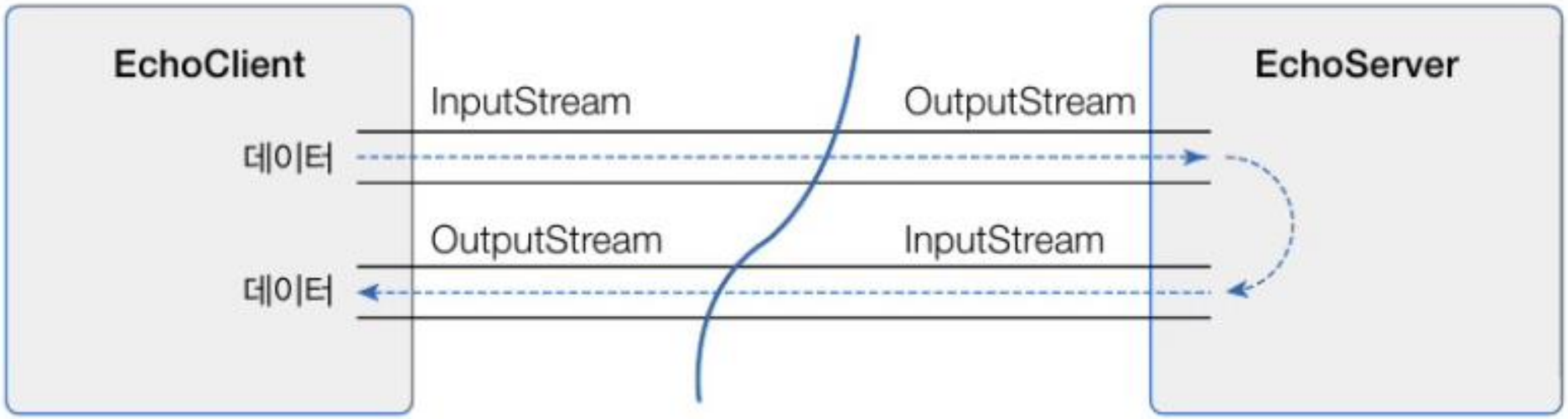
입출력 스트림으로 데이터 주고 받기

- 데이터를 받기 위해서는 받은 데이터를 저장할 byte[ ] 배열을 하나 생성하고, 이것을 매개값으로 해서 InputStream의 read() 메소드를 호출
- read() 메소드는 읽은 데이터를 byte[ ] 배열에 저장하고 읽은 바이트 수를 리턴
- 문자열을 좀 더 간편하게 받고 싶다면 보조 스트림인 DataInputStream을 연결해서 사용

```
byte[ ] bytes = new byte[1024];
InputStream is = socket.getInputStream();
int num = is.read(bytes);
String data = new String(bytes, 0, num, "UTF-8");
```

```
DataInputStream dis = new DataInputStream(socket.getInputStream());
String data = dis.readUTF();
```

에코 echo TCP 서버



EchoServer.java

```
package ch19.sec03.exam02;

import java.io.DataInputStream;
import java.io.DataOutputStream;
import java.io.IOException;
import java.io.InputStream;
import java.io.OutputStream;
import java.net.InetSocketAddress;
import java.net.ServerSocket;
import java.net.Socket;
import java.util.Scanner;

public class EchoServer {
  private static ServerSocket serverSocket = null;

  public static void main(String[] args) {
    System.out.println("-----------------------------------------------------------");
    System.out.println("서버를 종료하려면 q를 입력하고 Enter 키를 입력하세요.");
    System.out.println("-----------------------------------------------------------");

    //TCP 서버 시작
    startServer();
```

EchoServer.java

```
  //키보드 입력
  Scanner scanner = new Scanner(System.in);
  while(true) {
    String key = scanner.nextLine();
    if(key.toLowerCase().equals("q")) {
      break;
    }
  }
  scanner.close();

      //TCP 서버 종료
      stopServer();
}

public static void stopServer() {
  try {
    //ServerSocket을 닫고 Port 언바인딩
    serverSocket.close();
    System.out.println( "[서버] 종료됨 ");
  } catch (IOException e1) {}
}
```

EchoServer.java

```
public static void startServer() {
  //작업 스레드 정의
  Thread thread = new Thread() {
    @Override
    public void run() {
      try {
        //ServerSocket 생성 및 Port 바인딩
        serverSocket = new ServerSocket(50001);
        System.out.println( "[서버] 시작됨");

        //연결 수락 및 데이터 통신
        while(true) {
          System.out.println( "\n[서버] 연결 요청을 기다림\n");
          //연결 수락
          Socket socket = serverSocket.accept();

          //연결된 클라이언트 정보 얻기
          InetSocketAddress isa = (InetSocketAddress) socket.getRemoteSocketAddress();
          System.out.println("[서버] " + isa.getHostName() + "의 연결 요청을 수락함");
```

EchoServer.java

```
//-------------------------------------------------------------------------
/*
//데이터 받기
InputStream is = socket.getInputStream();
byte[] bytes = new byte[1024];
int readByteCount = is.read(bytes);
String message = new String(bytes, 0, readByteCount, "UTF-8");

//데이터 보내기
OutputStream os = socket.getOutputStream();
bytes = message.getBytes("UTF-8");
os.write(bytes);
os.flush();
System.out.println( "[서버] 받은 데이터를 다시 보냄: " + message);
*/
```

EchoServer.java

```
            //-------------------------------------------------------------------------
            //데이터 받기
            DataInputStream dis = new DataInputStream(socket.getInputStream());
            String message = dis.readUTF();

            //데이터 보내기
            DataOutputStream dos = new DataOutputStream(socket.getOutputStream());
            dos.writeUTF(message);
            dos.flush();
            System.out.println( "[서버] 받은 데이터를 다시 보냄: " + message);
            //-------------------------------------------------------------------------
            socket.close(); //연결 끊기
            System.out.println("[서버] " + isa.getHostName() + "의 연결을 끊음");
          }
        } catch(IOException e) {
          System.out.println("[서버] " + e.getMessage());
        }
      }
    };
    //스레드 시작
    thread.start();
  }
}
```

EchoClient.java

```
package ch19.sec03.exam02;

import java.io.DataInputStream;
import java.io.DataOutputStream;
import java.io.InputStream;
import java.io.OutputStream;
import java.net.Socket;

public class EchoClient {
  public static void main(String[] args) {
    try {
      //Socket 생성과 동시에 localhost의 50001 포트로 연결 요청;
      Socket socket = new Socket("localhost", 50001);

      System.out.println( "[클라이언트] 연결 성공");
```

EchoClient.java

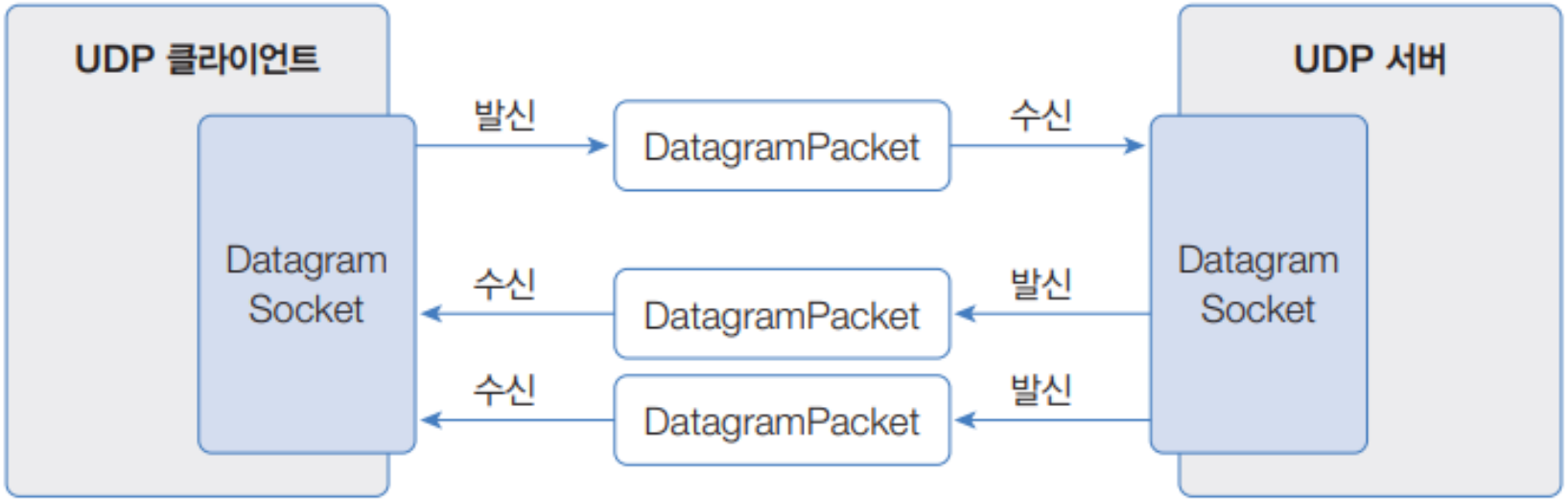
```
//-----------------------------------------------------------------------
/*
//데이터 보내기
String sendMessage = "나는 자바가 좋아~~";
OutputStream os = socket.getOutputStream();
byte[] bytes = sendMessage.getBytes("UTF-8");
os.write(bytes);
os.flush();
System.out.println("[클라이언트] 데이터 보냄: " + sendMessage);

//데이터 받기
InputStream is = socket.getInputStream();
bytes = new byte[1024];
int readByteCount = is.read(bytes);
String receiveMessage = new String(bytes, 0, readByteCount, "UTF-8");
System.out.println("[클라이언트] 데이터 받음: " + receiveMessage);
*/
```

EchoClient.java

```
        //-------------------------------------------------------------------
        //데이터 보내기
        String sendMessage = "나는 자바가 좋아~~";
        DataOutputStream dos = new DataOutputStream(socket.getOutputStream());
        dos.writeUTF(sendMessage);
        dos.flush();
        System.out.println("[클라이언트] 데이터 보냄: " + sendMessage);

        //데이터 받기
        DataInputStream dis = new DataInputStream(socket.getInputStream());
        String receiveMessage = dis.readUTF();
        System.out.println("[클라이언트] 데이터 받음: " + receiveMessage);
        //-------------------------------------------------------------------

        //연결 끊기
        socket.close();
        System.out.println("[클라이언트] 연결 끊음");
      } catch(Exception e) {
      }
    }
}
```

UDP

- 발신자가 일방적으로 수신자에게 데이터를 보내는 방식.
- TCP처럼 연결 요청 및 수락 과정이 없기 때문에 TCP보다 데이터 전송 속도가 상대적으로 빠름
- 데이터 전달의 신뢰성보다 속도가 중요하다면 UDP를 사용하고, 데이터 전달의 신뢰성이 중요하다면 TCP를 사용
- DatagramSocket은 발신점과 수신점에 해당하고 DatagramPacket은 주고받는 데이터에 해당



UDP 서버

- DatagramSocket 객체를 생성할 때에는 다음과 같이 바인딩할 Port 번호를 생성자 매개값으로 제공

```
DatagramSocket datagramSocket = new DatagramSocket(50001);
```

- receive() 메소드는 데이터를 수신할 때까지 블로킹되고, 데이터가 수신되면 매개값으로 주어진 DatagramPacket에 저장

```
DatagramPacket receivePacket = new DatagramPacket(new byte[1024], 1024);
datagramSocket.receive(receivePacket);
```

UDP 서버

- DatagramPacket 생성자의 첫 번째 매개값은 수신된 데이터를 저장할 배열이고 두 번째 매개값은 수신할 수 있는 최대 바이트 수

```
byte[] bytes = receivePacket.getData();
int num = receivePacket.getLength();
```

```
String data = new String(bytes, 0, num, "UTF-8");
```

- getSocketAddress () 메소드를 호출하면 정보가 담긴 SocketAddress 객체를 얻을 수 있음

```
SocketAddress socketAddress = receivePacket.getSocketAddress();
```

UDP 서버

- SocketAddress 객체는 클라이언트로 보낼 DatagramPacket을 생성할 때 네 번째 매개값으로 사용

```
String data = "처리 내용";
byte[] bytes = data.getBytes("UTF-8");
DatagramPacket sendPacket = new DatagramPacket( bytes, 0, bytes.length,
socketAddress );
```

- DatagramPacket을 클라이언트로 보낼 때는 DatagramSocket의 send() 메소드를 이용

```
datagramSocket.send( sendPacket );
```

- UDP 서버를 종료하고 싶을 경우에는 DatagramSocket의 close() 메소드를 호출

```
datagramSocket.close();
```

NewsServer.java

```
package ch19.sec04;

import java.net.DatagramPacket;
import java.net.DatagramSocket;
import java.net.SocketAddress;
import java.util.Scanner;

public class NewsServer {
  private static DatagramSocket datagramSocket = null;

  public static void main(String[] args) throws Exception {
    System.out.println("-----------------------------------------------------------");
    System.out.println("서버를 종료하려면 q를 입력하고 Enter 키를 입력하세요.");
    System.out.println("-----------------------------------------------------------");

    //UDP 서버 시작
    startServer();
```

NewsServer.java

```
  //키보드 입력
  Scanner scanner = new Scanner(System.in);
  while(true) {
    String key = scanner.nextLine();
    if(key.toLowerCase().equals("q")) {
      break;
    }
  }
  scanner.close();

  //TCP 서버 종료
  stopServer();
}

public static void stopServer() {
  //DatagramSocket을 닫고 Port 언바인딩
  datagramSocket.close();
  System.out.println( "[서버] 종료됨 ");
}
```

NewsServer.java

```
public static void startServer() {
  //작업 스레드 정의
  Thread thread = new Thread() {
    @Override
    public void run() {
      try {
        //DatagramSocket 생성 및 Port 바인딩
        datagramSocket = new DatagramSocket(50001);
        System.out.println( "[서버] 시작됨");

        while(true) {
          //클라이언트가 구독하고 싶은 뉴스 주제 얻기
          DatagramPacket receivePacket = new DatagramPacket(new byte[1024], 1024);
          datagramSocket.receive(receivePacket);
          String newsKind = new String(receivePacket.getData(), 0, receivePacket.getLength(), "UTF-8");

          //클라이언트의 IP와 Port 얻기
          SocketAddress socketAddress = receivePacket.getSocketAddress();
```

UDP 클라이언트

- 서버에 요청 내용을 보내고 그 결과를 받는 역할
- UDP 클라이언트를 위한 DatagramSocket 객체는 기본 생성자로 생성. Port 번호는 자동 부여

```
String data = "요청 내용";
byte[ ] bytes = data.getBytes("UTF-8");
DatagramPacket sendPacket = new DatagramPacket(
  bytes, bytes.length, new InetSocketAddress("localhost", 50001)
);
```

- 생성된 DatagramPacket을 매개값으로해서 DatagramSocket의 send() 메소드를 호출하면 UDP 서버로 DatagramPacket이 전송

```
datagramSocket.send(sendPacket);
```

- DatagramSocket을 닫으려면 close() 메소드를 호출

```
atagramSocket.close();
```

NewsClient.java

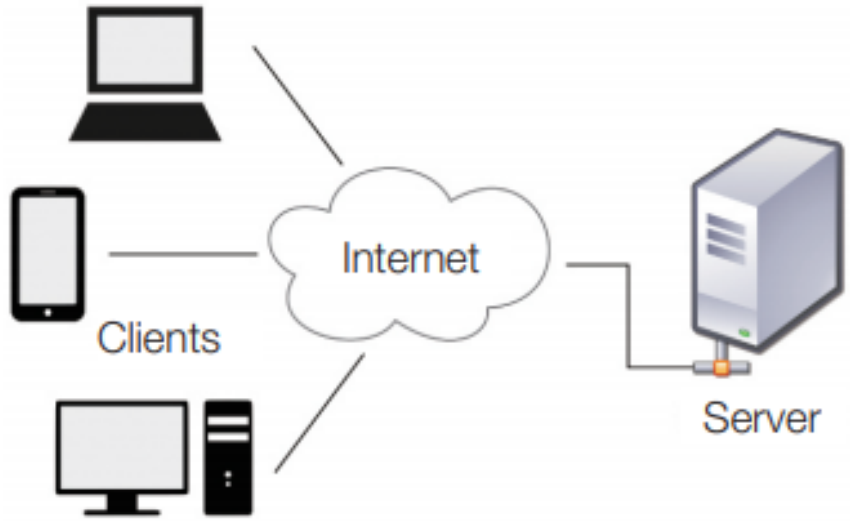
```
package ch19.sec04;

import java.net.DatagramPacket;
import java.net.DatagramSocket;
import java.net.InetSocketAddress;

public class NewsClient {
  public static void main(String[] args) {
    try {
      //DatagramSocket 생성
      DatagramSocket datagramSocket = new DatagramSocket();

      //구독하고 싶은 뉴스 주제 보내기
      String data = "정치";
      byte[] bytes = data.getBytes("UTF-8");
      DatagramPacket sendPacket = new DatagramPacket(
        bytes, bytes.length, new InetSocketAddress("localhost", 50001)
      );
      datagramSocket.send(sendPacket);
```

NewsClient.java

```
      while(true) {
        //뉴스 받기
        DatagramPacket receivePacket = new DatagramPacket(new byte[1024], 1024);
        datagramSocket.receive(receivePacket);

        //문자열로 변환
        String news = new String(receivePacket.getData(), 0, receivePacket.getLength(), "UTF-8");
        System.out.println(news);

        //10번째 뉴스를 받았을 경우 while 문 종료
        if(news.contains("뉴스10")) {
          break;
        }
      }

      //DatagramSocket 닫기
      datagramSocket.close();
    } catch(Exception e) {
    }
  }
}
```

서버의 동시 요청 처리

- 일반적으로 서버는 다수의 클라이언트와 통신.
- 서버는 클라이언트들로부터 동시에 요청을 받아서 처리하고, 처리 결과를 개별 클라이언트로 보내줌



서버의 동시 요청 처리

기존 서버의 문제점

- EchoServer

```
while(true) {
  //연결 수락
  Socket socket = serverSocket.accept();

  //데이터 받기
  ...
  //데이터 보내기
  ...
}
```

연결 수락 및 요청 처리

다음 연결 수락 및 요청 처리를 위한 반복

서버의 동시 요청 처리

- 기존 서버의 문제점
  - NewsServer

```
while(true) {
  //클라이언트가 구독하고 싶은 뉴스 종류 얻기
  DatagramPacket receivePacket = new DatagramPacket(new byte[1024], 1024);
  datagramSocket.receive(receivePacket);
  ...

  //10개의 뉴스를 클라이언트로 전송
  ...
}
```

요청 내용 받기 및 요청 처리

다음 요청 내용 받기 및 요청 처리를 위한 반복

서버의 동시 요청 처리

- accept()와 receive()를 제외한 요청 처리 코드
  → 별도의 스레드에서 작업

TCP 서버

```
while(true) {
  Socket socket = serverSocket.
  accept();

  //데이터 받기
  ...
  //데이터 보내기
  ...
}
```

스레드로 처리

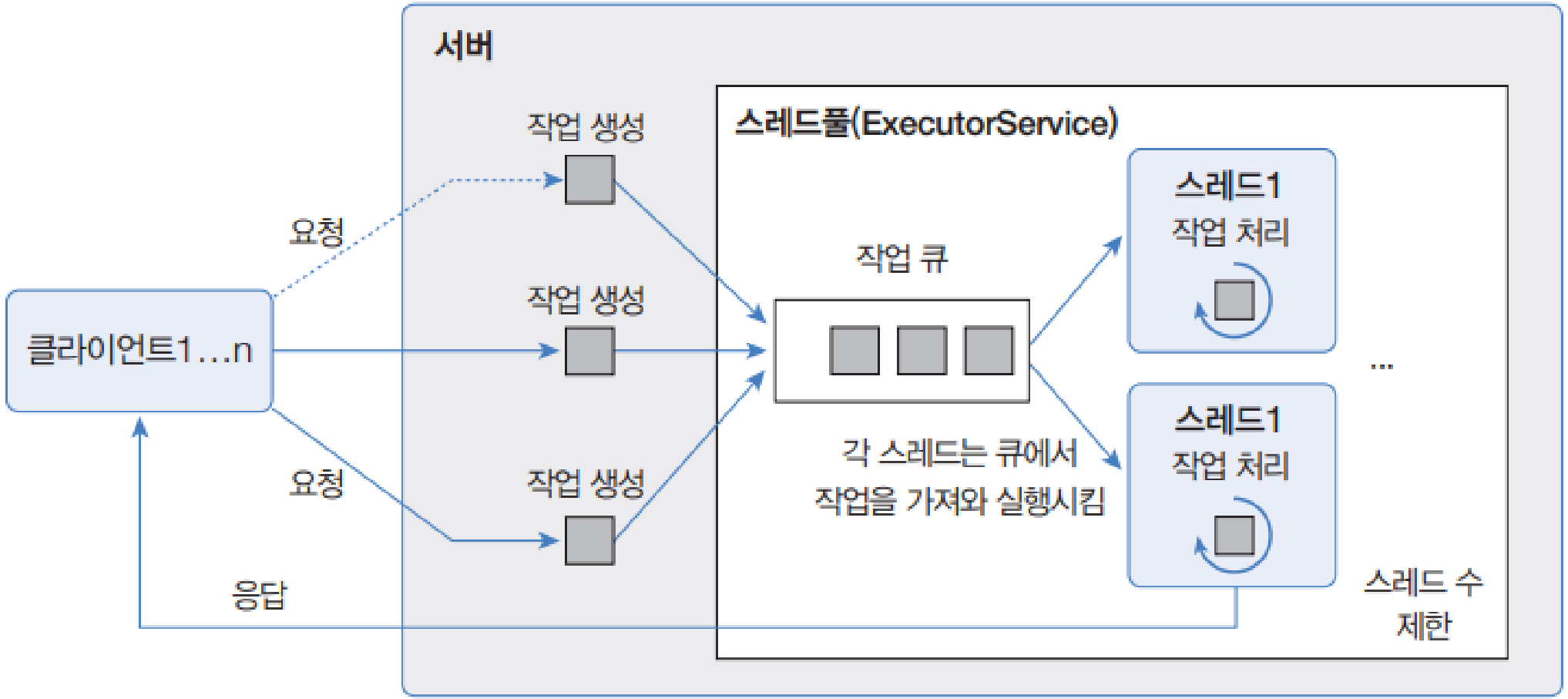
UDP 서버

```
while(true) {
  DatagramPacket receivePacket = ···
  datagramSocket.
receive(receivePacket);
  ...

  //10개의 뉴스를 클라이언트로 전송
  ...
}
```

스레드로 처리

서버의 동시 요청 처리

- 스레드를 처리할 때 클라이언트의 폭증으로 인한 서버의 과도한 스레드 생성을 방지
  --> 스레드풀을 사용



TCP EchoServer 동시 요청 처리

- 스레드풀을 이용해서 클라이언트의 요청을 동시에 처리

EchoServer.java

```
package ch19.sec05.exam01;

import java.io.DataInputStream;
import java.io.DataOutputStream;
import java.io.IOException;
import java.net.InetSocketAddress;
import java.net.ServerSocket;
import java.net.Socket;
import java.util.Scanner;
import java.util.concurrent.ExecutorService;
import java.util.concurrent.Executors;

public class EchoServer {
  private static ServerSocket serverSocket = null;
  private static ExecutorService executorService = Executors.newFixedThreadPool(10);

  public static void main(String[] args) {
    System.out.println("-----------------------------------------------------------");
    System.out.println("서버를 종료하려면 q를 입력하고 Enter 키를 입력하세요.");
    System.out.println("-----------------------------------------------------------");

    //TCP 서버 시작
    startServer();
```

EchoServer.java

```
  //키보드 입력
  Scanner scanner = new Scanner(System.in);
  while(true) {
    String key = scanner.nextLine();
    if(key.toLowerCase().equals("q")) {
      break;
    }
  }
  scanner.close();

  stopServer();       //TCP 서버 종료
}

public static void stopServer() {
  try {
    //ServerSocket을 닫고 Port 언바인딩
    serverSocket.close();
    executorService.shutdownNow();
    System.out.println( "[서버] 종료됨 ");
  } catch (IOException e1) {}
}
```

EchoServer.java

```
public static void startServer() {
  //작업 스레드 정의
  Thread thread = new Thread() {
    @Override
    public void run() {
      try {
        //ServerSocket 생성 및 Port 바인딩
        serverSocket = new ServerSocket(50001);
        System.out.println( "[서버] 시작됨\n");

        //연결 수락 및 데이터 통신
        while(true) {
          //연결 수락
          Socket socket = serverSocket.accept();

          executorService.execute(() -> {
            try {
              //연결된 클라이언트 정보 얻기
              InetSocketAddress isa = (InetSocketAddress) socket.getRemoteSocketAddress();
              System.out.println("[서버] " + isa.getHostName() + "의 연결 요청을 수락함");
```

EchoServer.java

```
                //데이터 받기
                DataInputStream dis = new DataInputStream(socket.getInputStream());
                String message = dis.readUTF();

                //데이터 보내기
                DataOutputStream dos = new DataOutputStream(socket.getOutputStream());
                dos.writeUTF(message);
                dos.flush();
                System.out.println( "[서버] 받은 데이터를 다시 보냄: " + message);

                //연결 끊기
                socket.close();
                System.out.println("[서버] " + isa.getHostName() + "의 연결을 끊음\n");
              } catch(IOException e) {
              }
            });
          }
        } catch(IOException e) {
          System.out.println("[서버] " + e.getMessage());
        }
      }
    };
    thread.start();     //스레드 시작
  }
}
```

UDP NewsServer 동시 요청 처리

- 스레드풀을 이용해서 클라이언트의 요청을 동시에 처리

NewsServer.java

```
package ch19.sec05.exam02;

import java.net.DatagramPacket;
import java.net.DatagramSocket;
import java.net.SocketAddress;
import java.util.Scanner;
import java.util.concurrent.ExecutorService;
import java.util.concurrent.Executors;

public class NewsServer {
  private static DatagramSocket datagramSocket = null;
  private static ExecutorService executorService = Executors.newFixedThreadPool(10);

  public static void main(String[] args) throws Exception {
    System.out.println("-----------------------------------------------------------");
    System.out.println("서버를 종료하려면 q 를 입력하고 Enter 키를 입력하세요.");
    System.out.println("-----------------------------------------------------------");

    //UDP 서버 시작
    startServer();
```

NewsServer.java

```
    //키보드 입력
    Scanner scanner = new Scanner(System.in);
    while(true) {
      String key = scanner.nextLine();
      if(key.toLowerCase().equals("q")) {
        break;
      }
    }
    scanner.close();

    //TCP 서버 종료
    stopServer();
}

public static void stopServer() {
    //DatagramSocket을 닫고 Port 언바인딩
    datagramSocket.close();
    executorService.shutdownNow();
    System.out.println( "[서버] 종료됨 ");
}
```

NewsServer.java

```
public static void startServer() {
  //작업 스레드 정의
  Thread thread = new Thread() {
    @Override
    public void run() {
      try {
        //DatagramSocket 생성 및 Port 바인딩
        datagramSocket = new DatagramSocket(50001);
        System.out.println( "[서버] 시작됨");

        while(true) {
          //클라이언트가 구독하고 싶은 뉴스 종류 얻기
          DatagramPacket receivePacket = new DatagramPacket(new byte[1024], 1024);
          datagramSocket.receive(receivePacket);

          executorService.execute(() -> {
            try {
             String newsKind = new String(receivePacket.getData(), 0, receivePacket.getLength(), "UTF-8");

              //클라이언트의 IP와 Port 얻기
              SocketAddress socketAddress = receivePacket.getSocketAddress();
```

NewsServer.java

```
                //10개의 뉴스를 클라이언트로 전송
                for(int i=1; i<=10; i++) {
                  String data = newsKind + ": 뉴스" + i;
                  byte[] bytes = data.getBytes("UTF-8");
                  DatagramPacket sendPacket = new DatagramPacket(bytes, 0, bytes.length, socketAddress);
                  datagramSocket.send(sendPacket);
                }
              } catch (Exception e) {
              }
            });
          }
        } catch (Exception e) {
          System.out.println("[서버] " + e.getMessage());
        }
      }
    };
    //스레드 시작
    thread.start();
  }

}
```

JSON

- 네트워크로 전달하는 데이터 형식
- 두 개 이상의 속성이 있으면 객체 { }로 표기. 두 개 이상의 값이 있으면 배열 [ ]로 표기

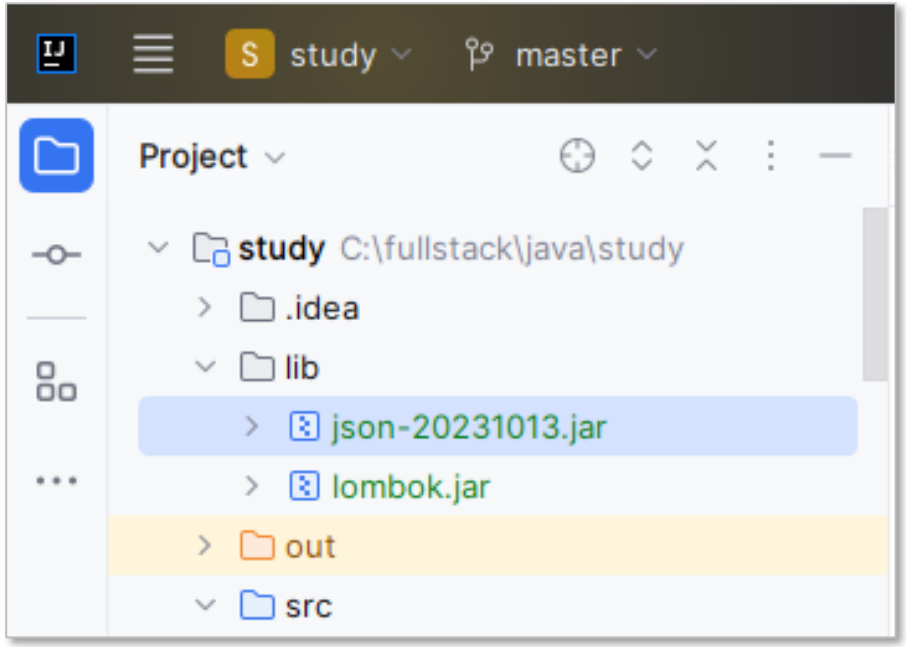
객체 표기	{
	"속성명": 속성값,
	"속성명": 속성값,
	…
	}
		속성명: 반드시 "로 감싸야 함
		속성값으로 가능한 것
		– "문자열", 숫자, true/false
		– 객체 { … }
		– 배열 […]

배열 표기	[항목, 항목, …]	항목으로 가능한 것 – "문자열", 숫자, true/false – 객체 { … } – 배열 […]

```
{
  "id": "winter",
  "name": "한겨울",
  "age": 25,
  "student": true,
  "tel": { "home": 02-123-1234", "mobile": "010-123-1234" },
  "skill": [ "java", "C", "C++" ]
}
```

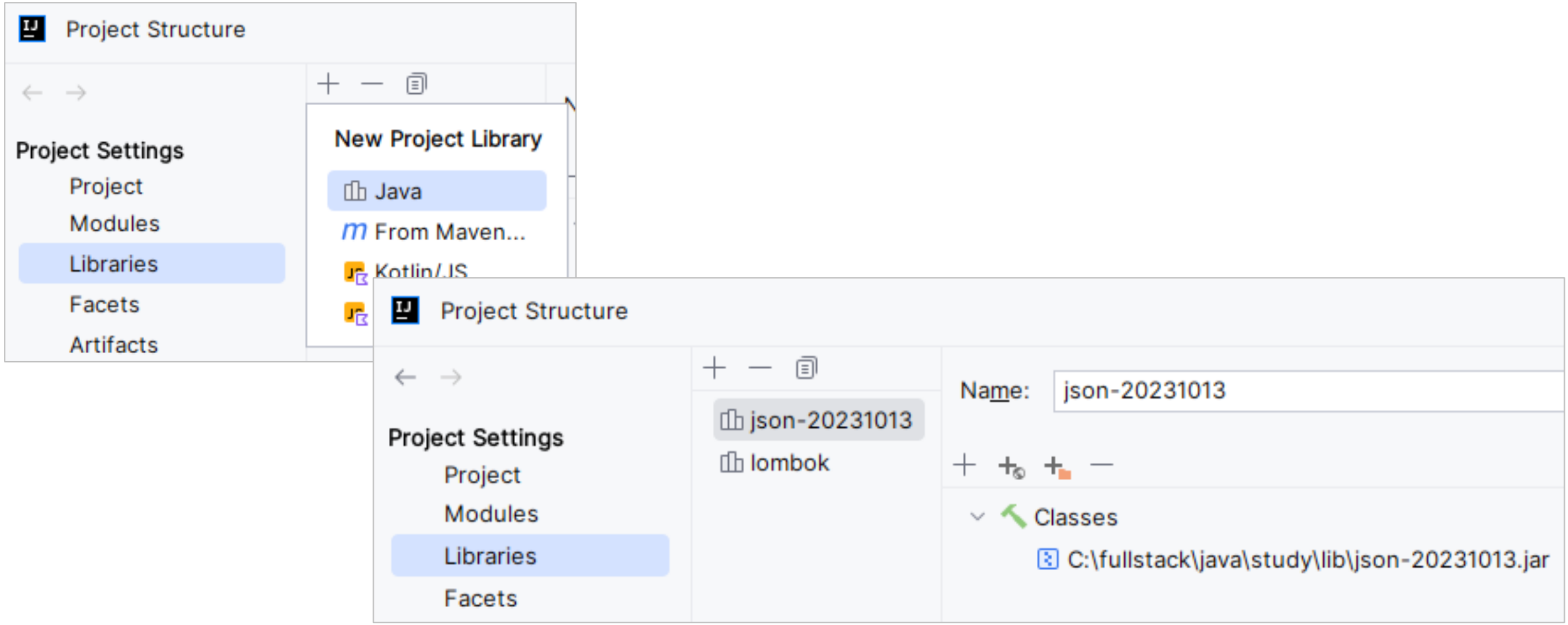
- **JSON-java 라이브러리**
  - https://github.com/stleary/JSON-java

json-20231013.jar → IntelliJ 프로젝트 lib에 복사



IntelliJ 프로젝트에 라이브러리 등록

- File > Project Structure...



JSON-java 라이브러리

클래스	용도
JSONObject	JSON 객체 표기를 생성하거나 파싱할 때 사용
JSONArray	JSON 배열 표기를 생성하거나 파싱할 때 사용

CreateJsonExample.java

```
package ch19.sec06;

import java.io.FileWriter;
import java.io.IOException;
import java.io.Writer;
import java.nio.charset.Charset;

import org.json.JSONArray;
import org.json.JSONObject;

public class CreateJsonExample {
  public static void main(String[] args) throws IOException {
    //JSON 객체 생성
    JSONObject root = new JSONObject();

    //속성 추가
    root.put("id", "winter");
    root.put("name", "한겨울");
    root.put("age", 25);
    root.put("student", true);
```

CreateJsonExample.java

```
    //객체 속성 추가
    JSONObject tel = new JSONObject();
    tel.put("home", "02-123-1234");
    tel.put("mobile", "010-123-1234");
    root.put("tel", tel);

    //배열 속성 추가
    JSONArray skill = new JSONArray();
    skill.put("java");
    skill.put("c");
    skill.put("c++");
    root.put("skill", skill);

    String json = root.toString();    //JSON 문자열 얻기
    System.out.println(json);

    //파일로 저장
    Writer writer = new FileWriter("C:/Temp/member.json", Charset.forName("UTF-8"));
    writer.write(json);
    writer.flush();
    writer.close();
  }
}
```

```
{"student":true,"skill":["java","c","c++"],"name":"한겨울","tel":{"mobile":"010-123-1234","home":"02-123-1234"},"id":"winter","age":25}
```

ParseJsonExample.java

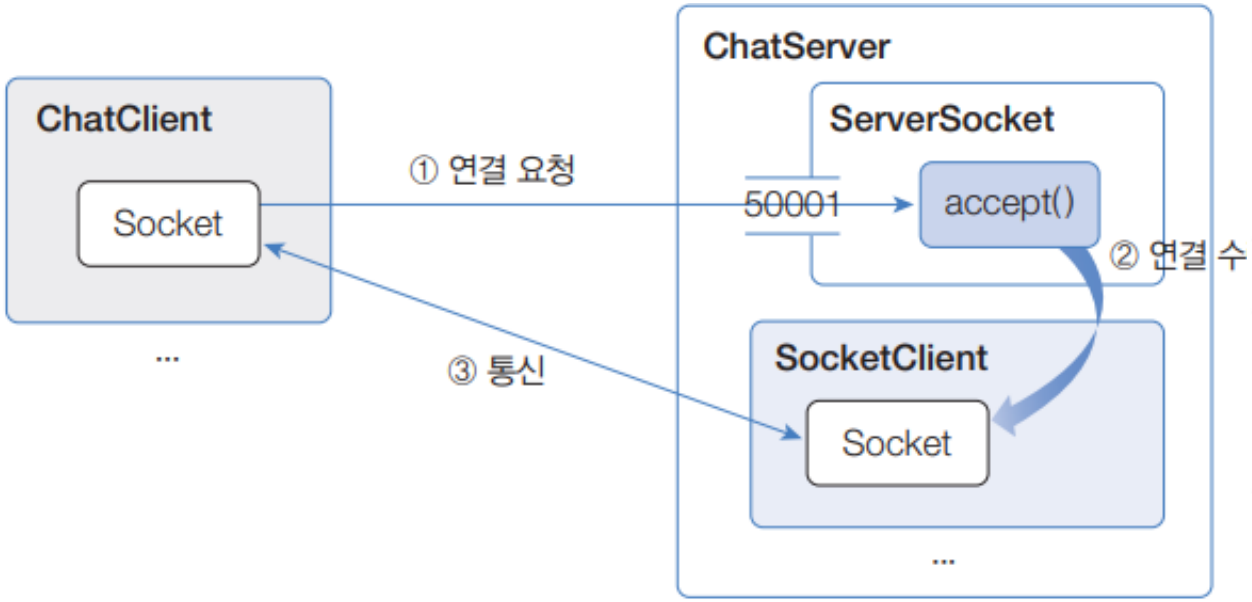
```
package ch19.sec06;

import java.io.BufferedReader;
import java.io.FileReader;
import java.io.IOException;
import java.nio.charset.Charset;

import org.json.JSONArray;
import org.json.JSONObject;

public class ParseJsonExample {
  public static void main(String[] args) throws IOException {
    //파일로부터 JSON 읽기
    BufferedReader br = new BufferedReader(
      new FileReader("C:/Temp/member.json", Charset.forName("UTF-8"))
    );
    String json  = br.readLine();
    br.close();
```

ParseJsonExample.java

```
    //JSON 파싱
    JSONObject root = new JSONObject(json);

    //속성 정보 읽기
    System.out.println("id: " + root.getString("id"));
    System.out.println("name: " + root.getString("id"));
    System.out.println("age: " + root.getInt("age"));
    System.out.println("student: " + root.getBoolean("student"));

    //객체 속성 정보 읽기
    JSONObject tel = root.getJSONObject("tel");
    System.out.println("home: " + tel.getString("home"));
    System.out.println("mobile: " + tel.getString("mobile"));

    //배열 속성 정보 읽기
    JSONArray skill = root.getJSONArray("skill");
    System.out.print("skill: ");
    for(int i=0; i<skill.length(); i++) {
      System.out.print(skill.get(i) + ", ");
    }
  }
}
```

채팅 서버와 클라이언트 구현

- TCP 네트워킹을 이용해서 채팅 서버와 클라이언트를 구현
- 채팅 서버: ChatServer는 채팅 서버 실행 클래스로 클라이언트의 연결 요청을 수락하고 통신용 SocketClient를 생성하는 역할
- 채팅 클라이언트: 단일 클래스 ChatClient는 채팅 서버로 연결을 요청하고, 연결된 후에는 제일 먼저 대화명을 보내며 다음 서버와 메시지를 주고 받음



클래스	용도
ChatServer	– 채팅 서버 실행 클래스 – ServerSocket을 생성하고 50001에 바인딩 – ChatClient 연결 수락 후 SocketClient 생성
SocketClient	– ChatClient와 1:1로 통신
ChatClient	– 채팅 클라이언트 실행 클래스 – ChatServer에 연결 요청 – SocketClient와 1:1로 통신