

15장- 네트워크 프로그래밍



Thread & Socket 데이터 통신



멀티 쓰레드

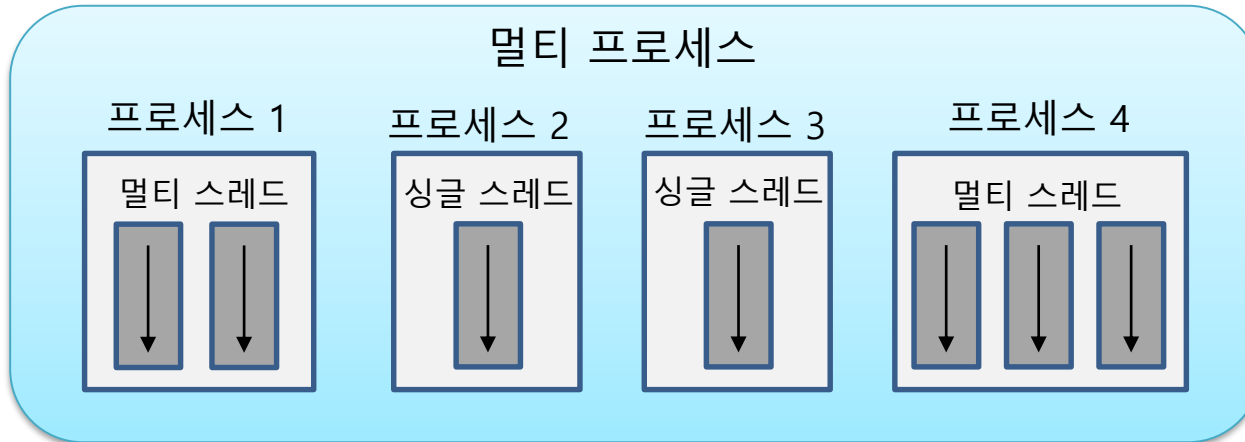
■ 프로세스와 쓰레드

프로세스(Process)

- 실행 중인 하나의 프로그램을 말한다.(하드디스크 -> 주기억장치)

멀티 태스킹(multi tasking)

- 두가지 이상의 작업을 동시에 처리하는 것.
- 멀티 프로세스 : 독립적으로 프로그램들을 실행하고 여러가지 작업처리.
- 멀티 스레드 : 한 개를 프로그램을 실행하고 내부적으로 여러 가지 작업 처리

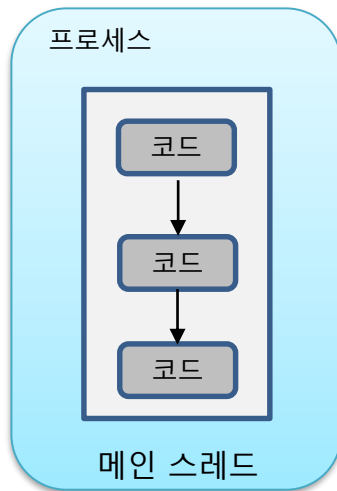


멀티 스레드

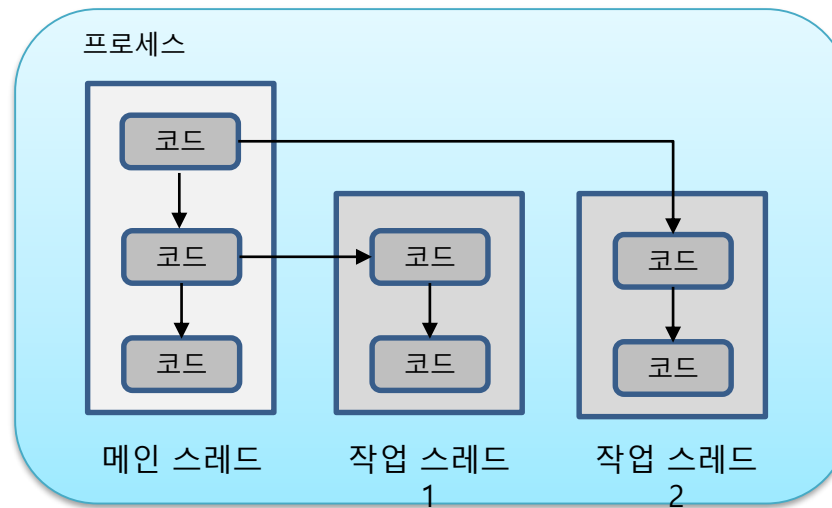
메인(main) 스레드

- 모든 자바 프로그램은 메인 스레드가 main() 메소드를 실행하면서 시작된다.
- main() 메소드의 첫 코드부터 아래로 순차적으로 실행한다.
- main() 메소드의 마지막 코드를 실행하거나, return문을 만나면 실행이 종료된다.

싱글 스레드 애플리케이션



멀티 스레드 애플리케이션



프로세스의 종료

- 싱글 스레드 : 메인 스레드가 종료되면 프로세스도 종료된다.
- 멀티 스레드 : 실행 중인 스레드가 하나라도 있다면, 프로세스는 종료되지 않는다.
(메인 스레드가 작업 스레드보다 먼저 종료되는 경우도 있다.)



Thread 클래스

작업 스레드 생성과 실행

- 자바에서는 작업 스레드도 객체로 생성되기 때문에 클래스가 필요하다.
- Java.lang.Thread 클래스를 직접 객체화하거나, Thread를 상속해서 하위 클래스를 만들어 생성

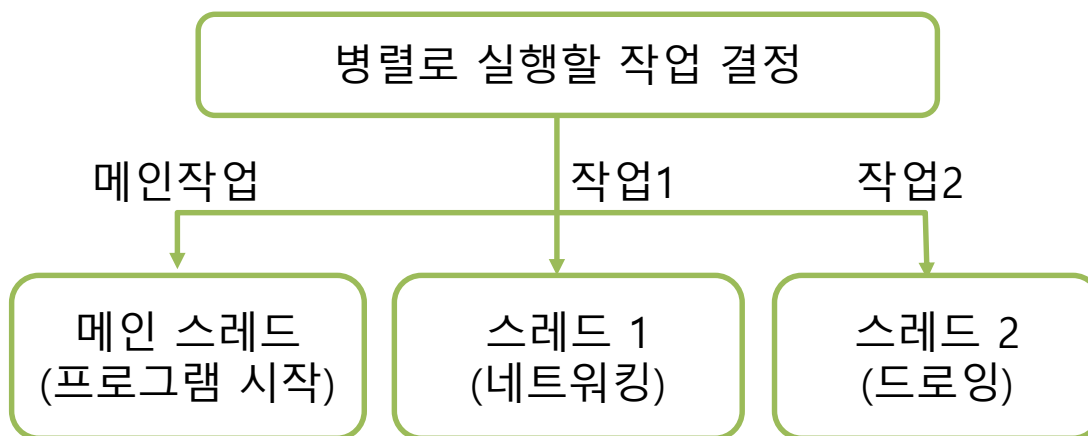
Class Thread

java.lang.Object
java.lang.Thread

All Implemented Interfaces:
Runnable

Direct Known Subclasses:
ForkJoinWorkerThread

```
public class Thread  
extends Object  
implements Runnable
```



Thread 클래스

방법 1-1) Thread 클래스로 부터 직접 생성

Runnable은 스레드가 작업을 실행할 때 사용하는 인터페이스이다.

Run() 메서드를 재정의해서 스레드가 실행할 코드를 가지고 있어야 함

```
class Task implements Runnable{  
  
    @Override  
    public void run(){  
        //스레드가 실행할 코드;  
    }  
}
```

```
Runnable task = new Task();
```

```
Thread thread = new Thread(task)
```

→ **thread.start()** 스레드 시작(실행)



Thread 클래스

방법 1-2) Runnable 익명 구현 객체를 매개값으로 사용

```
Thread thread = new Thread(new Runnable(){  
  
    @Override  
    public void run() {  
        //스레드가 실행할 코드  
    }  
});
```



Thread 클래스

방법 2) Thread 자식 클래스로 생성 (상속)

```
public class 스레드 클래스 extends Thread{  
    ...  
}
```

```
Thread thread = new 스레드클래스();
```

→ **thread.start()**

쓰레드 시작(실행)



Thread 이름

Thread 이름

스레드는 자신의 이름을 가지고 있다. 메인 스레드는 'main'이라는 이름을 가지고 있고, 직접 생성한 스레드는 자동으로 Thread-n 이라는 이름으로 설정된다.

다른 이름으로 설정하고 싶다면 Thread 클래스의 setName() 메소드로 변경한다.

```
main 실행  
Thread-0 실행  
Thread-1 실행  
chat-thread 실행  
chat-thread 실행
```



Thread 이름

Thread 이름

```
public class ThreadNameTest {  
  
    public static void main(String[] args) {  
        //현재 이 코드를 실행하는 스레드의 정보 얻기  
        Thread mainThread = Thread.currentThread();  
        System.out.println(mainThread.getName() + " 실행");  
  
        for(int i=0; i<2; i++) {  
            //Thread(작업 스레드) 객체 생성  
            Thread threadA = new Thread() {  
                //run() 메서드 재정의 함  
                @Override  
                public void run() {  
                    //getName() - threadA의 이름을 리턴  
                    System.out.println(getName() + " 실행");  
                }  
            };  
            threadA.start(); //실행 대기 상태임  
        }  
    }  
}
```



Thread 이름

Thread 이름

```
for(int i=0; i<2; i++) {  
    Thread chatThread = new Thread() {  
  
        @Override  
        public void run() {  
            System.out.println(getName() + " 실행");  
        }  
  
    };  
    chatThread.setName("chat-thread"); //쓰레드 이름 변경  
    chatThread.start();  
}
```



비프음과 문자 출력 동시 실행

메인 스레드만 이용한 경우

```
public class BeepPrintTest {  
    public static void main(String[] args) {  
        //메인 스레드만 실행  
        //"띵" 문자를 5번 출력하기 -> 1초 대기 간격  
        for(int i=0; i<5; i++) {  
            System.out.println("띵");  
            try {  
                Thread.sleep(1000); //1000ms -> 1s(1초)  
            } catch (InterruptedException e) {  
                e.printStackTrace();  
            }  
        }  
  
        //"띵" 소리를 5번 재생하기  
        Toolkit toolkit = Toolkit.getDefaultToolkit();  
        for(int i=0; i<5; i++) {  
            toolkit.beep();  
            try {  
                Thread.sleep(1000);  
            } catch (InterruptedException e) {  
                e.printStackTrace();  
            }  
        }  
    }  
}
```

Module java.desktop

Package java.awt

Class Toolkit

java.lang.Object

java.awt.Toolkit

```
public abstract class Toolkit  
extends Object
```



비프음과 문자 출력 동시 실행

비프음을 들려주는 작업스레드 정의

비프음을 발생시키면서 동시에 프린팅을 하고 싶다면 두 작업중 하나를 작업스레드에서 처리하도록 해야한다.

```
public class BeepTask implements Runnable{
    //Runnable 인터페이스를 구현한 BeepTask 클래스 생성
    //비프음을 재생하는 작업 정의
    @Override
    public void run() {
        Toolkit toolkit = Toolkit.getDefaultToolkit();
        for(int i=0; i<5; i++) {
            toolkit.beep();
            try {
                Thread.sleep(1000);
            } catch (InterruptedException e) {
                e.printStackTrace();
            }
        }
    }
}
```



비프음과 문자 출력 동시 실행

메인 스레드와 작업 스레드가 동시에 실행

```
public class BeepPrintTest2 {  
    public static void main(String[] args) {  
        //메인 스레드와 작업 스레드가 동시에 실행  
        Runnable beepTask = new BeepTask();  
        Thread thread = new Thread(beepTask);  
        thread.start(); //쓰레드 시작(실행)  
  
        for(int i=0; i<5; i++) {  
            System.out.println("땡");  
            try {  
                Thread.sleep(1000); //1000ms -> 1s(1초)  
            } catch (InterruptedException e) {  
                e.printStackTrace();  
            }  
        }  
    }  
}
```

EO EO EO EO EO



비프음과 문자 출력 동시 실행

익명 객체로 구현

```
public class BeepPrintTest3 {
    public static void main(String[] args) {
        //메인 스레드와 작업 스레드가 동시에 실행
        //익명 객체로 구현
        Thread thread = new Thread(new Runnable() {
            @Override
            public void run() {
                Toolkit toolkit = Toolkit.getDefaultToolkit();
                for(int i=0; i<5; i++) {
                    toolkit.beep();
                    try {
                        Thread.sleep(1000);
                    } catch (InterruptedException e) {
                        e.printStackTrace();
                    }
                }
            }
        });
        thread.start(); //스레드 시작

        for(int i=0; i<5; i++) {
            System.out.println("띵");
            try {
                Thread.sleep(1000); //1000ms -> 1s(1초)
            } catch (InterruptedException e) {
                e.printStackTrace();
            }
        }
    }
}
```



비프음과 문자 출력 동시 실행

람다식으로 구현

```
public class BeepPrintTest4 {  
    public static void main(String[] args) {  
        //메인 스레드와 작업 스레드가 동시에 실행  
        Thread thread = new Thread(()->{  
            Toolkit toolkit = Toolkit.getDefaultToolkit();  
            for(int i=0; i<5; i++) {  
                toolkit.beep();  
                try {  
                    Thread.sleep(1000);  
                } catch (InterruptedException e) {  
                    e.printStackTrace();  
                }  
            }  
        });  
        thread.start();  
  
        for(int i=0; i<5; i++) {  
            System.out.println("띵");  
            try {  
                Thread.sleep(1000); //1000ms -> 1s(1초)  
            } catch (InterruptedException e) {  
                e.printStackTrace();  
            }  
        }  
    }  
}
```



네트워크 기초

네트워크

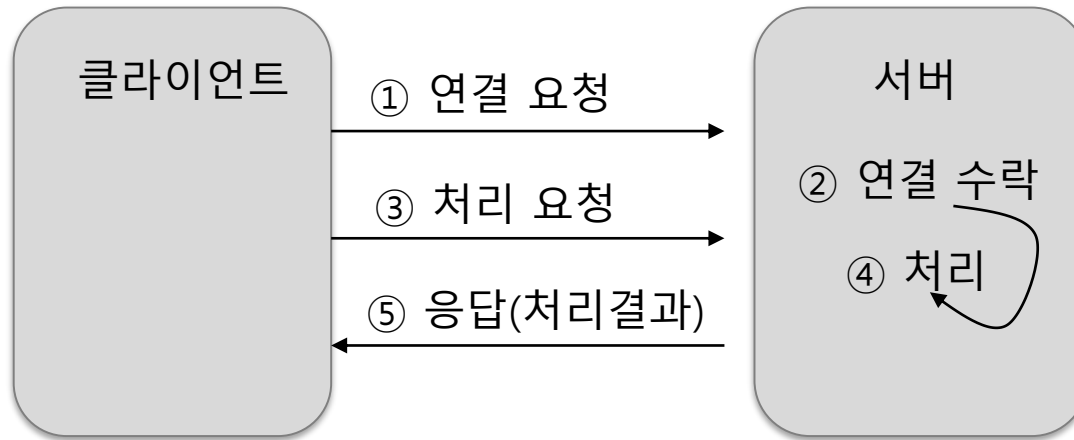
- 홈 네트워크 : 컴퓨터가 방마다 있고, 이들 컴퓨터를 유.무선 등의 통신회선으로 연결(LAN)
- 지역 네트워크 : 회사, 건물, 특정 영역에 존재하는 컴퓨터를 통신회선으로 연결한 것(MAN)
- 인터넷 : 지역네트워크를 통신 회선으로 연결한 것(WAN)

서버와 클라이언트

- **서버(Server)** : 서비스를 제공하는 프로그램
 - 웹서버, FTP서버, DBMS 서버, 메신저 서버
 - 클라이언트의 연결을 수락하고, 요청 내용을 처리한 후 응답을 보내는 역할
- **클라이언트(Client)** : 서비스를 받는 프로그램
 - 웹브라우저, FTP클라이언트, 메신저
 - 네트워크 데이터를 필요로 하는 모든 애플리케이션이 해당(모바일 App포함)



서버 / 클라이언트



- ▶ 클라이언트/서버(C/S: client/server) : 한 개의 서버와 다수의 클라이언트로 구성
- ▶ P2P(Peer to Peer) : 두 개의 프로그램이 서버인 동시에 클라이언트 역할을 함
먼저 접속을 시동한 컴퓨터가 클라이언트가 된다. (1:1 채팅과, 파일 공유 프로그램)



IP 주소와 포트(Port)

IP 주소

- IP(Internet Protocol) 주소 : 컴퓨터의 고유한 주소 - IPv4
- xxx.xxx.xxx.xxx(xxx는 0~255 사이의 정수)
- 네트워크 어댑터(Lan 카드) 마다 할당 - 유선/무선 랜카드

명령 프롬프트(cmd)

C:W>ipconfig 입력

```
C:\Users\김기용>ipconfig
```

```
Windows IP 구성
```

```
이더넷 어댑터 이더넷:
```

```
미디어 상태 . . . . . : 미디어 연결 끊김
```

```
연결별 DNS 접미사. . . . . :
```

```
무선 LAN 어댑터 Wi-Fi:
```

```
연결별 DNS 접미사. . . . . :
```

```
링크-로컬 IPv6 주소 . . . . . : fe80::fdb4:956c:171d:19c7%14
```

```
IPv4 주소 . . . . . : 192.168.0.6
```

```
서브넷 마스크 . . . . . : 255.255.255.0
```

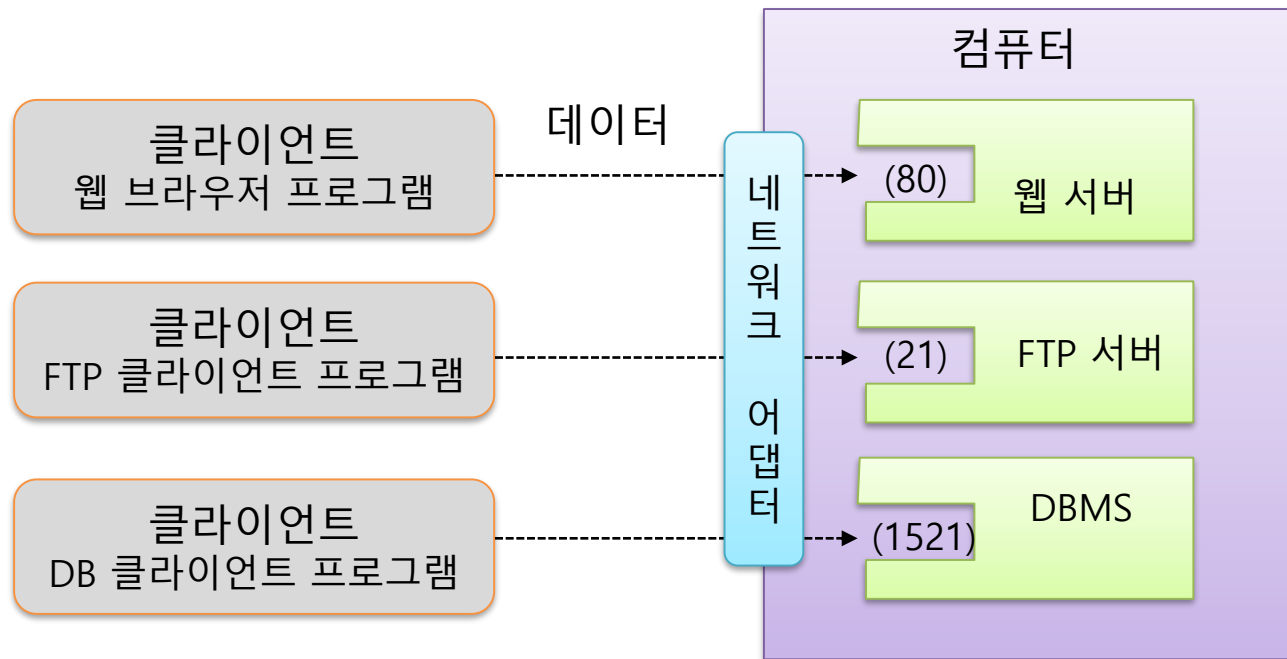
```
기본 게이트웨이 . . . . . : 192.168.0.1
```



IP 주소와 포트(Port)

포트(Port)

- 같은 컴퓨터 내에서 프로그램을 식별하는 번호
- 클라이언트는 서버 연결 요청시 IP 주소와 Port를 같이 제공
- 포트번호의 전체 범위 : 0 ~ 65535 범위의 값을 가짐



IP 주소와 포트(Port)

- Port는 운영체제가 관리하는 서버 프로그램의 연결 번호이다. 서버는 시작할 때 Port 번호에 바인딩한다. 예를 들어 서버는 80번, DBMS(Oracle)는 1521번으로 바인딩 할 수 있음.
- 클라이언트도 서버에서 보낸 정보를 받기 위해서는 Port 번호가 필요한데, 서버와 같이 고정적인 Port 번호에 바인딩하는 것이 아니라 운영체제가 자동으로 부여하는 번호를 사용한다.

클래스	범위	용 도
잘 알려진 포트번호 (Well Know Port Numbers)	0~1023	국제인터넷 주소관리기구(CANN)가 특정 애플리케이션용으로 미리 예약한 Port
예약된 포트번호 (Registered Port Numbers)	1024~49151	회사에서 등록해서 사용할 수 있는 Port
동적인 또는 개인 포트번호 (Dynamic Or Private PortNumbers)	49152~65535	운영체제가 부여하는 동적 Port 또는 개인 목적으로 사용할 수 있는 Port



IP 주소와 포트(Port)

IP 주소 얻어 오기

- **Java.net.InetAddress**
- IP 주소를 표현한 클래스
- 로컬 컴퓨터의 IP 주소 뿐만아니라 도메인 이름을 DNS에서 검색한 후 IP 주소를 가져 오는 기능 제공 (예. www.naver.com -> IP 주소)

1. 로컬 컴퓨터에서 얻기

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

2. 도메인 이름으로 얻기

```
InetAddress ia = InetAddress.getByName(String host)
```

```
InetAddress[ ] iaArr = InetAddress.getAllByName(String host)
```



IP 주소와 포트(Port)

IP 주소 얻어 오기

```
package inetaddress;
import java.net.InetAddress;
import java.net.UnknownHostException;

public class InetAddressEx {

    public static void main(String[] args) {

        try {
            //내 컴퓨터
            InetAddress local = InetAddress.getLocalHost();
            System.out.println("내 컴퓨터 IP 주소 : " + local.getHostAddress());

            //서버 컴퓨터
            //InetAddress server = InetAddress.getByName("www.naver.com");
            //System.out.println(server);
            InetAddress[] servers = InetAddress.getAllByName("www.naver.com");
            for(InetAddress remote : servers)
                System.out.println(remote);
        } catch (UnknownHostException e) {
            e.printStackTrace();
        }
    }
}
```

내 컴퓨터 IP 주소 : 192.168.0.6

네이버 컴퓨터 IP 주소 : www.naver.com/223.130.195.200

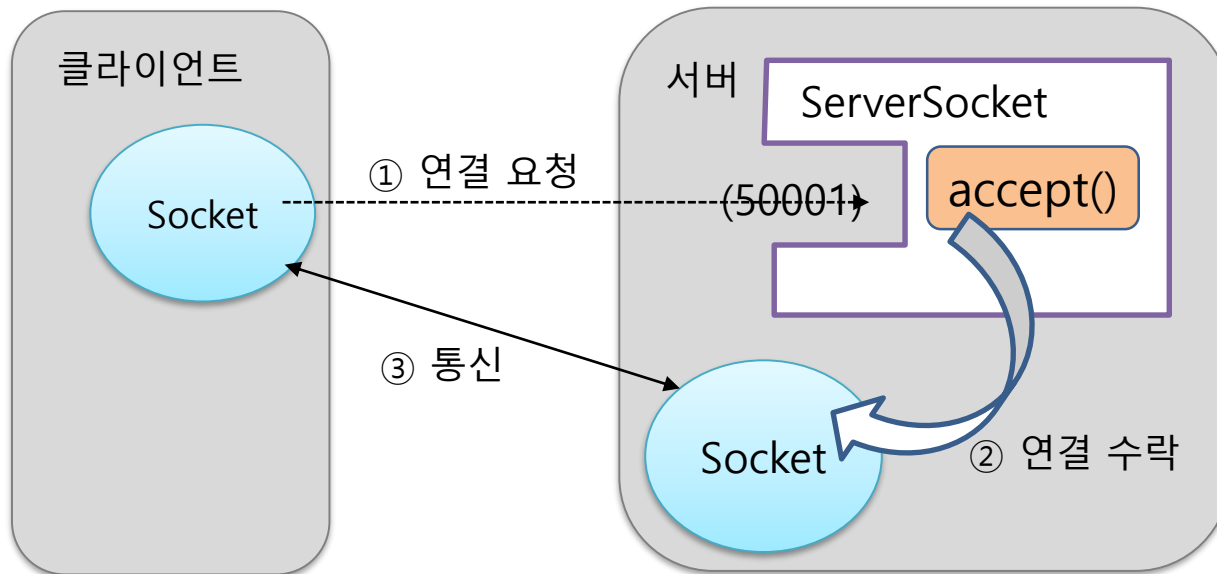
네이버 컴퓨터 IP 주소 : www.naver.com/223.130.195.95



TCP 네트워크

TCP(Transmission Control Protocol)

- 연결 지향적 프로토콜 : 클라이언트와 서버가 연결된 상태에서 데이터를 주고 받는 프로토콜이다.
- 데이터를 정확하고 안정적으로 전달 - 데이터를 순차적으로 보내고 받을 때도 순차적으로 받음
- **Java.net API – ServerSocket 클래스, Socket 클래스**



TCP 네트워킹

```
Problems @ Javadoc Declaration Console x
ServerSample [Java Application] C:\Users\wkiyon\p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.f
=====
서버를 종료하려면 q 또는 Q를 입력하고 Enter를 누르세요.
=====
[서버] 시작됨

[서버] 연결 요청을 기다림

[서버] 127.0.0.1의 연결 요청을 수락함
[서버] 127.0.0.1의 연결을 끊음

Console x
<terminated> ClientExample [Java Application] C:\Users\wkiyon\p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.f
[클라이언트] 연결 성공
[클라이언트] 연결 끊음
```



ServerSocket 생성과 연결 수락

ServerSocket 생성과 연결 수락 – 서버 프로그램

1. ServerSocket 생성과 포트 바인딩

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

2. 연결 수락

```
try{  
    Socket socket = serverSocket.accept();  
}catch(Exception e){ }
```

3. 연결된 클라이언트 IP 주소 얻기

```
InetSocketAddress socketAddress =  
    (InetSocketAddress)socket.getRemoteSocketAddress();
```

2. 연결 끊기

```
try{  
    serverSocket.close();  
}catch(Exception e){ }
```



ServerSocket 연결 수락

연결 수락 - 서버 프로그램

```
public class ServerSample {
    //서버 소켓 객체 선언
    private static ServerSocket serverSocket;
    //main 스레드 - 키보드 입력 : 서버 종료하는 작업
    //작업 스레드 - 클라이언트 요청 받아서 수락하는 작업
    public static void main(String[] args) {
        System.out.println("=====");
        System.out.println("서버를 종료하려면 q 또는 Q를 입력하고 Enter를 누르세요.");
        System.out.println("=====");

        //TCP 서버 시작
        startServer();

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



ServerSocket 연결 수락

```
public static void startServer() {  
    //작업 스레드 정의  
    Thread thread = new Thread() {  
        @Override  
        public void run() {  
            //ServerSocket 생성 및 Port 바인딩  
            try {  
                serverSocket = new ServerSocket(50001);  
                System.out.println("[서버] 시작됨"); //실행 : BindException 확인  
  
                while(true) { //여러 클라이언트의 연결 요청 수락을 위해 필요함  
                    System.out.println("\n[서버] 연결 요청을 기다림\n");  
                    //연결 수락  
                    Socket socket = serverSocket.accept();  
  
                    InetAddress isa =  
                        (InetAddress)socket.getRemoteSocketAddress();  
                    //String clientIp = isa.getHostName(); //컴퓨터 이름이 나올수 있음  
                    String clientIp = isa.getHostString();  
                    System.out.println("[서버] " + clientIp + "의 연결 요청을 수락함");  
                    //웹 브라우저에 ip주소:50001 입력 -> 콘솔에 IP 주소 출력됨
```

[서버] 연결 요청을 기다림

[서버] 192.168.35.183의 연결 요청을 수락함

[서버] 192.168.35.183의 연결을 끊음

[서버] 연결 요청을 기다림



ServerSocket 연결 수락

```
        //연결 끊기
        socket.close();
        System.out.println("[서버] " + clientIp + "의 연결을 끊음");
    }
} catch (IOException e) {
    //System.out.println("[서버] " + e.getMessage());
    System.out.println("[서버] " + e.toString()); //한 번 더 실행
}
}
};
thread.start();
}

public static void stopServer() {
    try {
        serverSocket.close();
    } catch (IOException e) {
        e.printStackTrace();
    }
}
```

* q를 입력하면 SocketException 발생함

```
q
[서버] 종료됨
[서버] Socket closed
java.net.SocketException: Socket closed
    at java.base/sun.nio.ch.NioSocketImpl.endAccept(NioSocketImpl.java:689)
    at java.base/sun.nio.ch.NioSocketImpl.accept(NioSocketImpl.java:762)
    at java.base/java.net.ServerSocket.implAccept(ServerSocket.java:675)
    at java.base/java.net.ServerSocket.platformImplAccept(ServerSocket.java:641)
    at java.base/java.net.ServerSocket.implAccept(ServerSocket.java:617)
    at java.base/java.net.ServerSocket.implAccept(ServerSocket.java:574)
    at java.base/java.net.ServerSocket.accept(ServerSocket.java:532)
    at server.ServerSample$1.run(ServerSample.java:47)
```



TCP 클라이언트

Socket 생성과 연결 요청 – 소켓 클라이언트

1. Socket 생성과 포트 바인딩 -> 서버에 연결 요청

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

2. 연결 끊기

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



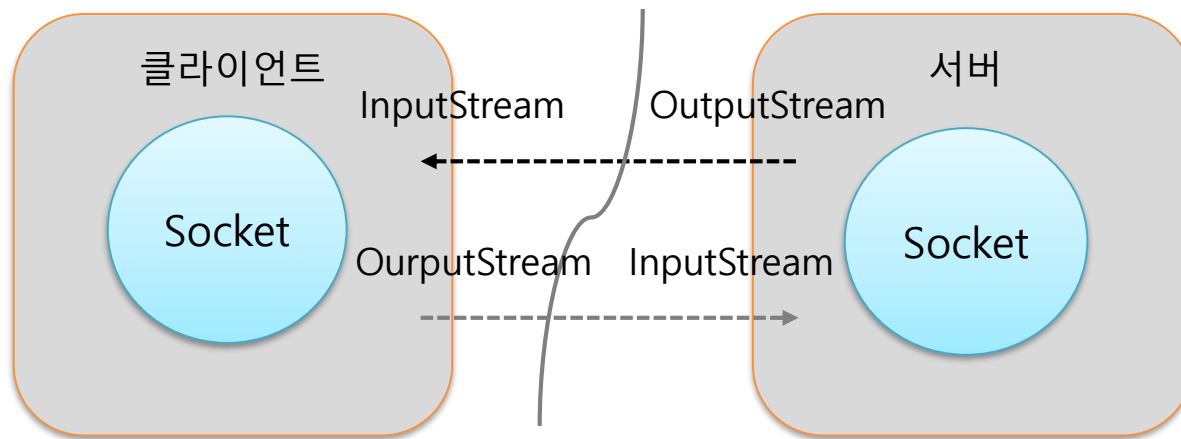
Socket 연결 요청

```
public class ClientExample {  
  
    public static void main(String[] args) {  
        try {  
            //Socket 생성과 동시에 연결 요청  
            Socket socket = new Socket("localhost", 50001);  
  
            System.out.println("[클라이언트] 연결 성공");  
  
            socket.close();  
            System.out.println("[클라이언트] 연결 끊음");  
        } catch (UnknownHostException e) {  
            //IP 또는 도메인 표기 방법이 잘못된 경우  
            System.out.println("UnknownHostException" + e.toString());  
        } catch (IOException e) {  
            //IP 또는 포트번호가 잘못된 경우  
            System.out.println("IOException" + e.toString());  
        }  
    }  
}
```



Socket 데이터 보내고 받기

Socket 데이터 통신



//입력스트림 얻기

InputStream is = socket.getInputStream()

//출력스트림 얻기

OutputStream os = socket.getOutputStream()



Socket 데이터 보내고 받기

Socket 데이터 통신

➤ 데이터 보내기(쓰기)

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

➤ 데이터 받기(읽기)

```
byte[ ] bytes = new byte[100];  
InputStream inputStream = socket.getInputStream()  
int readByteCount = inputStream.read(bytes);  
String data = new String(bytes, 0, readByteCount, "UTF-8")
```



Socket 데이터 보내고 받기

[클라이언트] 연결 성공
[클라이언트] 데이터 보냄: 오늘도 즐거운 하루 되세요~
[클라이언트] 데이터 받음: 오늘도 즐거운 하루 되세요~
[클라이언트] 연결 끊음

```
=====
서버를 종료하려면 q 또는 Q를 입력하고 Enter를 누르세요.
=====
[서버] 시작됨

[서버] 연결 요청을 기다림

[서버] 127.0.0.1의 연결 요청을 수락함
[서버] 받은 데이터를 다시 보냄: 오늘도 즐거운 하루 되세요~
[서버] 127.0.0.1의 연결을 끊음
```



ServerSocket 데이터 보내고 받기

Socket 데이터 통신 – 서버 프로그램 (EcoServer.java)

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

    InetSocketAddress isa = (InetSocketAddress)socket.getRemoteSocketAddress();
    //String clientIp = isa.getHostName(); //컴퓨터 이름이 나올수 있음
    String clientIp = isa.getHostString();
    System.out.println("[서버] " + clientIp + "의 연결 요청을 수락함");
    //-----
    //데이터 받기
    InputStream is = socket.getInputStream();
    byte[] bytes = new byte[1024]; //1KB
    int readByteCount = is.read(bytes); //읽은 바이트 수
    String message = new String(bytes, 0, readByteCount, "utf-8"); //디코딩 문자셋

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



ServerSocket 데이터 보내고 받기

보조 스트림 사용하기 – DataInputStream, DataOutputStream

```
//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 데이터 보내고 받기

Socket 데이터 통신 – 클라이언트 프로그램(EcoClient.java)

```
try {
    Socket socket = new Socket("localhost", 50001);

    System.out.println("[클라이언트] 연결 성공");
    //-----
    //데이터 보내기
    String sendMessage = "오늘도 즐거운 하루 되세요~";
    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); //읽은 바이트 수
    //디코딩 문자셋
    String receiveMessage = new String(bytes, 0, readByteCount, "utf-8");
    System.out.println("[클라이언트] 데이터 받음: " + receiveMessage);
    //-----
}
```



Socket 데이터 보내고 받기

보조 스트림 사용하기 – DataInputStream, DataOutputStream

```
//데이터 보내기
String sendMessage = "오늘도 즐거운 하루 되세요~";
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);
//-----
```



JSON 데이터 형식

● JSON이란?

네트워크로 전달하는 데이터가 복잡할 수록 구조화된 형식이 필요하다.

네트워크에서 데이터 송신과 수신에 많이 사용되는 데이터 형식은 **JSON**과 **XML**이다.

JSON은 **JavaScript Object Notation**의 약자로 자바스크립트의 객체 형식을 기반으로 만들어졌고, 표기법은 아래와 같다.

```
{
  "id": "sky123",
  "name": "이하늘",
  "age": 28,
  "tel": {"mobile": "010-1234-5678", "home": "02-111-2222"},
  "student": true,
  "skill": ["java", "c", "c++"]
}
```



JSON 데이터 형식

● JSON 표기법

객체 표기	<pre>{ "속성명" : 속성값 "속성명" : 속성값 ... }</pre>	<p>속성명: 반드시 쌍따옴표("")로 감싸야함 속성값으로 가능한 것</p> <ul style="list-style-type: none">- 문자열, 숫자, true/false- 객체{ }- 배열[...]
배열 표기	<pre>[요소1, 요소2]</pre>	<p>요소로 가능한 것</p> <ul style="list-style-type: none">- 문자열, 숫자, true/false- 객체{ }- 배열[...]




JSON 데이터 형식

JSON 라이브러리

JSON을 만들고 해석할 수 있는 라이브러리이다.

Maven Repository(메이븐 리포지터리) > json검색 > json in java > 20230628버전




JSON In Java » 20230618

JSON is a light-weight, language independent, data interchange format. See the files in this package implement JSON encoders/decoders in Java. It also includes examples of how to use JSON between JSON and XML, HTTP headers, Cookies, and CDL. This is a reference implementation of a large number of JSON packages in Java. Perhaps someday the Java community will have a standard. Until then, choose carefully.

License	Public
Categories	JSON Libraries
Tags	format bundle json serialization osgi
HomePage	https://github.com/douglascrockford/JSON-java
Date	Jun 18, 2023
Files	pom (6 KB) bundle (71 KB) View All
Repositories	Central Appodeal Knetminer Mulesoft Talend Public
Ranking	#90 in MvnRepository (See Top Artifacts) #5 in JSON Libraries

bundle 클릭

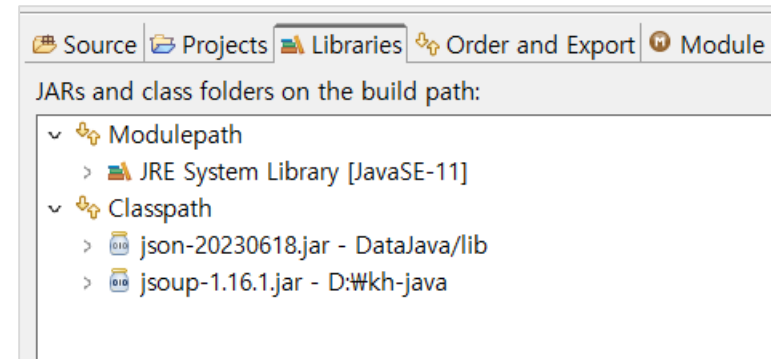
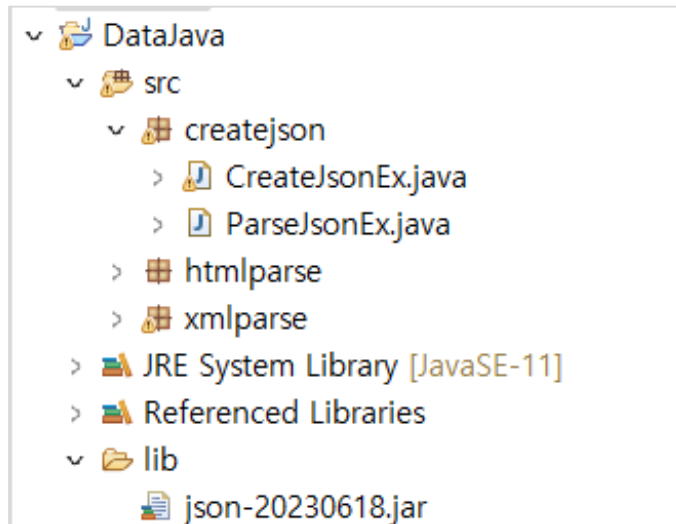
 json-20230618.jar



JSON 데이터 형식

이클립스에 세팅하기

프로젝트 > 우클릭 > 폴더 > lib 생성 > json jar 파일 복사 > 우클릭 > add to Build Path



JSON 데이터 형식

JSON 관련 주요 클래스

클래스	용 도
JSONObject	JSON으로 객체를 생성하거나 파싱할 때 사용
JSONArray	JSON으로 배열을 생성하거나 파싱할 때 사용



JSON 데이터 실습 예제

JSON 만들기 – 회원 정보

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

public class CreateJson {
    public static void main(String[] args) {
        //json 객체 생성
        JSONObject root = new JSONObject();

        System.out.println(root); //빈 json 생성

        //속성(객체) 추가
        root.put("id", "sky123");
        root.put("name", "이하늘");
        root.put("age", 28);
        root.put("student", true);

        //전화번호 객체 속성 추가
        JSONObject tel = new JSONObject();
        tel.put("home", "02-111-2222");
        tel.put("mobile", "010-1234-5678");
        root.put("tel", tel);
    }
}
```



JSON 데이터 실습 예제

```
//배열 속성 추가
JSONArray skill = new JSONArray();
skill.put("Java");
skill.put("C");
skill.put("C++");
root.put("skill", skill);

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

//json 데이터를 파일로 저장
try(Writer writer = new FileWriter("member.json",
    Charset.forName("utf-8"))){
    writer.write(json);
    writer.flush();
} catch (IOException e) {
    e.printStackTrace();
}
}
```

```
{"student":true,"skill":["java","c","c++"],"name":"이하늘","tel":{"mobile":"010-1234-5678","home":"02-111-2222"},"id":"sky123","age":28}
```



JSON 데이터 실습 예제

JSON 파싱(해석)하기 – 문자열 출력

```
public class ParseJson {  
    public static void main(String[] args) {  
        //파일의 내용을 읽어서 출력  
        try(BufferedReader br = new BufferedReader(new FileReader("member.json"))){  
            String json = br.readLine();  
  
            //파싱  
            JSONObject root = new JSONObject(json);  
  
            //속성 정보  
            System.out.println(root.getString("id"));  
            System.out.println(root.getString("name"));  
            System.out.println(root.getInt("age"));  
            System.out.println(root.getBoolean("student"));  
        }  
    }  
}
```



JSON 데이터 실습 예제

JSON 파싱(해석)하기 - 문자열 출력

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

//배열의 속성 정보
JSONArray skill = root.getJSONArray("skill");
System.out.print("skill: ");
for(int i = 0; i < skill.length(); i++) {
    System.out.println(skill.get(i) + ", ");
}
} catch (IOException e) {
    e.printStackTrace();
}
}
```

```
id: sky123
name: 이하늘
age: 28
student: true
home: 02-111-2222
mobile: 010-1234-5678
skill: java, c, c++,
```

