

20장. 네트워크 통신 프로그래밍

학습 목표

- TCP/IP 프로토콜에 대하여
- TCP/IP 통신 프로그램의 작성 방법

01. TCP/IP 프로토콜에 대하여

TCP/IP 프로토콜

- 인터넷에서 사용되는 프로토콜(protocol, 통신 규칙)
- TCP 프로토콜과 IP 프로토콜을 함께 부르는 이름

보안



주소

네트워크에 연결된 각각의 컴퓨터에
IP 주소(IP address)를 붙여서 관리하는 규칙

01. TCP/IP 프로토콜에 대하여

TCP/IP 프로토콜

- 컴퓨터의 IP 주소 확인하는 방법

```
C:\>ipconfig

Windows 2000 IP Configuration

Ethernet adapter 로컬 영역 연결:

    Connection-specific DNS Suffix  . : 
    IP Address. . . . . : 219.153.12.14
    Subnet Mask . . . . . : 255.255.255.0
    Default Gateway . . . . . : 219.153.12.1

C:\>
```

통신 대상 프로그램을 찾기 위해서는
IP 주소만으로는 부족

01. TCP/IP 프로토콜에 대하여

TCP/IP 프로토콜 16 bit

- 포트(port) : 네트워크를 통해 데이터를 주고 받을 때 사용되는 가상의 출구

- 0 ~ 65535 범위의 정수 번호로 식별됨

0 ~ 1023 : well known port

80, 21, 25, 110 등

보통 서비스

mysql 3306 사용
port 번호

01. TCP/IP 프로토콜에 대하여

TCP/IP 프로토콜

- 컴퓨터에서 사용 중인 포트 번호 확인하는 방법

```
C:\W>netstat -na

Active Connections

Proto Local Address          Foreign Address         State
TCP   0.0.0.0:135             0.0.0.0:0               LISTENING
TCP   0.0.0.0:445             0.0.0.0:0               LISTENING
UDP   0.0.0.0:445             *:*
```

사용 중인 포트 번호

주의: 0 ~ 1023까지는 시스템이 사용하는 포트번호

01. TCP/IP 프로토콜에 대하여

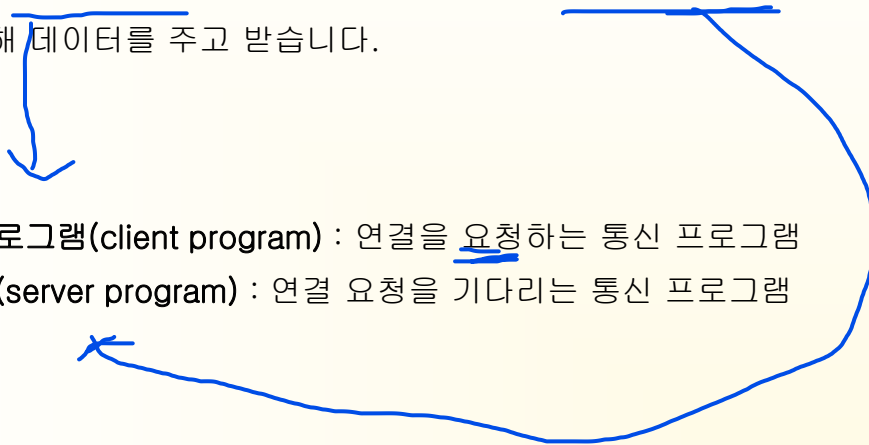
네트워크 통신 프로그램

- 통신 프로그램의 작성 방법

- 1) 한 프로그램은 연결을 요청하고 다른 프로그램은 그 요청을 받아서 연결을 맺습니다.
- 2) 그 연결을 통해 데이터를 주고 받습니다.

- 용어 설명

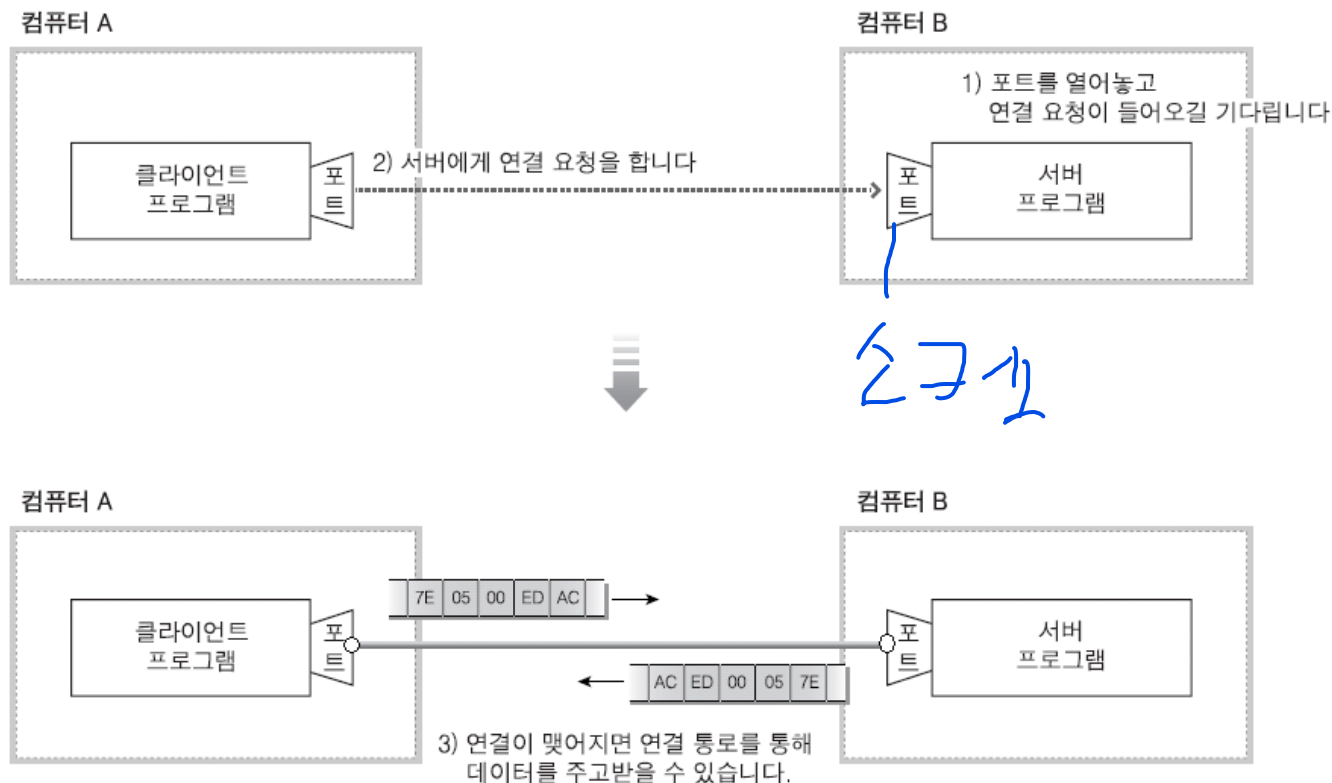
- 클라이언트 프로그램(client program) : 연결을 요청하는 통신 프로그램
- 서버 프로그램(server program) : 연결 요청을 기다리는 통신 프로그램



01. TCP/IP 프로토콜에 대하여

네트워크 통신 프로그램

- 클라이언트 프로그램과 서버 프로그램의 통신 과정



02. TCP/IP 통신 프로그램의 작성 방법

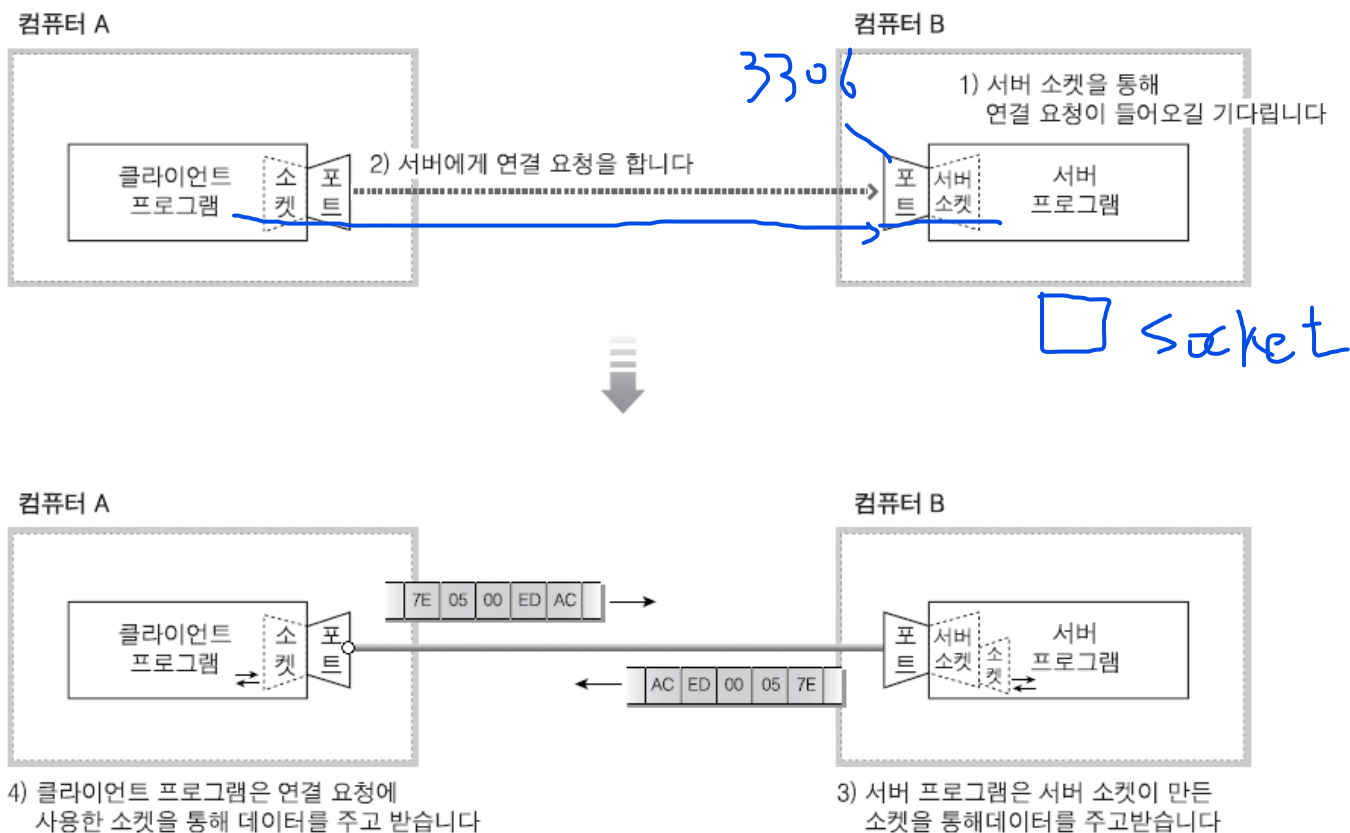
소켓에 대하여

- 소켓(socket) : 프로그램 내에서 보았을 때의 데이터 통신 출입구
 - 서버 소켓, 클라이언트 소켓 두 종류
- 서버 소켓
 - 서버 프로그램에서만 사용되는 소켓
 - 연결 요청을 기다리다가, 연결 요청이 오면 연결을 맺고 또 다른 소켓을 생성
- 클라이언트 소켓
 - 클라이언트 프로그램과 서버 프로그램에서 모두 사용되는 소켓
 - 실제 데이터 전송에 사용되는 것은 이 소켓임
 - 서버 프로그램에서는 서버 소켓에 의해 생성됨
 - 클라이언트 프로그램에서는 직접 생성해야 함

02. TCP/IP 통신 프로그램의 작성 방법

소켓을 이용한 통신

- 소켓을 이용한 클라이언트 프로그램과 서버 프로그램의 통신 과정



02. TCP/IP 통신 프로그램의 작성 방법

소켓을 이용한 통신 : 서버 프로그램

- 서버 소켓을 생성하고 사용하는 방법

1) ServerSocket 객체를 생성합니다.

```
ServerSocket serverSocket = new ServerSocket(9000);
```



포트 번호

02. TCP/IP 통신 프로그램의 작성 방법


소켓을 이용한 통신 : 서버 프로그램

- 서버 소켓을 생성하고 사용하는 방법

2) ServerSocket 객체에 대해 accept 메소드를 호출합니다.



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



서버 소켓으로 연결 요청이 들어오면 연결을 맺고,
클라이언트 소켓을 생성해서 리턴하는 메소드

02. TCP/IP 통신 프로그램의 작성 방법

소켓을 이용한 통신 : 클라이언트/서버 프로그램

- 클라이언트 소켓을 생성하고 사용하는 방법

1) Socket 객체를 생성합니다.

```
Socket socket = new Socket("219.153.21.14", 9000);
```

↑
서버 프로그램이 있는
컴퓨터의 IP 주소

↑
서버 프로그램이
열어 놓은 포트 번호

내 IP 주소

2프백 주소

내 컴퓨터 내 가상

127.0.0.1

02. TCP/IP 통신 프로그램의 작성 방법

소켓을 이용한 통신 : 클라이언트/서버 프로그램

- 클라이언트 소켓을 생성하고 사용하는 방법

2) 데이터 송수신에 사용할 입력 스트림 객체와 출력 스트림 객체를 얻습니다.

```
InputStream in = socket.getInputStream();
```

수신

데이터 수신에 사용할
입력 스트림 객체를 리턴하는 메소드

바이트
스트림
방식

```
OutputStream out = socket.getOutputStream();
```

송신

데이터 송신에 사용할
출력 스트림 객체를 리턴하는 메소드

02. TCP/IP 통신 프로그램의 작성 방법

소켓을 이용한 통신 : 클라이언트/서버 프로그램

- 클라이언트 소켓을 생성하고 사용하는 방법

3) write 메소드와 read 메소드를 호출하여 데이터 송신 또는 수신합니다.

```
out.write(data);
```

↑
파라미터로 넘겨준
데이터를 송신합니다

```
int data = in.read();
```

↑
수신된 데이터를
읽어서 리턴합니다

02. TCP/IP 통신 프로그램의 작성 방법

소켓을 이용한 통신 : 클라이언트/서버 프로그램

- 클라이언트 소켓을 생성하고 사용하는 방법
 - 4) 클라이언트 소켓을 닫습니다.

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

↑
소켓을 닫는 메소드

02. TCP/IP 통신 프로그램의 작성 방법

소켓을 이용한 통신 : 서버 프로그램

- 서버 소켓도 모두 사용하고 난 후에는 닫아야 합니다.

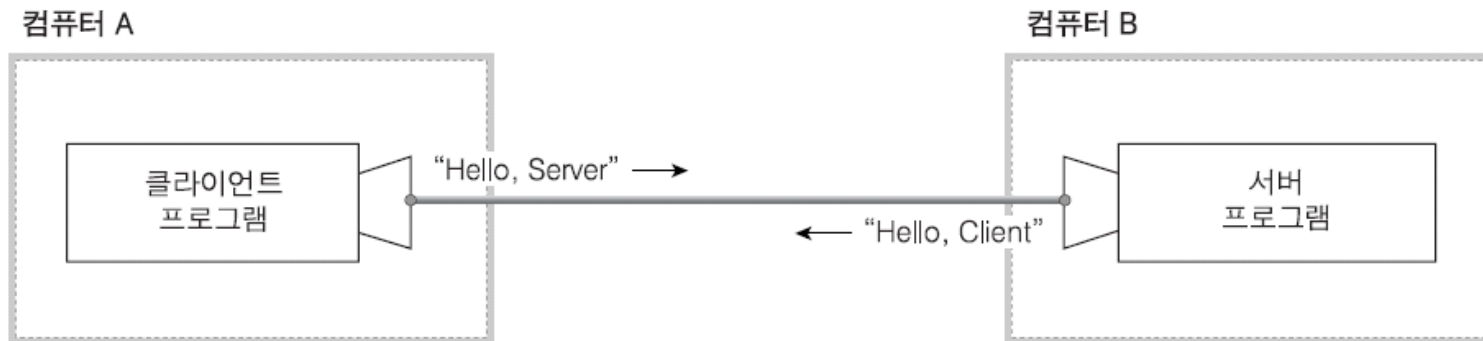
```
serverSocket.close();
```

↑
서버 소켓을 닫는 메소드

02. TCP/IP 통신 프로그램의 작성 방법

클라이언트/서버 프로그램 작성

- 지금부터 작성할 예제가 하는 일



02. TCP/IP 통신 프로그램의 작성 방법

클라이언트/서버 프로그램 작성

[예제 20-1] TCP/IP로 통신하는 클라이언트 프로그램과 서버 프로그램 (1)

클라이언트 프로그램

```

1 import java.io.*;
2 import java.net.*;
3 class ClientExample1 {
4     public static void main(String[] args) {
5         Socket socket = null;
6         try {
7             socket = new Socket("###.###.###.###", 9000); --- 소켓을 생성합니다
8             InputStream in = socket.getInputStream();
9             OutputStream out = socket.getOutputStream();
10            String str = "Hello, Server";
11            out.write(str.getBytes()); } 데이터 송신합니다
12            byte arr[] = new byte[100];
13            in.read(arr); } 수신된 데이터를 출력합니다
14            System.out.println(new String(arr));
15        }
16        catch (Exception e) {
17            System.out.println(e.getMessage());
18        }
19        finally {
20            try {
21                socket.close(); ----- 소켓을 닫습니다
22            }
23            catch (Exception e) {
24            }
25        }
26    }
27 }

```

서버 프로그램

```

1 import java.io.*;
2 import java.net.*;
3 class ServerExample1 {
4     public static void main(String[] args) {
5         ServerSocket serverSocket = null;
6         Socket socket = null;
7         try {
8             serverSocket = new ServerSocket(9000); ----- 서버 소켓을 생성합니다
9             socket = serverSocket.accept(); ----- 연결 요청이 오면 소켓을 생성합니다
10            InputStream in = socket.getInputStream();
11            OutputStream out = socket.getOutputStream();
12            byte arr[] = new byte[100];
13            in.read(arr); } 수신된 데이터를 출력합니다
14            System.out.println(new String(arr));
15            String str = "Hello, Client";
16            out.write(str.getBytes()); } 데이터 송신합니다
17        }
18        catch (Exception e) {
19            System.out.println(e.getMessage());
20        }
21        finally {
22            try {
23                socket.close(); ----- 소켓을 닫습니다
24            }
25            catch (Exception ignored) {
26            }
27            try {
28                serverSocket.close(); ----- 서버 소켓을 닫습니다
29            }
30            catch (Exception ignored) {
31            }
32        }
33    }
34 }

```

02. TCP/IP 통신 프로그램의 작성 방법

클라이언트/서버 프로그램 작성

[예제 20-1] TCP/IP로 통신하는 클라이언트 프로그램과 서버 프로그램 (1) - 실행 결과

컴퓨터 B

```
명령 프롬프트 - java ServerExample1
E:\work\chap20\20-2-1\example1\server>java ServerExample1
```

1) 서버 프로그램을 먼저 실행합니다.

```
명령 프롬프트
E:\work\chap20\20-2-1\example1\server>java ServerExample1
Hello, Server
E:\work\chap20\20-2-1\example1\server>_
```

3) 서버 프로그램은 클라이언트 프로그램으로부터 전송된 메시지를 출력합니다.

컴퓨터 A

```
명령 프롬프트
E:\work\chap20\20-2-1\example1\client>java ClientExample1_
```

2) 클라이언트 프로그램을 실행합니다.

```
명령 프롬프트
E:\work\chap20\20-2-1\example1\client>java ClientExample1
Hello, Client
E:\work\chap20\20-2-1\example1\client>
```

4) 클라이언트 프로그램도 서버 프로그램으로부터 전송된 메시지를 출력합니다.

02. TCP/IP 통신 프로그램의 작성 방법

클라이언트/서버 프로그램 작성

- 바이트 스트림을 문자 스트림으로 바꾸는 방법

참고

- InputStream 객체를 가지고 InputStreamReader 객체를 만들 수 있습니다.

```
InputStreamReader reader = new InputStreamReader(in);
```

↑
InputStream 객체

- OutputStream 객체를 가지고 PrintWriter 객체를 만들 수 있습니다.

```
PrintWriter writer = new PrintWriter(out);
```

↑
OutputStream 객체

02. TCP/IP 통신 프로그램의 작성 방법

클라이언트/서버 프로그램 작성

[예제 20-2] TCP/IP로 통신하는 클라이언트 프로그램과 서버 프로그램 (2)

클라이언트 프로그램

```

1 import java.io.*;
2 import java.net.*;
3 class ClientExample2 {
4     public static void main(String[] args) {
5         Socket socket = null;
6         try {
7             socket = new Socket("###.###.###.###", 9000);
8             BufferedReader reader = new BufferedReader(
9                 new InputStreamReader(socket.getInputStream()));
10            PrintWriter writer =
11                new PrintWriter(socket.getOutputStream());
12            writer.println("Hello, Server"); } 데이터를 송신합니다
13            writer.flush();
14            String str = reader.readLine(); } 수신된 데이터를 출력합니다
15            System.out.println(str);
16        }
17    } catch (Exception e) {
18        System.out.println(e.getMessage());
19    }
20    finally {
21        try {
22            socket.close();
23        } catch (Exception ignored) {
24        }
25    }
26 }
```

서버 프로그램

```

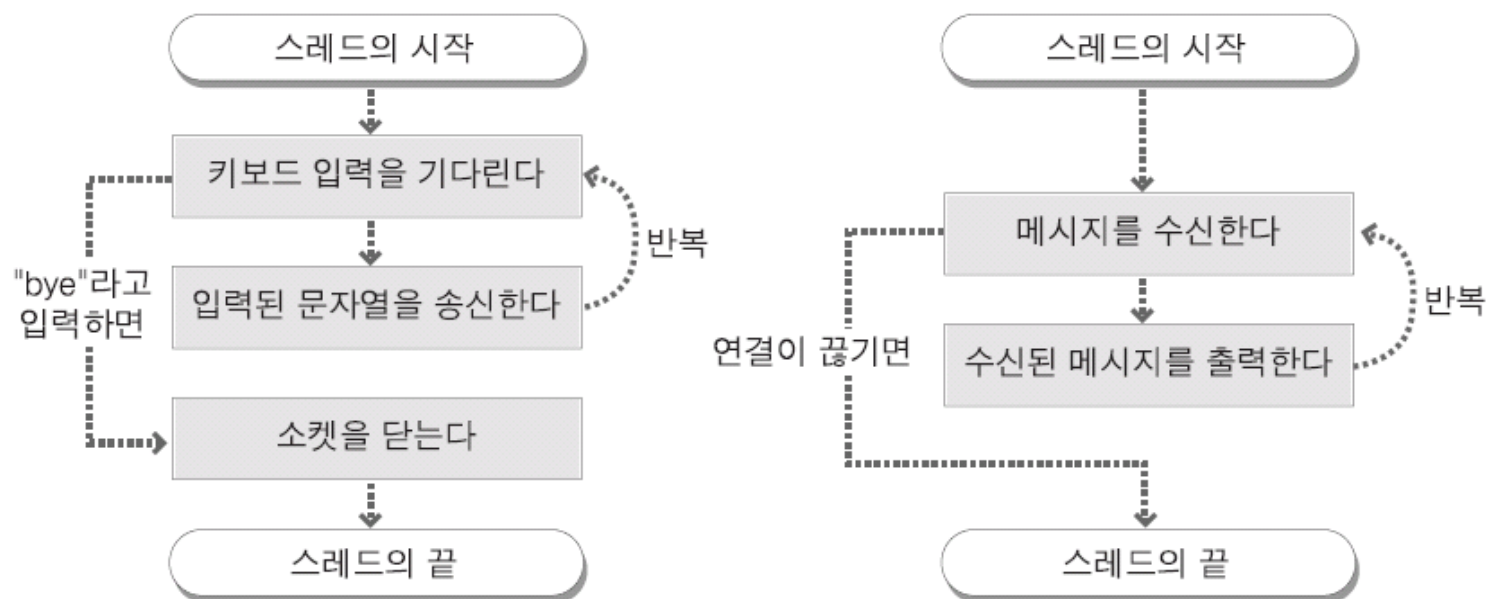
1 import java.io.*;
2 import java.net.*;
3 class ServerExample2 {
4     public static void main(String[] args) {
5         ServerSocket serverSocket = null;
6         Socket socket = null;
7         try {
8             serverSocket = new ServerSocket(9000);
9             socket = serverSocket.accept();
10            BufferedReader reader = new BufferedReader(
11                new InputStreamReader(socket.getInputStream()));
12            PrintWriter writer =
13                new PrintWriter(socket.getOutputStream());
14            String str = reader.readLine(); } 수신된 데이터를 출력합니다
15            System.out.println(str);
16            writer.println("Hello, Client"); } 데이터를 송신합니다
17            writer.flush();
18        } catch (Exception e) {
19            System.out.println(e.getMessage());
20        }
21    } finally {
22        try {
23            socket.close();
24        } catch (Exception ignored) {
25        }
26    } try {
27        serverSocket.close();
28    } catch (Exception ignored) {
29    }
30    }
31 }
32 }
33 }
```

실행 결과는 [예제 20-1]과 동일

02. TCP/IP 통신 프로그램의 작성 방법

송신과 수신을 동시에 하는 프로그램

- 클라이언트 프로그램과 서버 프로그램의 실행 흐름



02. TCP/IP 통신 프로그램의 작성 방법

송신과 수신을 동시에 하는 프로그램

[예제 20-3] 일대일 채팅 프로그램 – 클라이언트 프로그램

클라이언트 프로그램

```

1 import java.net.*;
2 class ClientExample3 {
3     public static void main(String[] args) {
4         Socket socket = null;
5         try {
6             socket = new Socket("###.###.###.###", 9001);
7             Thread thread1 = new SenderThread(socket);
8             Thread thread2 = new ReceiverThread(socket);
9             thread1.start();
10            thread2.start();
11        }
12        catch (Exception e) {
13            System.out.println(e.getMessage());
14        }
15    }
16 }

```

메시지를 송신하는 쓰레드 클래스

```

1 import java.io.*;
2 import java.net.*;
3 class SenderThread extends Thread {
4     Socket socket;
5     SenderThread(Socket socket) {
6         this.socket = socket;
7     }
8     public void run() {
9         try {
10            BufferedReader reader = new BufferedReader(
11                new InputStreamReader(System.in));
12            PrintWriter writer =
13                new PrintWriter(socket.getOutputStream());
14            while (true) {
15                String str = reader.readLine();
16                if (str.equals("bye"))
17                    break;
18                writer.println(str);
19                writer.flush();
20            }
21        }
22        catch (Exception e) {
23            System.out.println(e.getMessage());
24        }
25        finally {
26            try {
27                socket.close();
28            }
29            catch (Exception ignored) {
30            }
31        }
32    }
33 }

```

메시지를 수신하는 쓰레드 클래스

```

1 import java.io.*;
2 import java.net.*;
3 class ReceiverThread extends Thread {
4     Socket socket;
5     ReceiverThread(Socket socket) {
6         this.socket = socket;
7     }
8     public void run() {
9         try {
10            BufferedReader reader = new BufferedReader(
11                new InputStreamReader(socket.getInputStream()));
12            while (true) {
13                String str = reader.readLine();
14                if (str == null)
15                    break;
16                System.out.println("수신>" + str);
17            }
18        }
19        catch (Exception e) {
20            System.out.println(e.getMessage());
21        }
22    }
23 }

```

02. TCP/IP 통신 프로그램의 작성 방법

송신과 수신을 동시에 하는 프로그램

[예제 20-4] 일대일 채팅 프로그램 – 서버 프로그램

클라이언트 프로그램

```

1 import java.net.*;
2 class ServerExample3 {
3     public static void main(String[] args) {
4         ServerSocket serverSocket = null;
5         Socket socket = null;
6         try {
7             serverSocket = new ServerSocket(9001);
8             socket = serverSocket.accept();
9             Thread thread1 = new SenderThread(socket);
10            Thread thread2 = new ReceiverThread(socket);
11            thread1.start();
12            thread2.start();
13        }
14        catch (Exception e) {
15            System.out.println(e.getMessage());
16        }
17        finally {
18            try {
19                serverSocket.close();
20            }
21            catch (Exception ignored) {
22            }
23        }
24    }
25 }

```

메시지를 송신하는 쓰레드 클래스

```

1 import java.io.*;
2 import java.net.*;
3 class SenderThread extends Thread {
4     Socket socket;
5     SenderThread(Socket socket) {
6         this.socket = socket;
7     }
8     public void run() {
9         try {
10            BufferedReader reader = new BufferedReader(
11                new InputStreamReader(System.in));
12            PrintWriter writer =
13                new PrintWriter(socket.getOutputStream());
14            while (true) {
15                String str = reader.readLine();
16                if (str.equals("bye"))
17                    break;
18                writer.println(str);
19                writer.flush();
20            }
21        }
22        catch (Exception e) {
23            System.out.println(e.getMessage());
24        }
25        finally {
26            try {
27                socket.close();
28            }
29            catch (Exception ignored) {
30            }
31        }
32    }
33 }

```

메시지를 수신하는 쓰레드 클래스

```

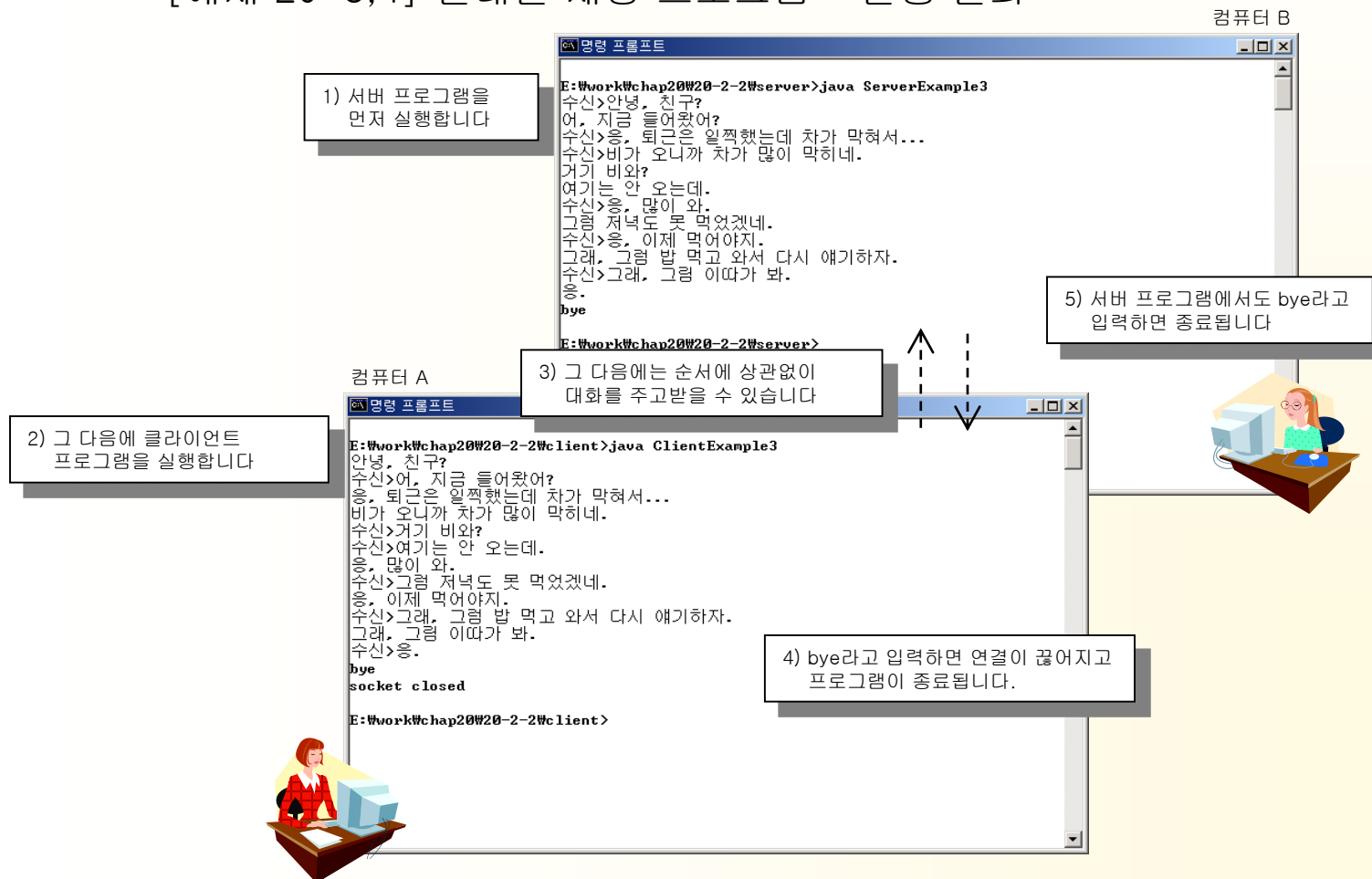
1 import java.io.*;
2 import java.net.*;
3 class ReceiverThread extends Thread {
4     Socket socket;
5     ReceiverThread(Socket socket) {
6         this.socket = socket;
7     }
8     public void run() {
9         try {
10            BufferedReader reader = new BufferedReader(
11                new InputStreamReader(socket.getInputStream()));
12            while (true) {
13                String str = reader.readLine();
14                if (str == null)
15                    break;
16                System.out.println("수신> " + str);
17            }
18        }
19        catch (Exception e) {
20            System.out.println(e.getMessage());
21        }
22    }
23 }

```


02. TCP/IP 통신 프로그램의 작성 방법

송신과 수신을 동시에 하는 프로그램

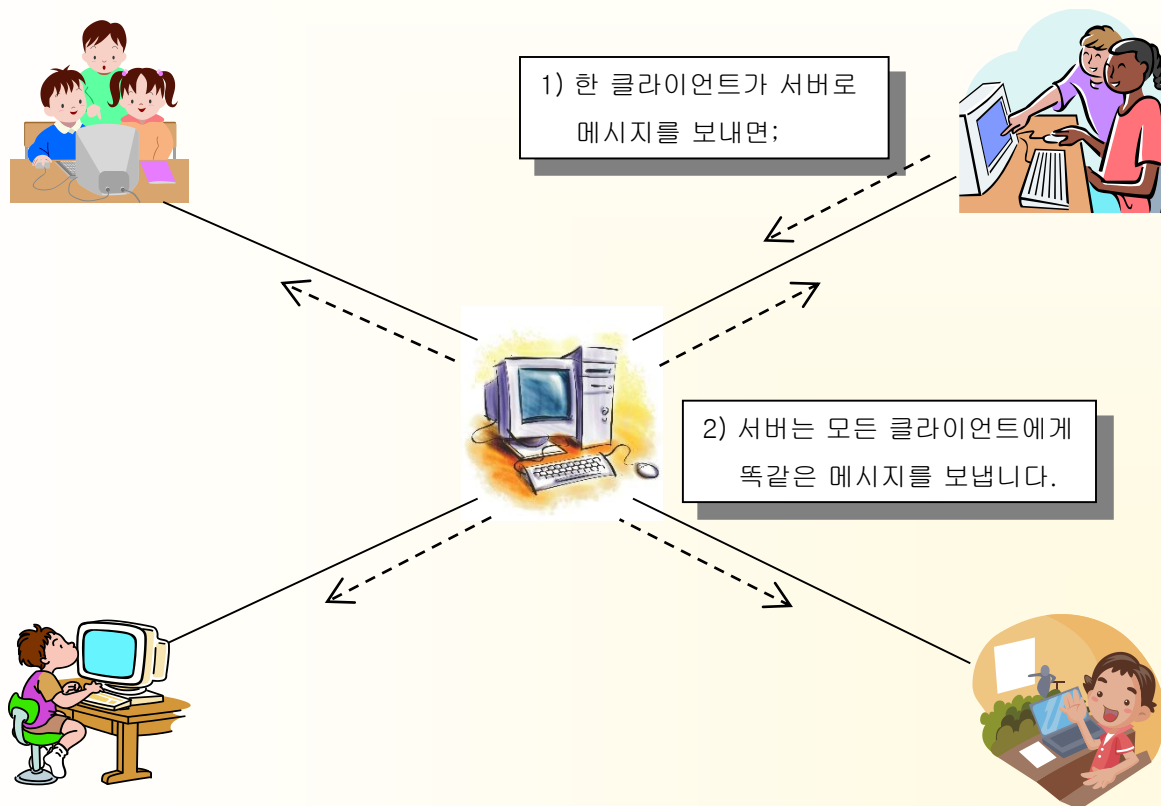
[예제 20-3,4] 일대일 채팅 프로그램 - 실행 결과



02. TCP/IP 통신 프로그램의 작성 방법

여러 명이 참여하는 채팅 프로그램

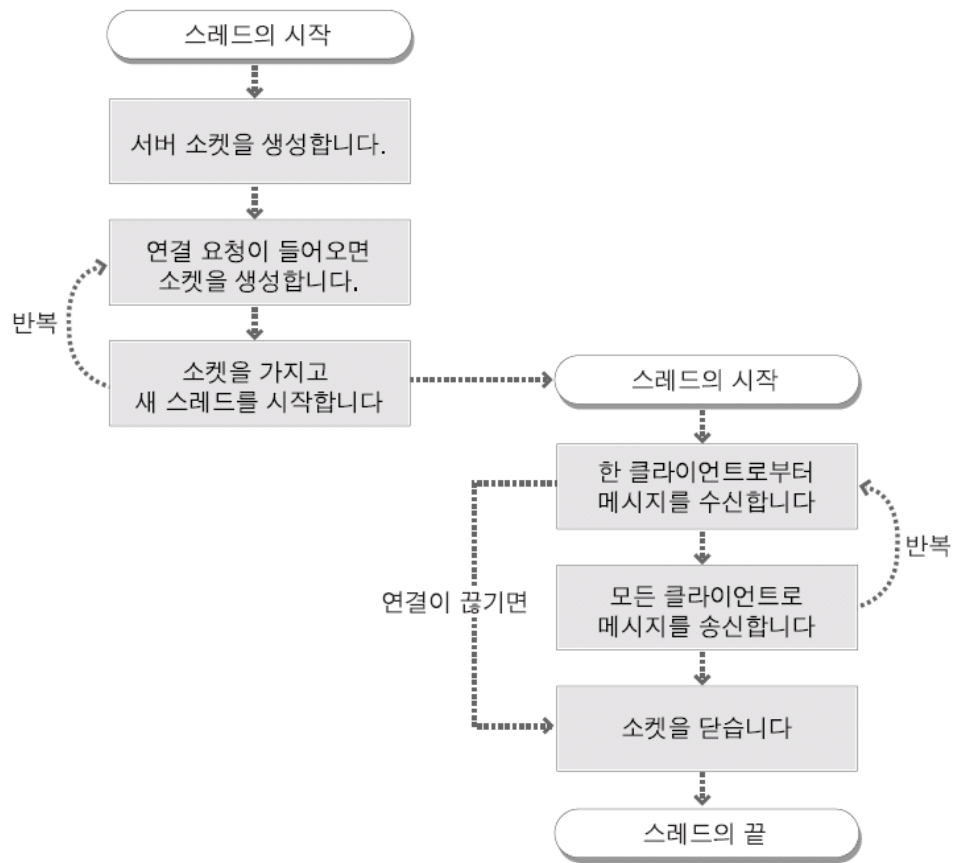
- 일반적인 채팅 프로그램의 작동 방식



02. TCP/IP 통신 프로그램의 작성 방법

여러 명이 참여하는 채팅 프로그램

- 여러 명이 참여하는 채팅 프로그램의 실행 흐름



02. TCP/IP 통신 프로그램의 작성 방법

여러 명이 참여하는 채팅 프로그램

[예제 20-5] 여러 사용자가 함께 채팅하는 프로그램 – 서버 프로그램 (미완성)

클라이언트 프로그램

```

1 import java.io.*;
2 import java.net.*;
3 class ClientExample1 {
4     public static void main(String[] args) {
5         Socket socket = null;
6         try {
7             socket = new Socket("###.###.###.###", 9000); --- 소켓을 생성합니다
8             InputStream in = socket.getInputStream();
9             OutputStream out = socket.getOutputStream();
10            String str = "Hello, Server";
11            out.write(str.getBytes()); } 데이터를 송신합니다
12            byte arr[] = new byte[100];
13            in.read(arr); } 수신된 데이터를 출력합니다
14            System.out.println(new String(arr));
15        }
16        catch (Exception e) {
17            System.out.println(e.getMessage());
18        }
19        finally {
20            try {
21                socket.close(); ----- 소켓을 닫습니다
22            }
23            catch (Exception e) {
24            }
25        }
26    }
27 }
```

서버 프로그램

```

1 import java.io.*;
2 import java.net.*;
3 class ServerExample1 {
4     public static void main(String[] args) {
5         ServerSocket serverSocket = null;
6         Socket socket = null;
7         try {
8             serverSocket = new ServerSocket(9000); ----- 서버 소켓을 생성합니다
9             socket = serverSocket.accept(); ----- 연결 요청이 오면 소켓을 생성합니다
10            InputStream in = socket.getInputStream();
11            OutputStream out = socket.getOutputStream();
12            byte arr[] = new byte[100];
13            in.read(arr); } 수신된 데이터를 출력합니다
14            System.out.println(new String(arr));
15            String str = "Hello, Client";
16            out.write(str.getBytes()); } 데이터를 송신합니다
17        }
18        catch (Exception e) {
19            System.out.println(e.getMessage());
20        }
21        finally {
22            try {
23                socket.close(); ----- 소켓을 닫습니다
24            }
25            catch (Exception ignored) {
26            }
27            try {
28                serverSocket.close(); ----- 서버 소켓을 닫습니다
29            }
30            catch (Exception ignored) {
31            }
32        }
33    }
34 }
```

02. TCP/IP 통신 프로그램의 작성 방법

여러 명이 참여하는 채팅 프로그램

- ArrayList 객체의 멀티 스레드 접근을 안전하게 만드는 방법

```
List list2 = Collections.synchronizedList(list1);
```

동기화된 ArrayList 객체

ArrayList 객체

02. TCP/IP 통신 프로그램의 작성 방법

여러 명이 참여하는 채팅 프로그램

[예제 20-6] 여러 사용자가 함께 채팅하는 프로그램 – 서버 프로그램 (완성)

클라이언트 프로그램

```

1 import java.io.*;
2 import java.net.*;
3 class ClientExample1 {
4     public static void main(String[] args) {
5         Socket socket = null;
6         try {
7             socket = new Socket("###.###.###.###", 9000); --- 소켓을 생성합니다
8             InputStream in = socket.getInputStream();
9             OutputStream out = socket.getOutputStream();
10            String str = "Hello, Server";
11            out.write(str.getBytes()); } 데이터를 송신합니다
12            byte arr[] = new byte[100];
13            in.read(arr); } 수신된 데이터를 출력합니다
14            System.out.println(new String(arr));
15        }
16        catch (Exception e) {
17            System.out.println(e.getMessage());
18        }
19        finally {
20            try {
21                socket.close(); ----- 소켓을 닫습니다
22            }
23            catch (Exception e) {
24            }
25        }
26    }
27 }

```

서버 프로그램

```

1 import java.io.*;
2 import java.net.*;
3 class ServerExample1 {
4     public static void main(String[] args) {
5         ServerSocket serverSocket = null;
6         Socket socket = null;
7         try {
8             serverSocket = new ServerSocket(9000); ----- 서버 소켓을 생성합니다
9             socket = serverSocket.accept(); ----- 연결 요청이 오면 소켓을 생성합니다
10            InputStream in = socket.getInputStream();
11            OutputStream out = socket.getOutputStream();
12            byte arr[] = new byte[100];
13            in.read(arr); } 수신된 데이터를 출력합니다
14            System.out.println(new String(arr));
15            String str = "Hello, Client";
16            out.write(str.getBytes()); } 데이터를 송신합니다
17        }
18        catch (Exception e) {
19            System.out.println(e.getMessage());
20        }
21        finally {
22            try {
23                socket.close(); ----- 소켓을 닫습니다
24            }
25            catch (Exception ignored) {
26            }
27            try {
28                serverSocket.close(); ----- 서버 소켓을 닫습니다
29            }
30            catch (Exception ignored) {
31            }
32        }
33    }
34 }

```

02. TCP/IP 통신 프로그램의 작성 방법

여러 명이 참여하는 채팅 프로그램

[예제 20-7] 여러 사용자가 함께 채팅하는 프로그램 – 클라이언트 프로그램

클라이언트 프로그램

```
1 import java.net.*;
2 class ClientExample4 {
3     public static void main(String[] args) {
4         if (args.length != 1) {
5             System.out.println(
6                 "Usage: java ClientExample4 <user-name>");
7             return;
8         }
9         try {
10             Socket socket = new Socket("###.###.###.###",
11                                     9002);
12             Thread thread1 = new SenderThread(socket,
13                                                args[0]);
14             Thread thread2 = new ReceiverThread(socket);
15             thread1.start();
16             thread2.start();
17         }
18         catch (Exception e) {
19             System.out.println(e.getMessage());
20         }
21     }
22 }
```

메시지를 송신하는 쓰레드 클래스

```
1 import java.net.*;
2 import java.io.*;
3 class SenderThread extends Thread {
4     Socket socket;
5     String name;
6     SenderThread(Socket socket, String name) {
7         this.socket = socket;
8         this.name = name;
9     }
10    public void run() {
11        try {
12            BufferedReader reader = new BufferedReader(
13                new InputStreamReader(System.in));
14            PrintWriter writer =
15                new PrintWriter(socket.getOutputStream());
16            writer.println(name);
17            writer.flush();
18            while (true) {
19                String str = reader.readLine();
20                if (str.equals("bye"))
21                    break;
22                writer.println(str);
23                writer.flush();
24            }
25        }
26        catch (Exception e) {
27            System.out.println(e.getMessage());
28        }
29        finally {
30            try {
31                socket.close();
32            }
33            catch (Exception ignored) {}
34        }
35    }
36 }
```

메시지를 수신하는 쓰레드 클래스

```
1 import java.io.*;
2 import java.net.*;
3 class ReceiverThread extends Thread {
4     Socket socket;
5     ReceiverThread(Socket socket) {
6         this.socket = socket;
7     }
8    public void run() {
9        try {
10            BufferedReader reader = new BufferedReader(
11                new InputStreamReader(socket.getInputStream()));
12            while (true) {
13                String str = reader.readLine();
14                if (str == null)
15                    break;
16                System.out.println(str);
17            }
18        }
19        catch (IOException e) {
20            System.out.println(e.getMessage());
21        }
22    }
23 }
```

02. TCP/IP 통신 프로그램의 작성 방법

여러 명이 참여하는 채팅 프로그램

[예제 20-6,7] 실행 결과

컴퓨터 A

```
E:\work\chap20\20-2-3\example2\server>java ServerExample4
```

1) 서버 프로그램을 먼저 실행합니다



컴퓨터 C

```
E:\work\chap20\20-2-3\example2\client>java ClientExample4 마술피리
#마술피리님이 들어오셨습니다
안녕하세요?
마술피리>안녕하세요?
체리트리>안녕하세요, 마술피리님
#스컬리님이 들어오셨습니다
스컬리>안녕하세요?
스컬리>제가 너무 늦었죠?
아니요, 저도 지금 막...
마술피리>아니요, 저도 지금 막...
체리트리>...
말더듬이>...
#말더듬이님이 들어오셨습니다
말더듬이>말더듬이요?
스컬리>오를 못 온다고 전해달라고 했는데...
말더듬이>왜요?
스컬리>아근이래요...
스컬리>그럼 우리끼리 얘기해서 결정한 다음에 말더듬이께는 메일로 보내기로 하죠.
```

3) 마찬가지로 방법으로 다른 컴퓨터에서 클라이언트 프로그램을 실행합니다

컴퓨터 B

```
E:\work\chap20\20-2-3\example2\client>java ClientExample4 체리트리
#체리트리님이 들어오셨습니다
안녕하세요?
체리트리>안녕하세요?
아무도 없나요?
체리트리>아무도 없나요?

체리트리>...
마술피리>안녕하세요?
마술피리>안녕하세요, 마술피리님
안녕하세요, 마술피리님
체리트리>안녕하세요, 마술피리님
#스컬리님이 들어오셨습니다
스컬리>안녕하세요?
스컬리>제가 너무 늦었죠?
마술피리>아니요, 저도 지금 막...

체리트리>...
말더듬이>말더듬이요?
스컬리>오를 못 온다고 전해달라고 했는데...
마술피리>왜요?
스컬리>아근이래요...
스컬리>우리끼리 얘기해서 결정한 다음에 말더듬이께는 메일로 보내기로 하죠.
체리트리>그럼 우리끼리 얘기해서 결정한 다음에 말더듬이께는 메일로 보내기로 하죠.
```

2) 닉네임을 명령행 파라미터로 사용하여 클라이언트 프로그램을 시작합니다



컴퓨터 D

```
E:\work\chap20\20-2-3\example2\client>java ClientExample4 스컬리
#스컬리님이 들어오셨습니다
안녕하세요?
스컬리>안녕하세요?
제가 너무 늦었죠?
스컬리>제가 너무 늦었죠?
마술피리>아니요, 저도 지금 막...
체리트리>...
말더듬이>말더듬이요?
마술피리>오를 못 온다고 전해달라고 했는데...
스컬리>오를 못 온다고 전해달라고 했는데...
마술피리>왜요?
아근이래요...
스컬리>아근이래요...
스컬리>그럼 우리끼리 얘기해서 결정한 다음에 말더듬이께는 메일로 보내기로 하죠.
```

4) 또 다른 컴퓨터에서도 같은 방법으로 클라이언트 프로그램을 실행합니다

