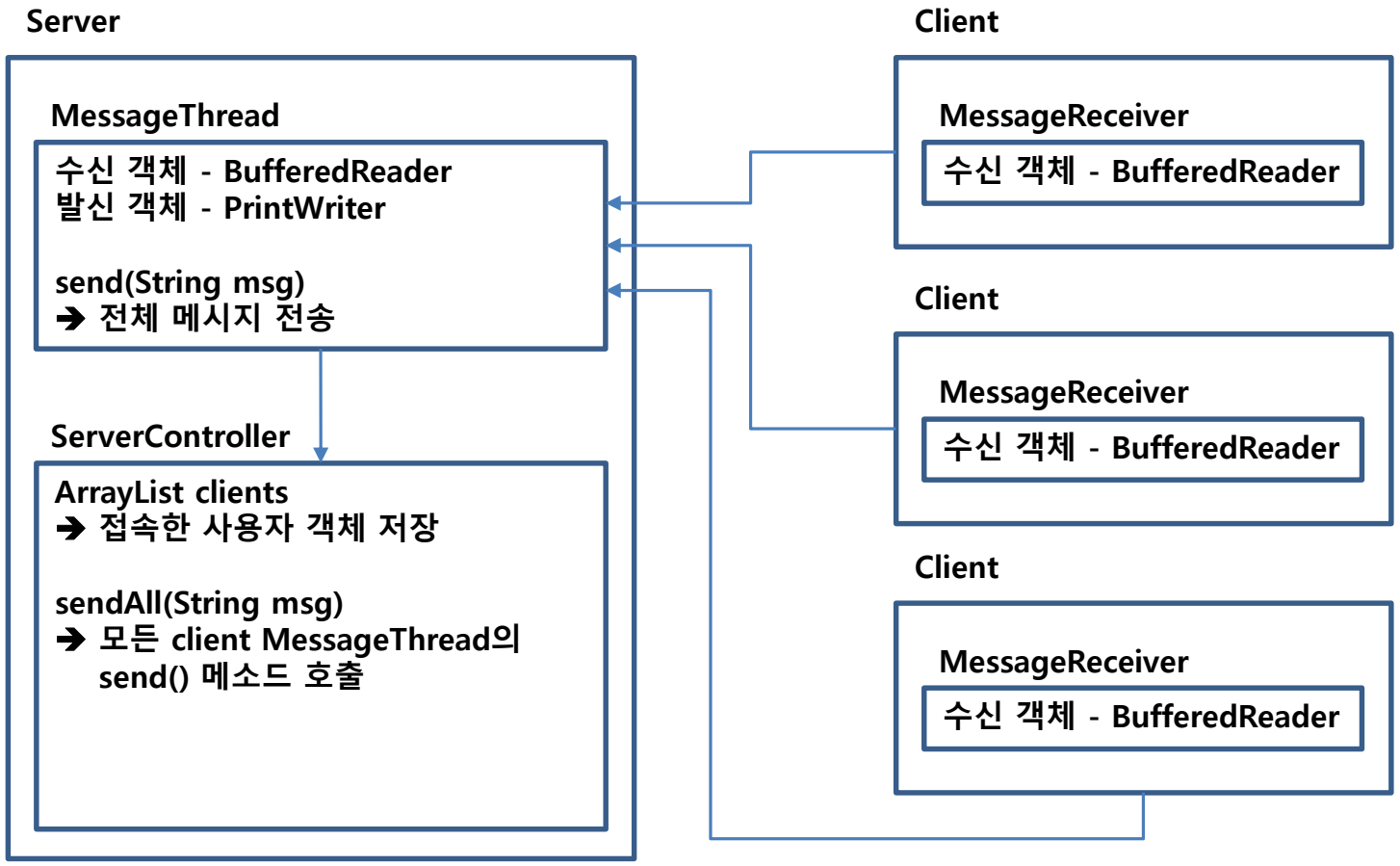
■ Multi Socket 통신

● Server

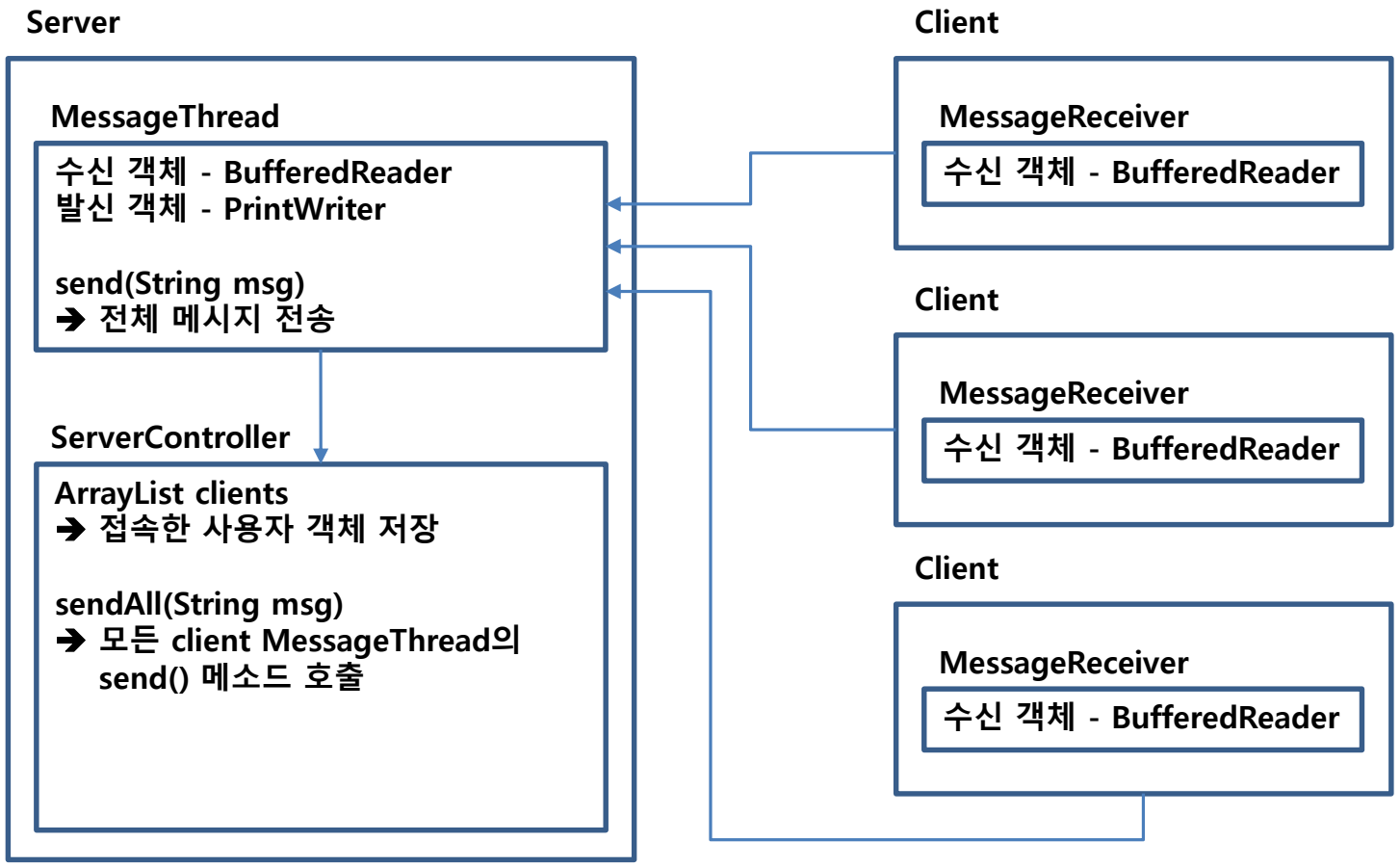
- MessageThread (접속한 사용자의 개별 Thread)
- ServerController (개발 Thread-MessageThread 관리)



■ Multi Socket 통신

● Client

- MessageReceiver



■ Socket 통신 - Server

● socket.multi.server.ServerController.java (1 / 2)

```
/**
 * 접속 클라이언트 관리
 * 접속자 추가, 제거, 모든 클라이언트에게 메세지 전송
 */
public class ServerController {
    private static ServerController controller = new ServerController();
    private ArrayList<MessageThread> clients = new ArrayList<MessageThread>();

    // 싱글톤 패턴 사용을 위해서 생성자 숨김 (private)
    private ServerController() {}

    public static ServerController getInstance() {
        if(controller == null) {
            controller = new ServerController();
        }
        return controller;
    }
```

■ Socket 통신 - Server

● socket.multi.server.ServerController.java (2 / 2)

```
    // 클라이언트 접속시 추가
    public void addClient(MessageThread client) {
        clients.add(client);
    }

    // 클라이언트 퇴장시 제거
    public void removeClient(MessageThread client) {
        clients.remove(client);
    }

    // 현재 접속되어 있는 모든 클라이언트(Thread)에게 메세지 전송
    public void sendAll(String message) {
        for(MessageThread client : clients) {
            client.send(message);
        }
    }
}
```

■ Socket 통신 - Server

● socket.multi.server.MessageThread.java (1 / 3)

```
/**
 * 서버로 접속한 클라이언트 ex) 2명 접속시 2개의 객체 생성
 * 메세지를 보내고 받을 수 있는 실제 입출력스트림을 가짐
 */
public class MessageThread extends Thread {
    private BufferedReader reader;
    private PrintWriter writer;
    private ServerController controller;
    private String nickName;
    private Socket socket;
    // 생성자를 이용하여 입출력스트림과 닉네임 저장
    public MessageThread(Socket socket, BufferedReader reader, PrintWriter writer, String
      nickName) {
        this.socket = socket;   this.reader = reader;   this.writer = writer;
        this.controller = ServerController.getInstance();
        this.nickName = nickName;
    }
```

■ Socket 통신 - Server

● socket.multi.server.MessageThread.java (2 / 3)

```
@Override
public void run() {
    boolean isContinue = true;
    while(isContinue) {
        try {
            String message = reader.readLine(); // 클라이언트 메세지 수신
            // 메세지 수신시 모든 클라이언트에게 메세지 전달
            controller.sendAll("[" + this.nickName + "] " + message);
        } catch (IOException e) {
            // 클라이언트와의 연결 해제시 퇴장으로 간주, 입출력스트림 해제
            isContinue = false;
            controller.removeClient(this);
            controller.sendAll("[" + nickName + "] 퇴장");
            if(reader != null) {
                try { reader.close(); } catch (IOException e1) {}
            }
            if(writer != null) { writer.close(); }
```

■ Socket 통신 - Server

● socket.multi.server.MessageThread.java (3 / 3)

```
                if(socket != null) {
                    try { socket.close(); } catch (IOException e1) {}
                }
            }
        }
    }

    public void send(String message) {
        this.writer.println(message);
    }
}
```

■ Socket 통신 - Server

● socket.multi.server.ServerMain.java (1 / 3)

```
/**
 * 서버의 메인 프로그램
 * 클라이언트 접속시 Thread 생성, 컨트롤러에게 접속자 알림
 */
public class ServerMain {

    public static void main(String[] args) {
        ServerSocket sSocket = null; // 서버 사용 소켓
        Socket socket = null; // 클라이언트와 연결될 소켓

        InputStream in = null;
        InputStreamReader isr = null;
        BufferedReader reader = null;

        OutputStream out = null;
        PrintWriter writer = null;
```

■ Socket 통신 - Server

● socket.multi.server.ServerMain.java (2 / 3)

```
try {
    sSocket = new ServerSocket(20000);

    // 컨트롤러 객체 생성 (싱글톤 패턴 - 객체 1개만 생성)
    ServerController controller = ServerController.getInstance();

    while(true) {
        System.out.println("접속 대기");
        socket = sSocket.accept(); // 클라이언트 접속 대기

        in = socket.getInputStream();
        isr = new InputStreamReader(in, "utf-8");
        reader = new BufferedReader(isr);

        out = socket.getOutputStream();
        writer = new PrintWriter(out, true);
```

■ Socket 통신 - Server

● socket.multi.server.ServerMain.java (3 / 3)

```
                String nickName = reader.readLine(); // 최초 메세지를 이용해서 닉네임 저장
                System.out.println("[" + nickName + "] 접속");

                // 클라이언트 1개 당 스레드 1개 생성
                MessageThread client = new MessageThread(socket, reader, writer, nickName);
                client.start();

                // 컨트롤러에게 접속자를 알려줘서 추가
                controller.addClient(client);
            }
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

■ Socket 통신 - Client

● socket.multi.client.MessageReceiver.java (1 / 2)

```
/**
 * 서버 메세지 수신
 */
public class MessageReceiver extends Thread {
    private BufferedReader reader;

    public MessageReceiver(BufferedReader reader) {
        this.reader = reader;
    }
```

■ Socket 통신 - Client

● socket.multi.client.MessageReceiver.java (2 / 2)

```
    @Override
    public void run() {
        boolean isContinue = true;
        while(isContinue) {
            try {
                String message = reader.readLine();
                System.out.println(message);
            } catch (IOException e) {
                isContinue = false;
            }
        }
    }
}
```

■ Socket 통신 - Client

● socket.multi.client.ClientMain.java (1 / 3)

```
/**
 * 클라이언트의 메인 프로그램
 */
public class ClientMain {

    public static void main(String[] args) {
        Socket socket = null;

        InputStream in = null;
        InputStreamReader isr = null;
        BufferedReader reader = null;

        OutputStream out = null;
        PrintWriter writer = null;

        Scanner scan = new Scanner(System.in);
```

■ Socket 통신 - Client

● socket.multi.client.ClientMain.java (2 / 3)

```
try {
    socket = new Socket("127.0.0.1", 20000);

    in = socket.getInputStream();
    isr = new InputStreamReader(in, "utf-8");
    reader = new BufferedReader(isr);

    out = socket.getOutputStream();
    writer = new PrintWriter(out, true);

    new MessageReceiver(reader).start();

    // 서버 접속 후 닉네임 전달
    System.out.println("닉네임을 입력하세요.");
    System.out.print("=> ");
    String nickName = scan.nextLine();
    writer.println(nickName);
```

■ Socket 통신 - Client

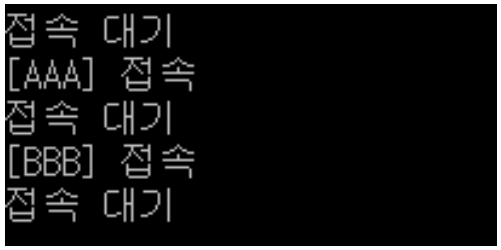
● socket.multi.client.ClientMain.java (3 / 3)

```
            // 메세지 입력 대기
            while(true) {
                String msg = scan.nextLine();
                writer.println(msg);
            }
        } catch (UnknownHostException e) {
            e.printStackTrace();
        } catch (IOException e) {
            e.printStackTrace();
        } finally {
            scan.close();
            writer.close();
        }
    }
}
```

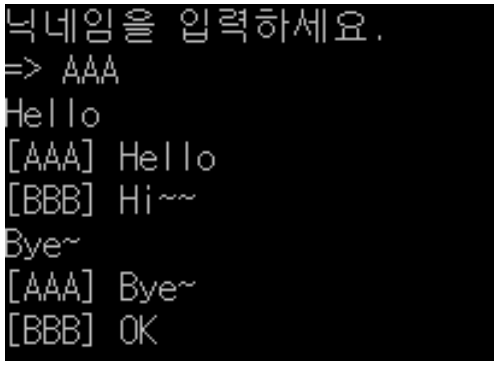
■ Socket 통신

● 실행결과

- Server



- Client1



- Client2