

2강

TCP 소켓 프로그래밍 I

1. TCP 소켓 프로그래밍 기본

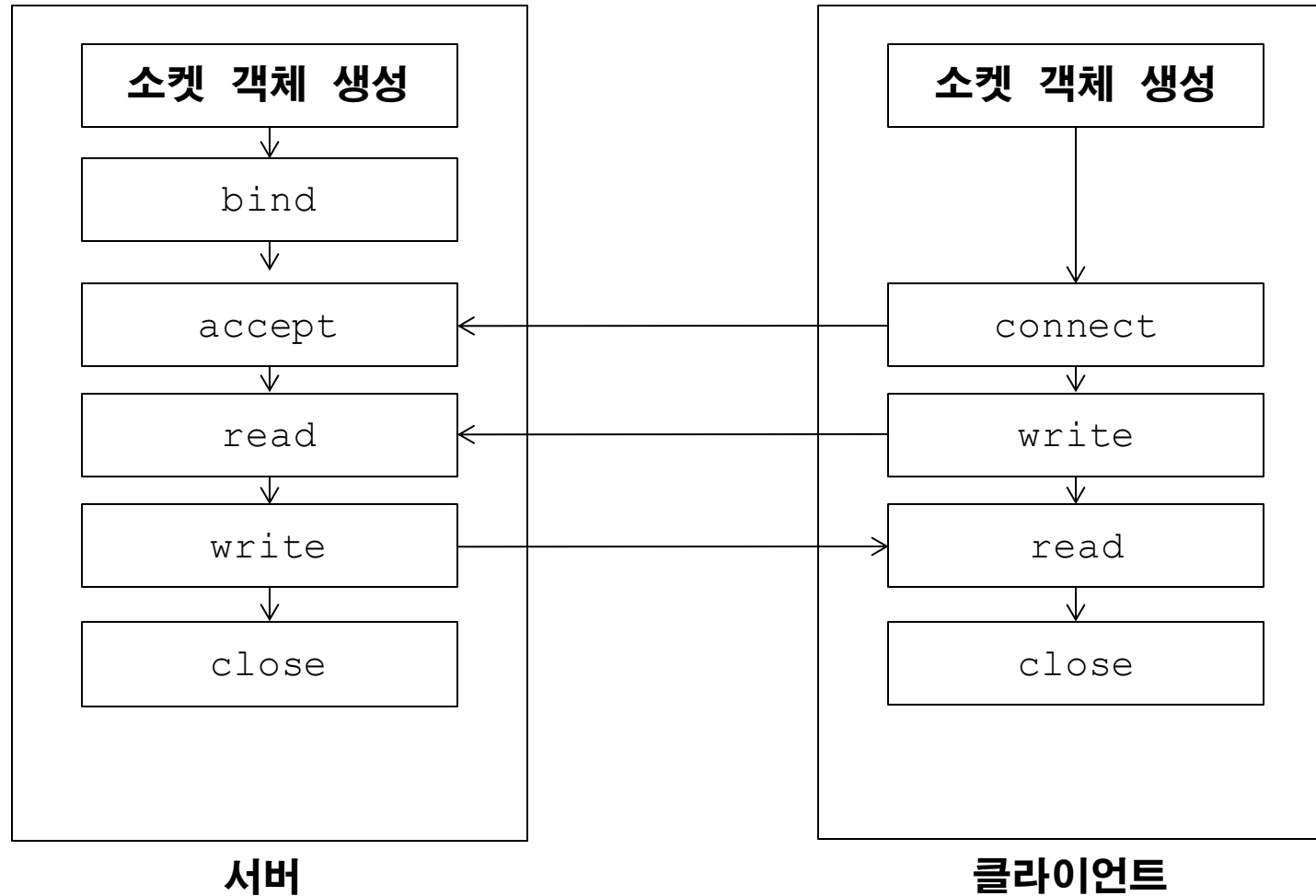
- 1. TCP 소켓 프로그래밍 기본
- 2. TCP 서버
- 3. TCP 클라이언트

1. TCP 소켓 프로그래밍 기본

- ❑ 스트림(stream) 통신 프로토콜
- ❑ 양쪽의 소켓이 연결된 상태에서 통신이 가능하다. (연결지향 프로토콜)
- ❑ 신뢰성 있는 데이터 통신
- ❑ 한 번 연결이 되면 연결이 끊어 질 때까지 송신한 데이터는 차례대로 목적지의 소켓에 전달
- ❑ 자바는 `java.net` 패키지에 **TCP** 소켓 프로그래밍을 쉽게 하도록 관련 클래스를 제공하고 있다.
- ❑ 라이브러리의 사용법과 동작순서를 정확하게 이해하고 있어야 한다.
- ❑ **ServerSocket**과 **Socket** 클래스를 사용하게 된다.

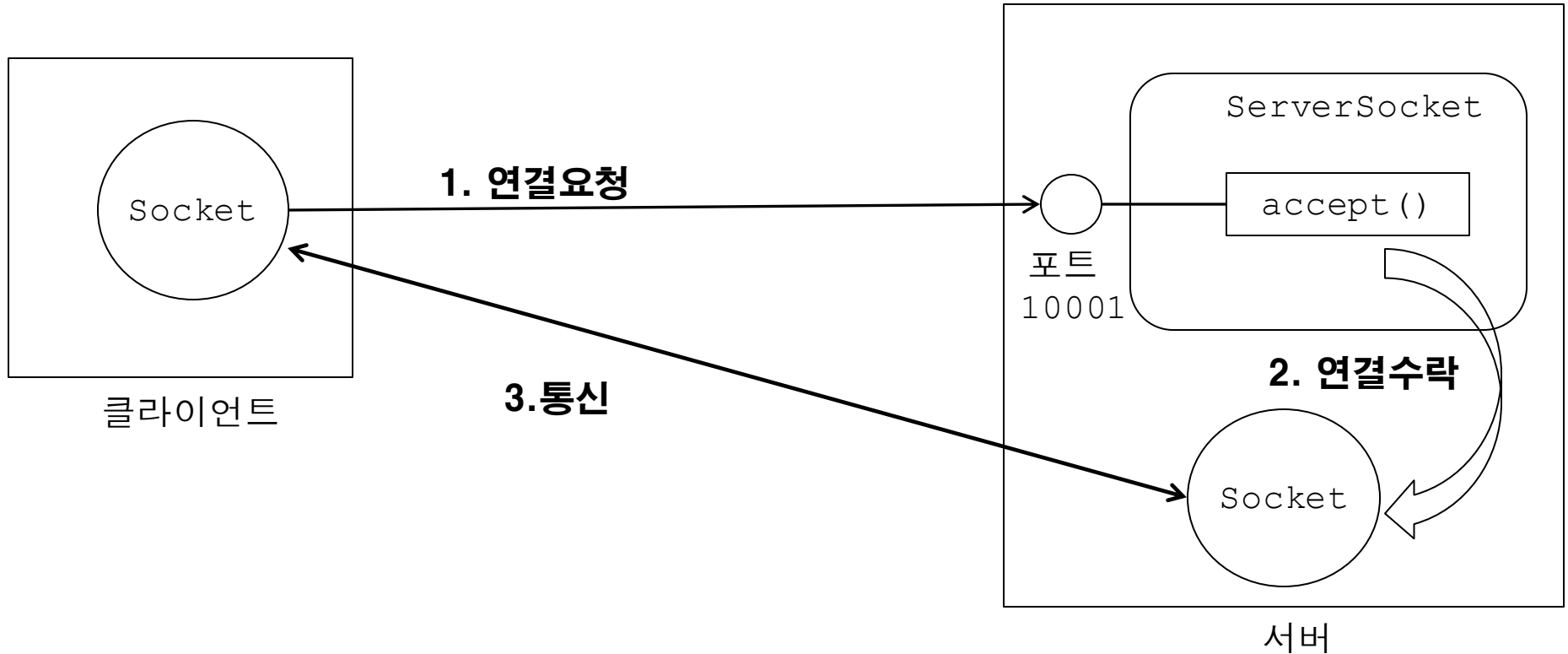
1. TCP 소켓 프로그래밍 기본

□ TCP 소켓 프로그래밍 절차



1. TCP 소켓 프로그래밍 기본

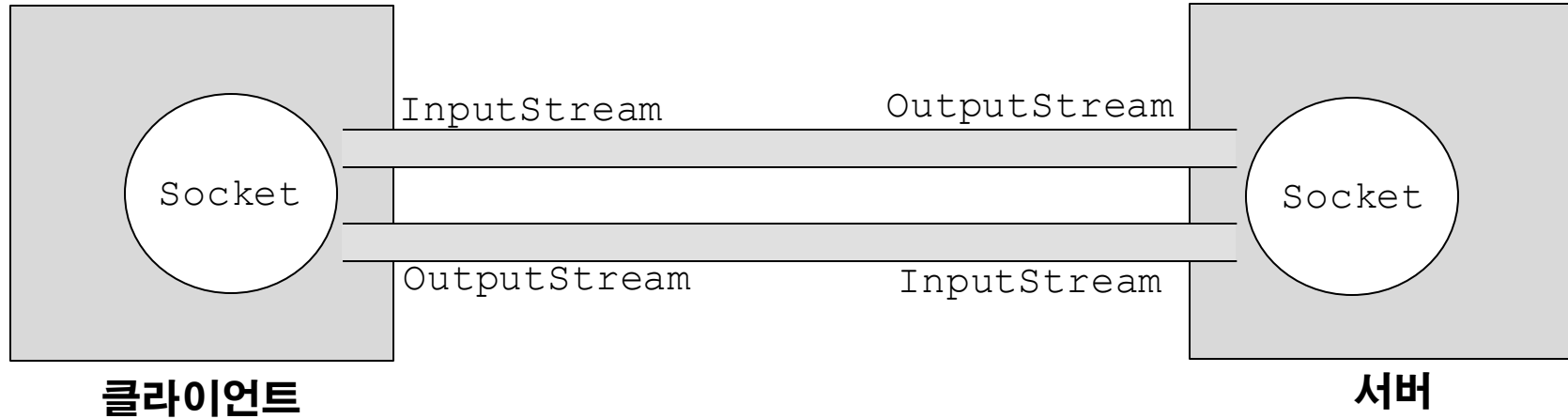
❑ ServerSocket과 Socket



- **ServerSocket** : 클라이언트의 연결요청을 기다리면서 연결 요청에 대한 수락을 담당한다.
- **Socket** : 클라이언트와 통신을 직접 담당한다.

1. TCP 소켓 프로그래밍 기본

□ Socket 객체의 데이터 통신



- 양쪽의 **Socket** 객체로 부터 **InputStream**, **OutputStream**를 얻어와 데이터 통신에 사용한다.
- 보낸 데이터는 **byte[]** 배열로 생성해서 **write()**, **read()**의 매개변수로 전달하여 데이터를 전송하게 된다.

2강

TCP 소켓 프로그래밍 I

2. TCP 서버

- 1. TCP 소켓 프로그래밍 기본
- 2. TCP 서버
- 3. TCP 클라이언트

1. TCP Server 작성을 위한 기본 코드

❑ 서버

1. TCP 소켓 객체를 생성

```
ServerSocket severSocket = new ServerSocket();
```

2. 소켓을 호스트의 포트와 연결(Bind)

```
serverSocket.bind( new InetSocketAddress( "localhost", 10001 ) );
```

3. 클라이언트로 부터 소켓 연결이 올 때 까지 대기(Accept)한다.

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

4. 연결이 되면 통신을 위한 **stream** 객체를 얻는다.

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

1. TCP Server 작성을 위한 기본 코드

❑ 서버

5. 데이터 통신을 한다.

```
is.read( buffer );  
os.write( buffer );
```

6. 클라이언트와 연결을 종료

```
is.close();  
os.close();  
Socket.close();
```


2. TCP Server – 연결 요청 수락

[실습2-1]

TCP 통신 서버의 기본 코드를 작성하고 몇 가지 테스트를 하는 실습이다.

다음 코드를 참고해서 코드를 작성하고 실행 시켜보자.

```
01 public class TCPServer {
02     public static void main(String[] args) {
03         ServerSocket serverSocket = null ;
04         try {
05             serverSocket = new ServerSocket();
06             serverSocket.bind( new InetSocketAddress( "localhost", 10001 ) );
07             System.out.println( "[서버] 연결 기다림");
08             Socket socket = serverSocket.accept();
09             System.out.println( "[서버] 연결됨");
10             socket.close();
11         } catch (IOException e) {
12             e.printStackTrace();
13         } finally {
14             if( serverSocket != null && serverSocket.isClosed() == false ) {
15                 try {
16                     serverSocket.close();
17                 } catch( IOException ex ) {
18                     ex.printStackTrace();
19                 }
20             }
21         }
22     }
23 }
```

2. TCP Server – 연결 요청 수락

[실습2-1]

아직 클라이언트 프로그램을 작성하지 않았기 때문에 **telnet**으로 서버연결 테스트를 할 수 있다.

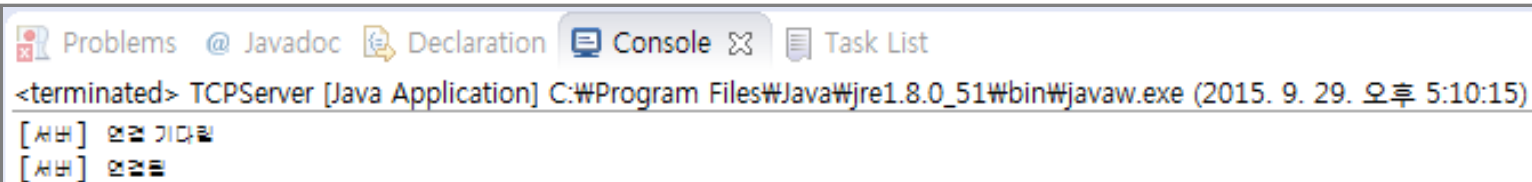
```
[c:\~]$ telnet 127.0.0.1 10001

Connecting to 127.0.0.1:10001...
Connection established.
To escape to local shell, press 'Ctrl+Alt+J'.

Connection closed by foreign host.

Disconnected from remote host(127.0.0.1:10001) at 17:20:51.

Type `help' to learn how to use Xshell prompt.
```



The screenshot shows an IDE interface with tabs for Problems, Javadoc, Declaration, Console, and Task List. The Console tab is active, displaying the following text:

```
<terminated> TCPServer [Java Application] C:\Program Files\Java\jre1.8.0_51\bin\javaw.exe (2015. 9. 29. 오후 5:10:15)
[서버] 연결 기다림
[서버] 연결됨
```

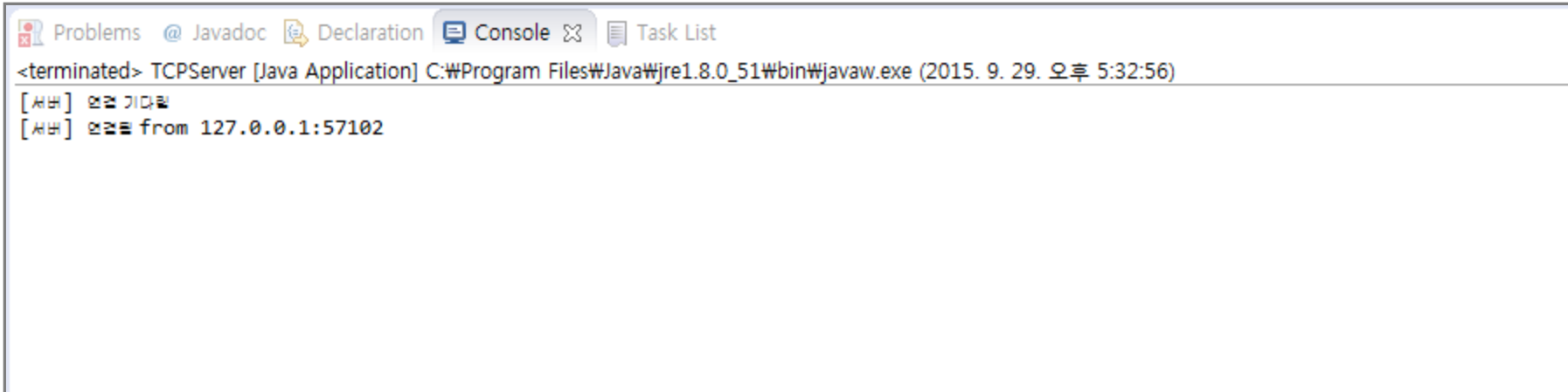
3. TCP Server – 클라이언트 정보 얻어오기

[실습2-2]

연결 요청을 수락한 후, 클라이언트의 정보(IP, 포트)를 얻어서 화면에 출력하는 실습이다.
다음 코드를 참고해서 수정하고 다시 서버를 실행시키고 확인해 보자.

```
InetSocketAddress inetSocketAddress = (InetSocketAddress) socket.getRemoteSocketAddress();  
System.out.println( "[서버] 연결됨 from " +  
                    inetSocketAddress.getHostName() + ":" +  
                    inetSocketAddress.getPort() );
```

실행 결과



```
<terminated> TCPServer [Java Application] C:\Program Files\Java\jre1.8.0_51\bin\javaw.exe (2015. 9. 29. 오후 5:32:56)  
[서버] 연결 기다림  
[서버] 연결됨 from 127.0.0.1:57102
```

4. TCP Server – 데이터 보내기

[실습2-3]

연결 요청을 수락한 후, 연결을 끊기 전에 데이터를 보내고 연결을 끊는 실습이다.
다음 코드를 참고해서 수정하고 다시 서버를 실행시키고 확인해 보자.

```
// 데이터 보내기
OutputStream os = socket.getOutputStream();

String data = "Hello World";
os.write( data.getBytes( "UTF-8" ) );
os.flush();
os.close();
```

실행 결과

```
[c:\~]$ telnet 127.0.0.1 10001

Connecting to 127.0.0.1:10001...
Connection established.
To escape to local shell, press 'Ctrl+Alt+J'.
Hello World
Connection closed by foreign host.

Disconnected from remote host(127.0.0.1:10001) at 17:53:14.

Type `help' to learn how to use Xshell prompt.
```

5. TCP Server – 데이터 받기

[실습2-4]

연결 요청을 수락한 후, 클라이언트로 부터 계속 데이터를 받아 서버 콘솔에 출력하는 실습이다.
다음 코드를 참고해서 수정하고 다시 서버를 실행시키고 확인해 보자.

```
// 데이터 받기
InputStream is = socket.getInputStream();
while( true ) {
    byte[] buffer = new byte[128];
    int readByteCount = is.read( buffer );

    if( readByteCount < 0 ) {
        System.out.println( "[서버] 클라이언트로 부터 연결끊김" );
        is.close();
        socket.close();
        break;
    }

    String data = new String( buffer, 0, readByteCount, "UTF-8" );
    System.out.print( data );
}
```

1. **read** 메소드 는 읽은 데이터만큼의 바이트 수를 반환한다.
2. **-1**이 반환되면 상대방이 정상적으로 연결을 끊었다.
3. 상대방이 비정상적으로 종료하면 **IOException** 이 발생한다.

5. TCP Server – 데이터 받기

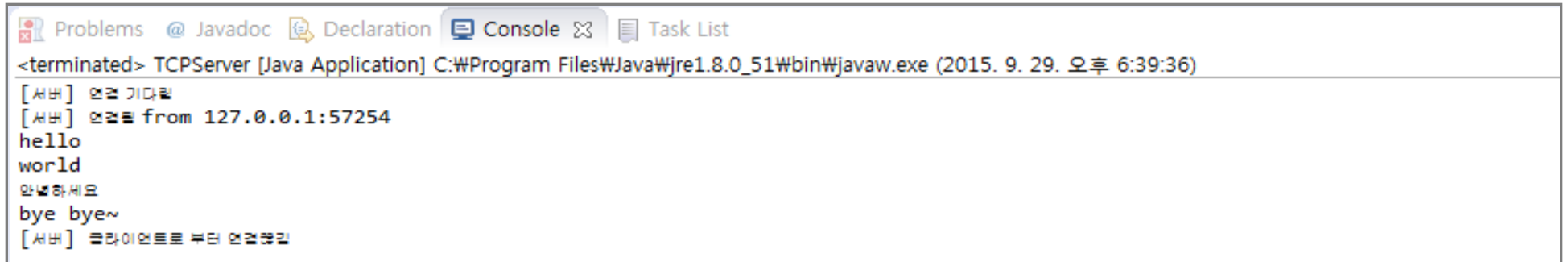
[실습2-4]

```
[c:\~]$ telnet 127.0.0.1 10001

Connecting to 127.0.0.1:10001...
Connection established.
To escape to local shell, press 'Ctrl+Alt+J'.
hello
world
안녕하세요
bye bye~

Escape to local shell..... To return to remote host, enter "exit".
To close connection, enter "disconnect".

Type `help' to learn how to use Xshell prompt.
[c:\~]$ disconnect
Connection closed.
Disconnected from remote host(127.0.0.1:10003) at 18:40:10.
```



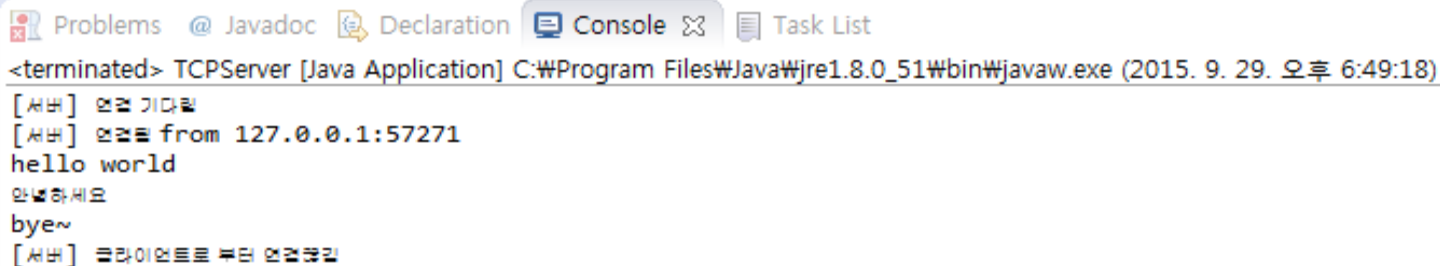
6. 과제 TCP Server 완성 (에코 기능 추가)

[과제2-1] TCP Server 모두 다 합치기

연결 요청을 수락한 후, 데이터를 받아 다시 보내는 에코 기능을 가진 서버를 작성해 봅니다.

```
[c:\~]$ telnet 127.0.0.1 10001
Connecting to 127.0.0.1:10001...
Connection established.
To escape to local shell, press 'Ctrl+Alt+J'.
hello world
hello world
안녕하세요
안녕하세요
bye~
bye~

Escape to local shell..... To return to remote host, enter "exit".
To close connection, enter "disconnect".
Type `help` to learn how to use Xshell prompt.
[c:\~]$ disconnect
Connection closed.
Disconnected from remote host(127.0.0.1:10003) at 18:50:04.
```



```
<terminated> TCPServer [Java Application] C:\Program Files\Java\jre1.8.0_51\bin\javaw.exe (2015. 9. 29. 오후 6:49:18)
[서버] 연결 기다림
[서버] 연결됨 from 127.0.0.1:57271
hello world
안녕하세요
bye~
[서버] 클라이언트로 부터 연결종료
```

2강

TCP 소켓 프로그래밍 I

3. TCP 클라이언트

- 1. TCP 소켓 프로그래밍 기본
- 2. TCP 서버
- 3. TCP 클라이언트

1. TCP Client 작성을 위한 기본 코드

□ 클라이언트

1. TCP 소켓 객체를 생성

```
Socket socket = new Socket();
```

2. 서버와 연결

```
socket.connect( new InetSocketAddress( "localhost", 10001 ) );
```

3. 연결이 되면 통신을 위한 **stream** 객체를 얻는다.

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

4. 데이터 통신을 한다.

```
is.read( buffer );  
os.write( buffer );
```

1. TCP Client 작성을 위한 기본 코드

□ 클라이언트

5. 서버와 연결을 끊는다.

```
is.close();  
os.close();  
Socket.close();
```

2. TCP Client - 연결하기

[실습 3-1]

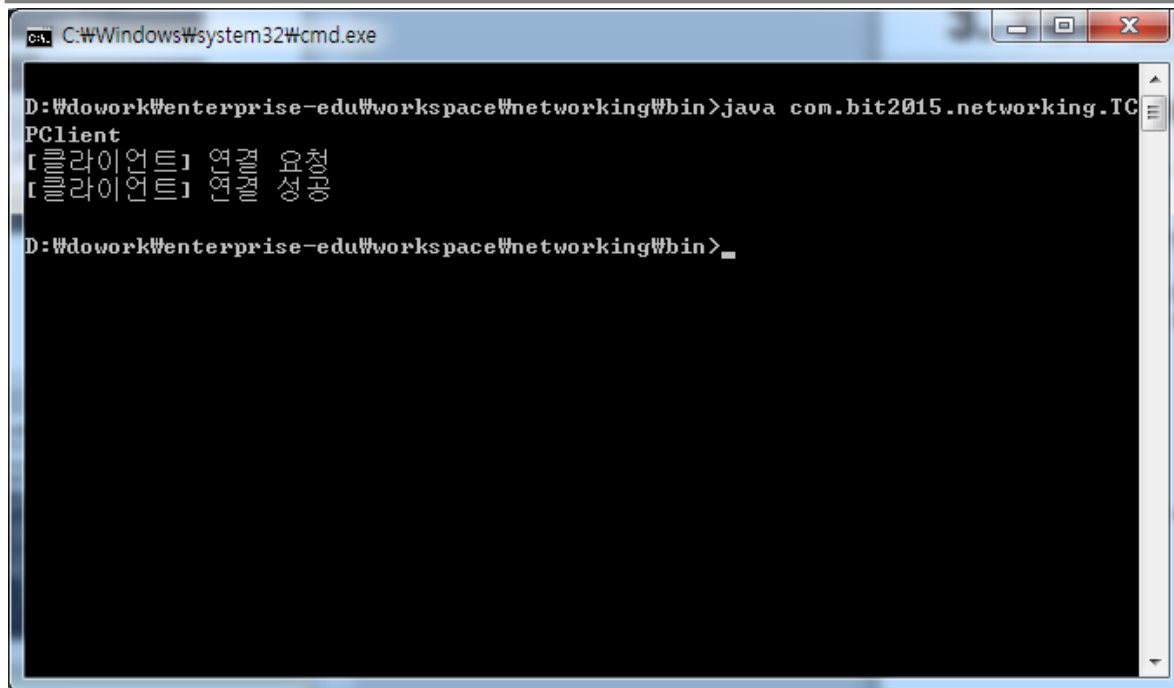
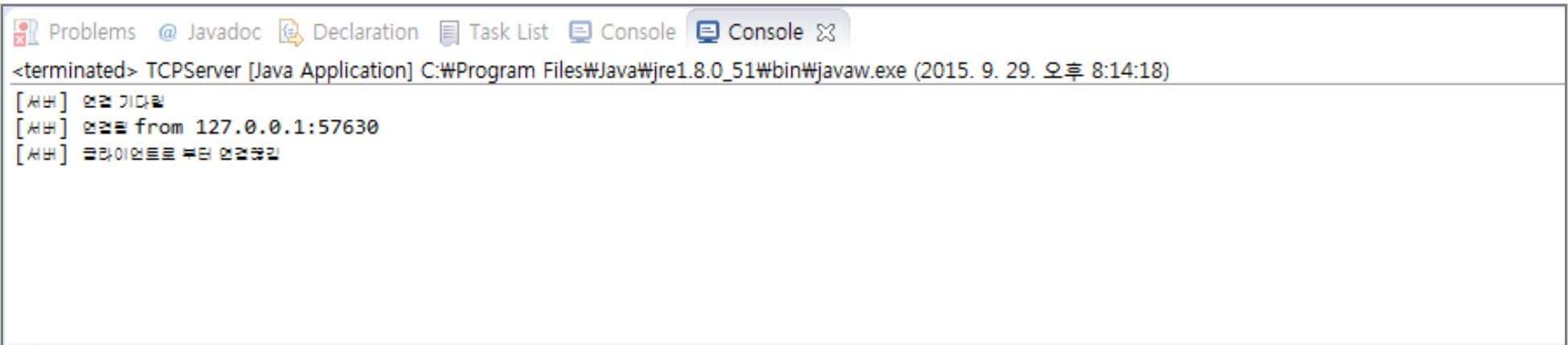
TCP 통신 클라이언트의 기본 코드를 작성하고 몇 가지 테스트를 하는 실습이다.

다음 코드를 참고해서 코드를 작성하고 실행 시켜보자.

```
01 public class TCPClient {
02     public static void main(String[] args) {
03         Socket socket = null ;
04         try {
05             socket = new Socket();
06             System.out.println( "[클라이언트] 연결 요청" );
07             socket.connect( new InetSocketAddress( "localhost", 10001 ) );
08             System.out.println( "[클라이언트] 연결 성공" );
09
10         } catch (IOException e) {
11             e.printStackTrace();
12         } finally {
13             if( socket != null && socket.isClosed() == false ) {
14                 try {
15                     socket.close();
16                 } catch( IOException ex ) {
17                     ex.printStackTrace();
18                 }
19             }
20         }
21     }
22 }
23 }
```

2. TCP Client - 연결하기

[실습 3-1]

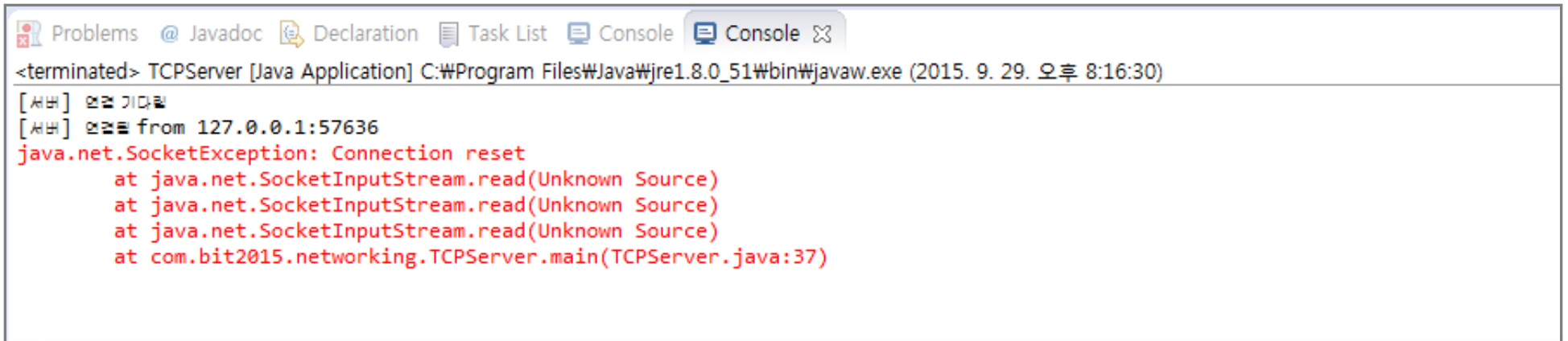


3. TCP Client – 비정상 종료하기

□ [실습 3-2]

TCP 통신 클라이언트에서 정상적으로 `socket.close()` 를 하지 않았을 때 서버의 예외를 확인하고 수정한다.

```
// try {  
//     if( socket != null && socket.isClosed() == false ) {  
//         socket.close();  
//     }  
// } catch( IOException ex ) {  
//     ex.printStackTrace();  
// }
```



The screenshot shows an IDE's console window with the following content:

```
<terminated> TCPServer [Java Application] C:\Program Files\Java\jre1.8.0_51\bin\javaw.exe (2015. 9. 29. 오후 8:16:30)  
[서버] 연결 기다림  
[서버] 연결 from 127.0.0.1:57636  
java.net.SocketException: Connection reset  
    at java.net.SocketInputStream.read(Unknown Source)  
    at java.net.SocketInputStream.read(Unknown Source)  
    at java.net.SocketInputStream.read(Unknown Source)  
    at com.bit2015.networking.TCPServer.main(TCPServer.java:37)
```

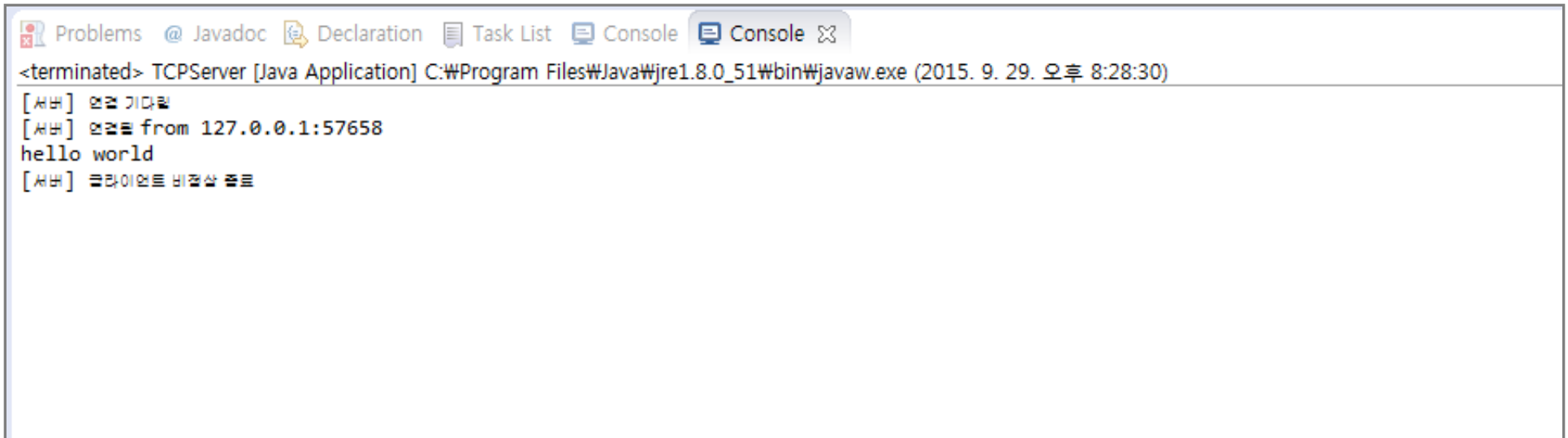
4. TCP Client – 데이터 쓰기

□ [실습 3-3]

TCP 통신 클라이언트에서 “hello world\n”를 서버로 송신하는 실습

```
// 데이터 쓰기
OutputStream os = socket.getOutputStream();

String data = "hello world\n";
os.write( data.getBytes( "UTF-8" ) );
os.close();
```



```
Problems @ Javadoc Declaration Task List Console Console
<terminated> TCPServer [Java Application] C:\Program Files\Java\jre1.8.0_51\bin\javaw.exe (2015. 9. 29. 오후 8:28:30)
[서버] 연결 기다림
[서버] 연결됨 from 127.0.0.1:57658
hello world
[서버] 클라이언트 연결상 종료
```

5. TCP Client – 데이터 쓰고/받기

[실습3-4]

TCP통신 클라이언트에서 서버로 부터 데이터 받기를 추가하고 보낸 데이터를 다시 받아 출력하는 실습

```
// 데이터 쓰고 받기
InputStream is = socket.getInputStream();
OutputStream os = socket.getOutputStream();

String data = "hello world\n";
os.write( data.getBytes( "UTF-8" ) );

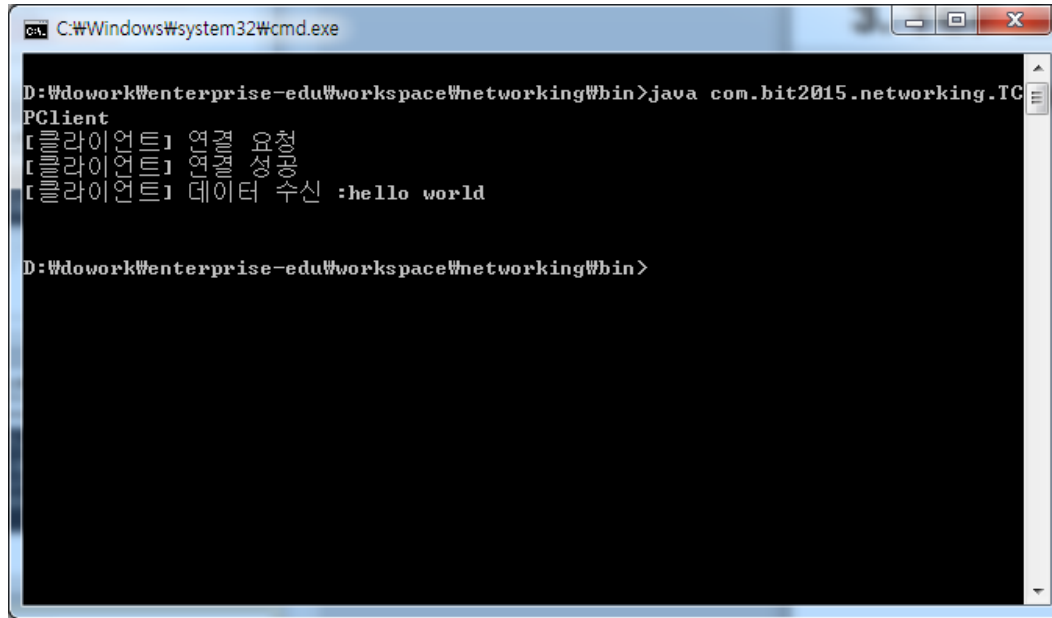
byte[] buffer = new byte[128];
int readByteCount = is.read( buffer );

data = new String( buffer, 0, readByteCount, "UTF-8" );
System.out.println( "[클라이언트] 데이터 수신 :" + data );

is.close();
os.close();
```

5. TCP Client – 데이터 쓰고/받기

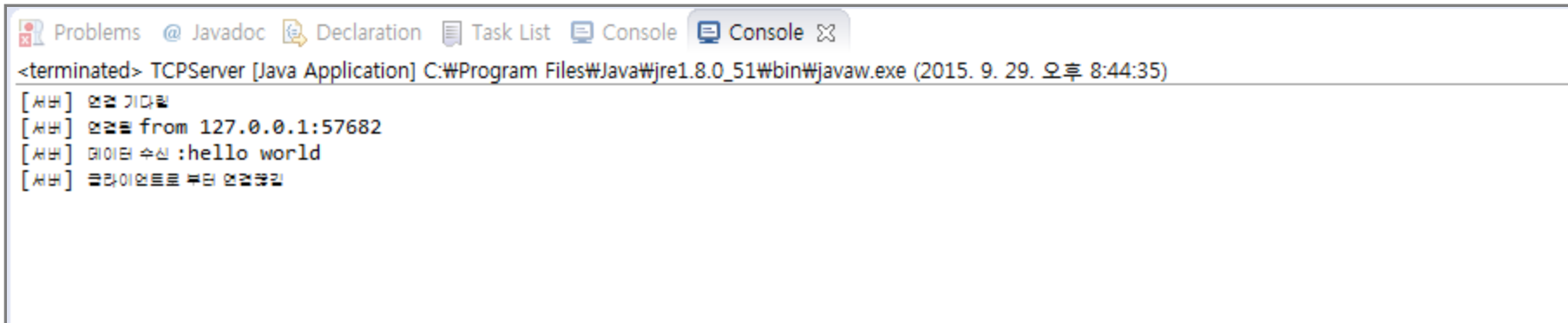
[실습 3-4]



```
C:\Windows\system32\cmd.exe

D:\work\workspace\networking\bin>java com.bit2015.networking.TC
PCClient
[클라이언트] 연결 요청
[클라이언트] 연결 성공
[클라이언트] 데이터 수신 :hello world

D:\work\workspace\networking\bin>
```



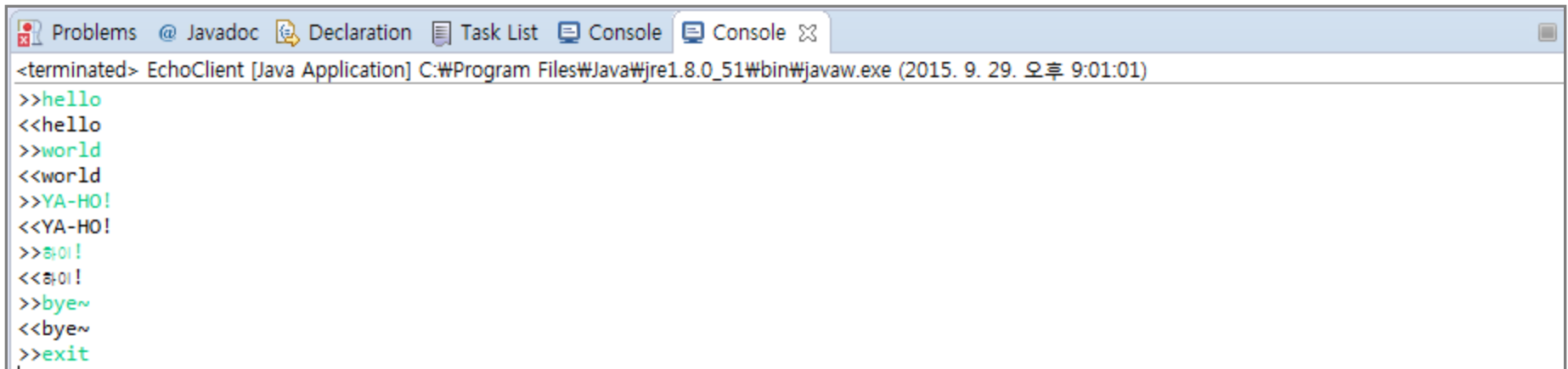
```
Problems @ Javadoc Declaration Task List Console Console X
<terminated> TCPServer [Java Application] C:\Program Files\Java\jre1.8.0_51\bin\javaw.exe (2015. 9. 29. 오후 8:44:35)
[서버] 연결 기다림
[서버] 연결됨 from 127.0.0.1:57682
[서버] 데이터 수신 :hello world
[서버] 클라이언트로 부터 연결끊김
```


6. 과제

EchoClient 작성하기

1. 아래 화면과 같은 에코 클라이언트 프로그램을 작성하기.
2. **Scanner**를 통해 사용자로 부터 입력 받은 문자열을 서버에 송신한다.
3. 서버로 부터 다시 수신받은 문자열을 화면에 출력한다.
4. **exit** 가 입력되면 프로그램을 종료한다.

<클라이언트 실행화면>

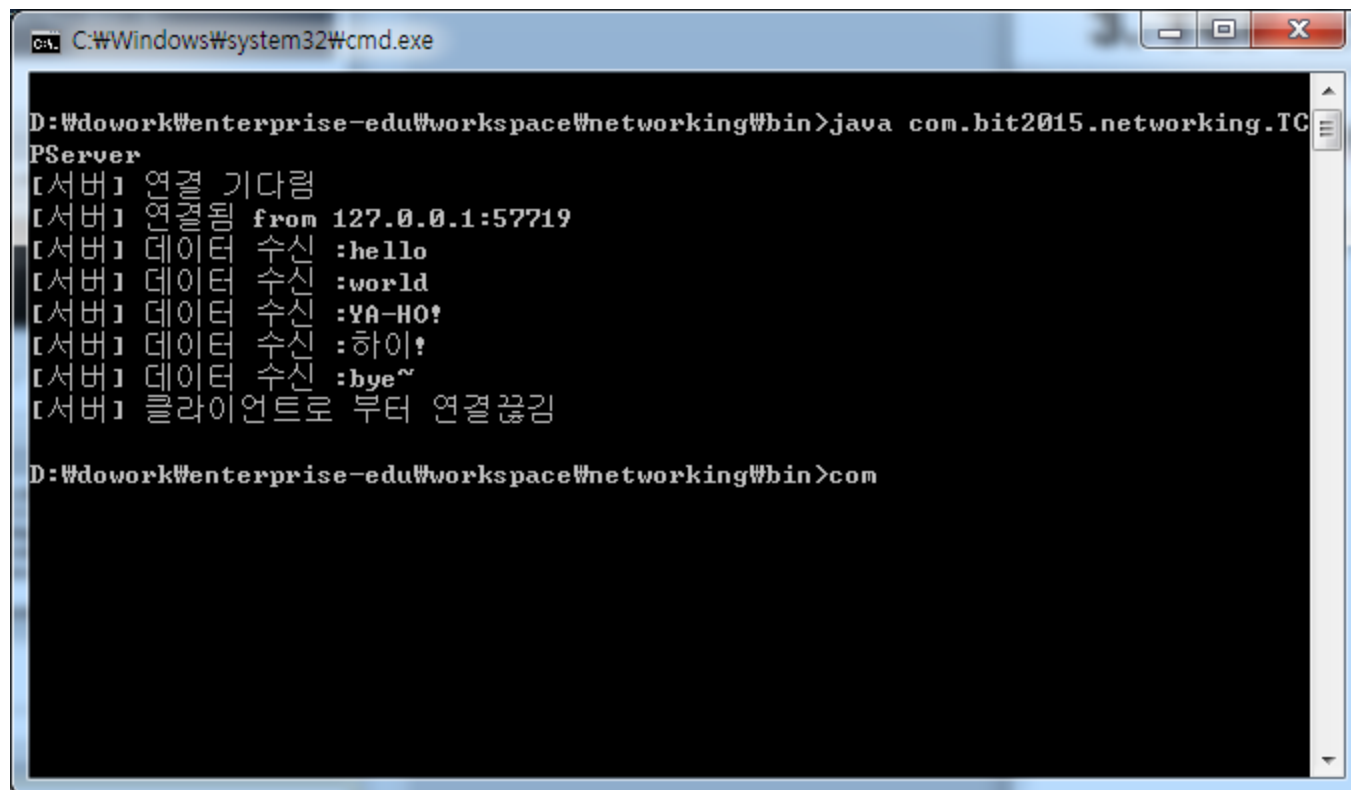


```
<terminated> EchoClient [Java Application] C:\Program Files\Java\jre1.8.0_51\bin\javaw.exe (2015. 9. 29. 오후 9:01:01)
>>hello
<<hello
>>world
<<world
>>YA-HO!
<<YA-HO!
>>하이!
<<하이!
>>bye~
<<bye~
>>exit
```

6. 과제

EchoClient 작성하기

<서버 실행화면>



```
C:\Windows\system32\cmd.exe

D:\work\enterprise-edu\workspace\networking\bin>java com.bit2015.networking.TCP
Server
[서버] 연결 기다림
[서버] 연결됨 from 127.0.0.1:57719
[서버] 데이터 수신 :hello
[서버] 데이터 수신 :world
[서버] 데이터 수신 :YA-HO!
[서버] 데이터 수신 :하이!
[서버] 데이터 수신 :bye~
[서버] 클라이언트로 부터 연결 끊김

D:\work\enterprise-edu\workspace\networking\bin>com
```

7. TCPServer의 문제점

- ❑ 소켓을 이해하기 위한 단순한 예제
- ❑ 동시접속(다중처리)을 처리 못함