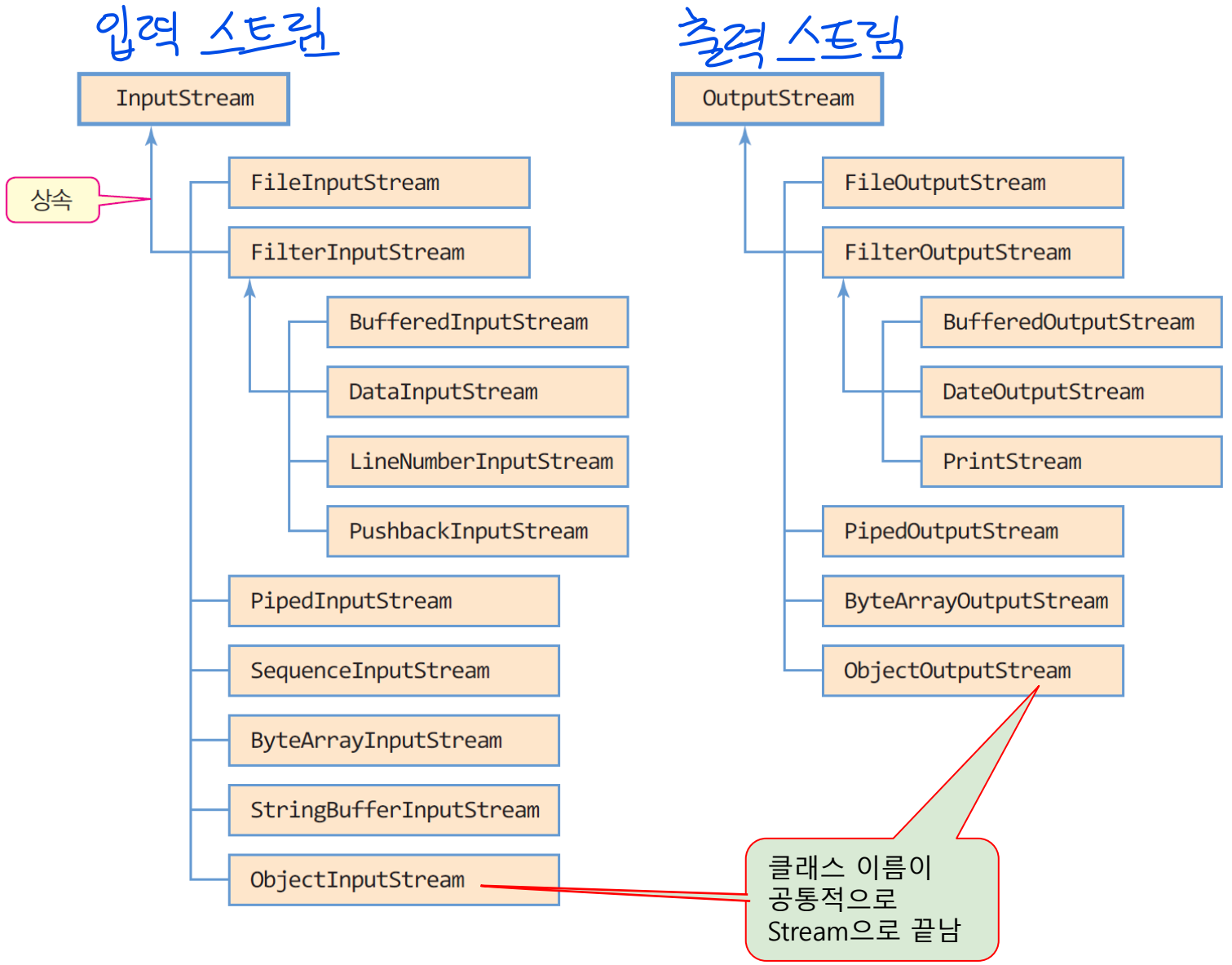
자바의 입출력 스트림 정리

한성대학교 컴퓨터공학부

신 성

JDK의 바이트 스트림 클래스 계층 구조

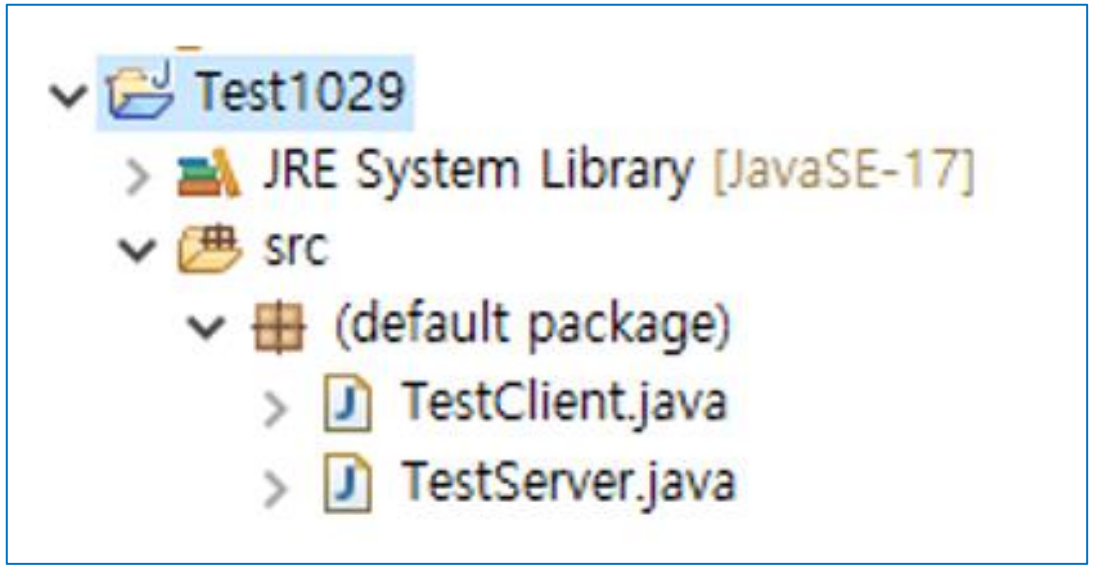


실습 준비

심플한 실습 코드

- 이클립스에서 새로운 프로젝트 생성 후

  TestServer.java와 TestClient.java 소스파일 생성



서버

```
서버입니다. 클라이언트를 기다립니다.
클라이언트와 연결되었습니다.
받은 메시지: 안녕하세요.
보낼 메시지 입력 >> 네 안녕하세요.
받은 메시지: 반갑습니다.
보낼 메시지 입력 >> 끝
연결을 종료합니다.
```

클라이언트

```
클라이언트입니다. 서버에 접속하였습니다.
보낼 메시지 입력 >> 안녕하세요.
받은 메시지: 네 안녕하세요.
보낼 메시지 입력 >> 반갑습니다.
받은 메시지: 끝
연결을 종료합니다.
```

- **서버/클라이언트 구현**

  ☞ 연결 후 클라이언트에서 먼저 메시지 보냄 ☞ 이후 순차적으로 메시지를 주고 받음(에코 서버/클라이언트)

  (서버와 클라이언트 중 어디서든 “끝”을 입력하면 서버와 클라이언트 모두 동시에 종료)

TestServer.java

(e-class에 올려드린 실습 파일)

서버

```
서버입니다. 클라이언트를 기다립니다.
클라이언트와 연결되었습니다.
받은 메시지: 안녕하세요.
보낼 메시지 입력 >> 네 안녕하세요.
받은 메시지: 반갑습니다.
보낼 메시지 입력 >> 끝
연결을 종료합니다.
```

클라이언트

```
클라이언트입니다. 서버에 접속하였습니다.
보낼 메시지 입력 >> 안녕하세요.
받은 메시지: 네 안녕하세요.
보낼 메시지 입력 >> 반갑습니다.
받은 메시지: 끝
연결을 종료합니다.
```

```java
//서버 구현, 연결 후 클라이언트에서 먼저 메시지 보냄, 이후 순차적으로 한 번씩 메시지를 주고 받음(에코 서버/클라이언트)
//서버와 클라이언트 중 어디서든 "끝"을 입력하면 서버와 클라이언트 모두 동시에 종료

import java.io.BufferedReader;

//소스 코드를 입력하고 Ctrl+Shift+O를 눌러서 필요한 파일을 포함
public class TestServer {
    public static void main(String[] args) {
        try {

            ServerSocket listener = new ServerSocket(9999); // 서버 소켓 생성

            System.out.println("서버입니다. 클라이언트를 기다립니다.");
            Socket socket = listener.accept(); // 클라이언트로부터 연결 요청 대기
            System.out.println("클라이언트와 연결되었습니다.");

            BufferedReader in = new BufferedReader(new InputStreamReader(socket.getInputStream()));
            BufferedWriter out = new BufferedWriter(new OutputStreamWriter(socket.getOutputStream()));

            Scanner scanner = new Scanner(System.in);

            String inputMessage;
            String outputMessage;

            while (true) {
                inputMessage = in.readLine(); // 클라이언트로부터 한 행의 텍스트 받음
                System.out.println("받은 메시지: " + inputMessage); // 클라이언트가 보낸 메시지 화면에 출력
                if (inputMessage.equals("끝")) { // 클라이언트가 "끝"을 보내면 연결 종료, 참고: equalsIgnoreCase()는 대소문자를 구분하지 않고 비교
                    System.out.println("접속을 종료합니다.");
                    break;
                }

                System.out.print("보낼 메시지 입력 >> ");
                outputMessage = scanner.nextLine(); // 키보드에서 한 행의 문자열 받음
                out.write(outputMessage+"\n"); // 클라이언트로 보냄
                out.flush();
                if (outputMessage.equals("끝")) { // "끝"이 입력되면 연결 종료
                    System.out.println("연결을 종료합니다.");
                    break;
                }
            }

            scanner.close();
            socket.close();
            listener.close();

        } catch (IOException e) {
            System.out.println("오류가 발생했습니다.");
        }
    }
}
```

TestClient.java

(e-class에 올려드린 실습 파일)

서버

```
서버입니다. 클라이언트를 기다립니다.
클라이언트와 연결되었습니다.
받은 메시지: 안녕하세요.
보낼 메시지 입력 >> 네 안녕하세요.
받은 메시지: 반갑습니다.
보낼 메시지 입력 >> 끝
연결을 종료합니다.
```

클라이언트

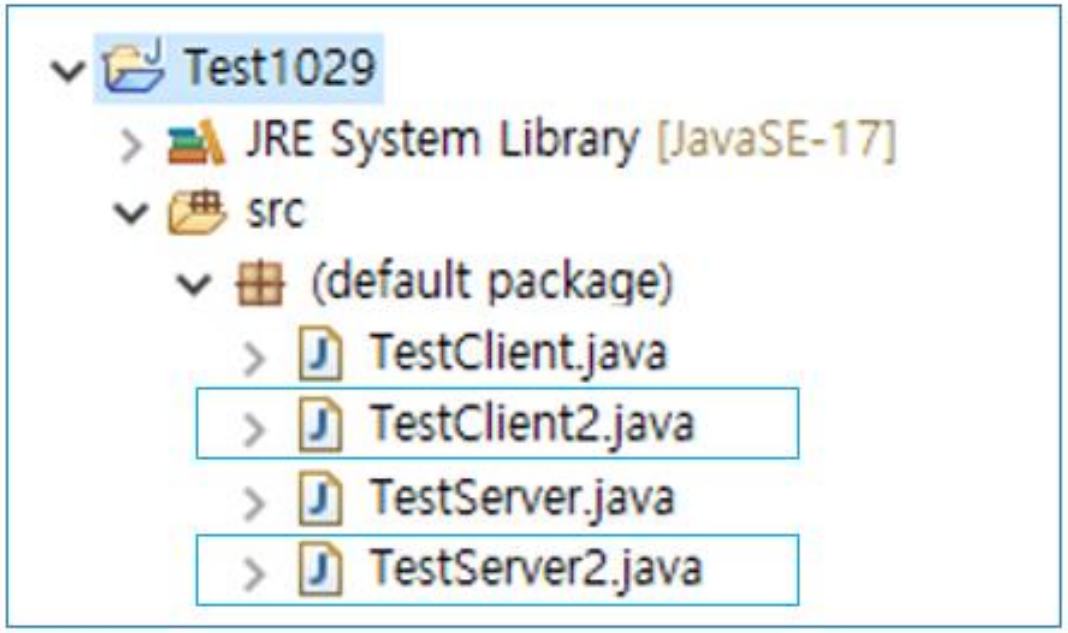
```
클라이언트입니다. 서버에 접속하였습니다.
보낼 메시지 입력 >> 안녕하세요.
받은 메시지: 네 안녕하세요.
보낼 메시지 입력 >> 반갑습니다.
받은 메시지: 끝
연결을 종료합니다.
```

```java
//클라이언트 구현, 연결 후 클라이언트에서 먼저 메시지 보냄, 이후 순차적으로 한 번씩 메시지를 주고 받음(에코 서버/클라이언트)
//서버와 클라이언트 중 어디서든 "끝"을 입력하면 서버와 클라이언트 모두 동시에 종료

import java.io.BufferedReader;

//소스 코드를 입력하고 Ctrl+Shift+O를 눌러서 필요한 파일을 포함
public class TestClient {
    public static void main(String[] args) {

        try {
            Socket socket = new Socket("localhost", 9999); // 클라이언트 소켓 생성. 서버에 바로 접속
            System.out.println("클라이언트입니다. 서버에 접속하였습니다.");

            BufferedReader in = new BufferedReader(new InputStreamReader(socket.getInputStream()));
            BufferedWriter out = new BufferedWriter(new OutputStreamWriter(socket.getOutputStream()));

            Scanner scanner = new Scanner(System.in);

            String inputMessage;
            String outputMessage;

            while (true) {

                System.out.print("보낼 메시지 입력 >> ");
                outputMessage = scanner.nextLine(); // 키보드에서 한 행의 문자열 읽음
                out.write(outputMessage+"\n"); // 서버로 보냄
                out.flush();
                if (outputMessage.equals("끝")) { // "끝"이 입력되면 연결 종료
                    System.out.println("연결을 종료합니다.");
                    break;
                }

                inputMessage = in.readLine(); // 서버로부터 한 행의 텍스트 받음
                System.out.println("받은 메시지: " + inputMessage); // 서버가 보낸 메시지 화면에 출력
                if (inputMessage.equals("끝")) { // 서버가 "끝"을 보내면 연결 종료
                    System.out.println("연결을 종료합니다.");
                    break;
                }
            }

            scanner.close();
            socket.close();

        } catch (IOException e) {
            System.out.println("오류가 발생했습니다.");
        }
    }

}
```

심플한 실습 코드 2

- **TestServer2.java와 TestClient2.java 소스파일 생성**

  (이클립스에서 기존 소스파일을 복사(ctrl+c)해서 붙여 넣으면(ctrl+v) 편하게 생성 가능, 다음 페이지)



서버

```
서버입니다. 클라이언트를 기다립니다.
클라이언트와 연결되었습니다.
보낼 메시지 입력 >> 안녕하세요.
보낼 메시지 입력 >> 반갑습니다.
보낼 메시지 입력 >> 그럼 또 만나요.
보낼 메시지 입력 >> 끝
연결을 종료합니다.
```

클라이언트

```
클라이언트입니다. 서버에 접속하였습니다.
받은 메시지: 안녕하세요.
받은 메시지: 반갑습니다.
받은 메시지: 그럼 또 만나요.
받은 메시지: 끝
연결을 종료합니다.
```

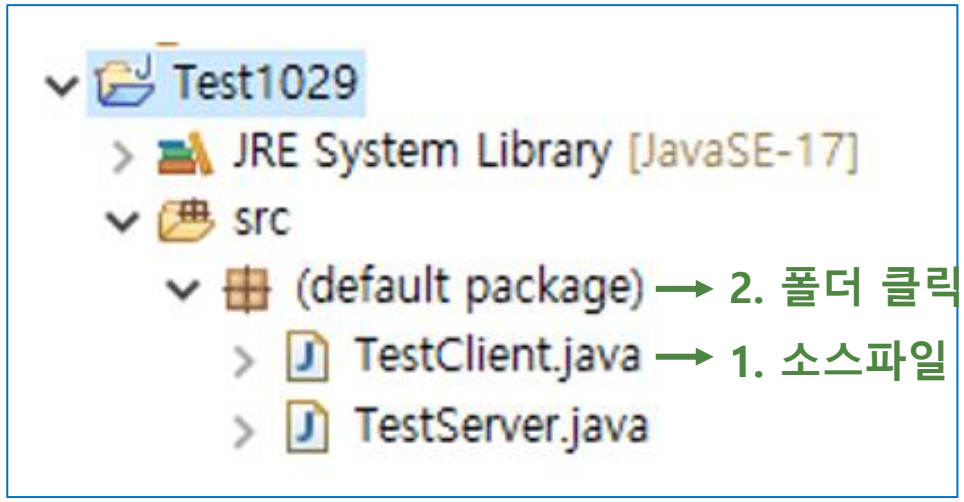
- 서버/클라이언트 구현 - 쉬운 이해를 위해 좀 더 간결한 코드 작성

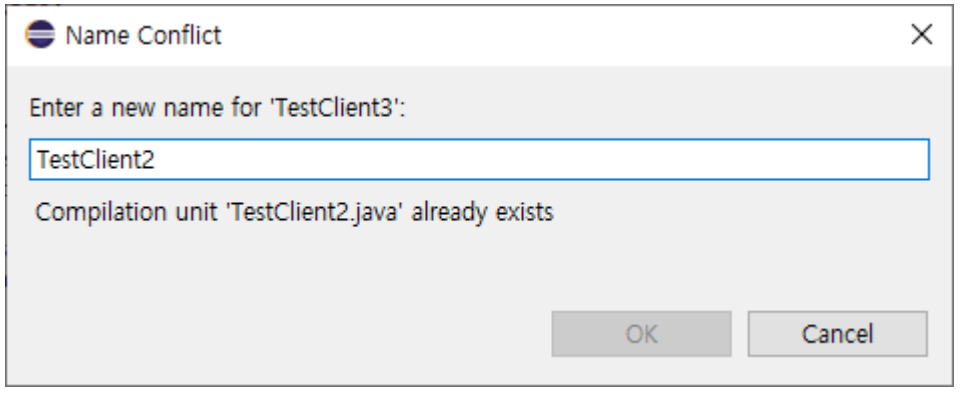
☞ 서버는 메시지를 보내기만 하도록 구현, 클라이언트는 메시지를 받기만 하도록 구현
(서버에서 "끝"을 보내면 서버와 클라이언트 모두 연결 종료)

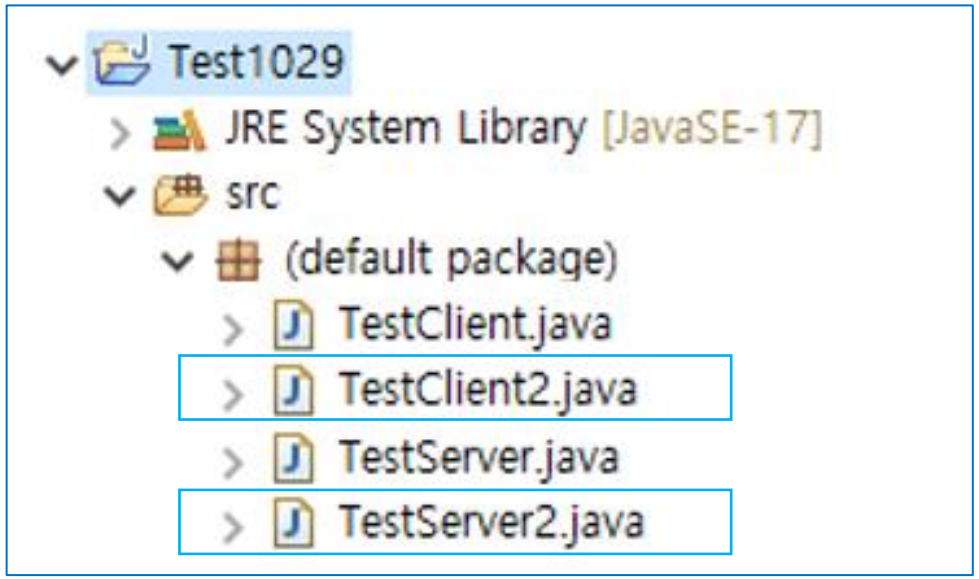
[참고]

- **TestServer2.java와 TestClient2.java 소스파일 생성**

  (이클립스에서 기존 소스파일을 복사(ctrl+c)해서 붙여 넣으면(ctrl+v) 편하게 생성 가능)







TestServer2.java

(e-class에 올려드린 실습 파일)

서버

```
서버입니다. 클라이언트를 기다립니다.
클라이언트와 연결되었습니다.
보낼 메시지 입력 >> 안녕하세요.
보낼 메시지 입력 >> 반갑습니다.
보낼 메시지 입력 >> 그럼 또 만나요.
보낼 메시지 입력 >> 끝
연결을 종료합니다.
```

클라이언트

```
클라이언트입니다. 서버에 접속하였습니다.
받은 메시지: 안녕하세요.
받은 메시지: 반갑습니다.
받은 메시지: 그럼 또 만나요.
받은 메시지: 끝
연결을 종료합니다.
```

```java
//서버는 메시지를 보내기만 하도록 구현, 서버에서 "끝"을 보내면 모두 연결 종료
//앞의 코드를 다음과 같이 수정, 이후 Ctrl+Shift+O 눌러서 import 파일 정리

import java.io.BufferedWriter;

public class TestServer2 {
    public static void main(String[] args) {
        try {

            ServerSocket listener = new ServerSocket(9999); // 서버 소켓 생성

            System.out.println("서버입니다. 클라이언트를 기다립니다.");
            Socket socket = listener.accept(); // 클라이언트로부터 연결 요청 대기
            System.out.println("클라이언트와 연결되었습니다.");

            BufferedWriter out = new BufferedWriter(new OutputStreamWriter(socket.getOutputStream()));

            Scanner scanner = new Scanner(System.in);

            String outputMessage;

            while (true) {

                System.out.print("보낼 메시지 입력 >> ");
                outputMessage = scanner.nextLine(); // 키보드에서 한 행의 문자열 읽음
                out.write(outputMessage+"\n"); // 클라이언트로 보냄
                out.flush();
                if (outputMessage.equals("끝")) { // "끝"이 입력되면 서버와 연결 종료
                    System.out.println("연결을 종료합니다.");
                    break;
                }
            }

            scanner.close();
            socket.close();
            listener.close();

        } catch (IOException e) {
            System.out.println("오류가 발생했습니다.");
        }
    }
}
```

TestClient2.java

(e-class에 올려드린 실습 파일)

서버

```
서버입니다. 클라이언트를 기다립니다.
클라이언트와 연결되었습니다.
보낼 메시지 입력 >> 안녕하세요.
보낼 메시지 입력 >> 반갑습니다.
보낼 메시지 입력 >> 그럼 또 만나요.
보낼 메시지 입력 >> 끝
연결을 종료합니다.
```

클라이언트

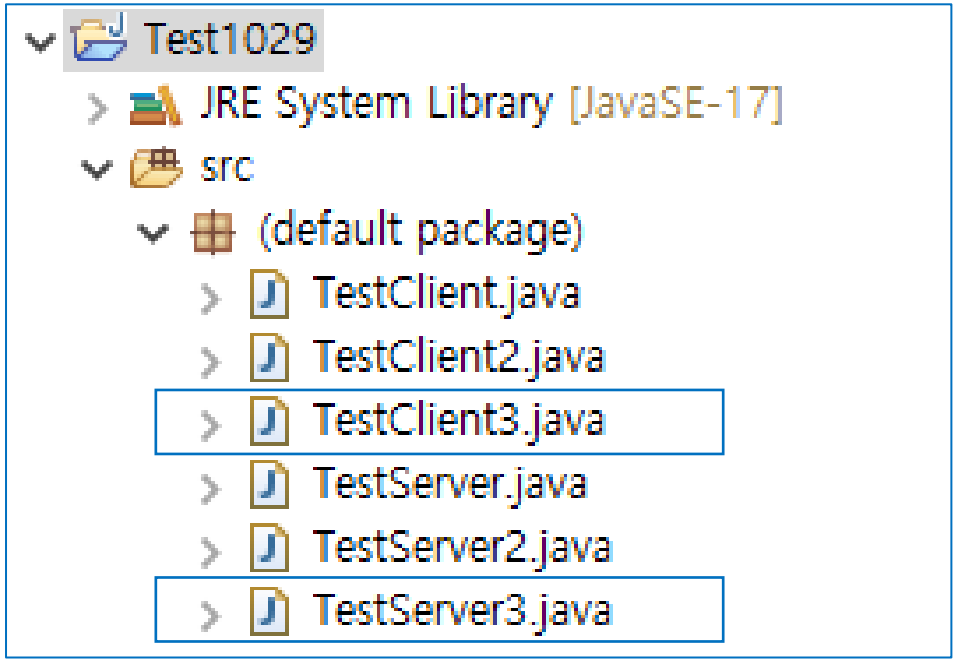
```
클라이언트입니다. 서버에 접속하였습니다.
받은 메시지: 안녕하세요.
받은 메시지: 반갑습니다.
받은 메시지: 그럼 또 만나요.
받은 메시지: 끝
연결을 종료합니다.
```

```java
//클라이언트는 메시지를 받기만 하도록 구현
//앞의 코드를 다음과 같이 수정, 이후 Ctrl+Shift+O 눌러서 import 파일 정리

import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.net.Socket;
import java.util.Scanner;

public class TestClient2 {
    public static void main(String[] args) {

        try {
            Socket socket = new Socket("localhost", 9999); // 클라이언트 소켓 생성. 서버에 바로 접속
            System.out.println("클라이언트입니다. 서버에 접속하였습니다.");

            BufferedReader in = new BufferedReader(new InputStreamReader(socket.getInputStream()));

            Scanner scanner = new Scanner(System.in);

            String inputMessage;

            while (true) {

                inputMessage = in.readLine(); // 서버로부터 한 행의 텍스트 받음
                System.out.println("받은 메시지: " + inputMessage); // 서버가 보낸 메시지 화면에 출력
                if (inputMessage.equals("끝")) { // 서버가 "끝"을 보내면 연결 종료
                    System.out.println("연결을 종료합니다.");
                    break;
                }
            }

            scanner.close();
            socket.close();

        } catch (IOException e) {
            System.out.println("오류가 발생했습니다.");
        }
    }

}
```

심플한 실습 코드 3

- **TestServer3.java와 TestClient3.java 소스파일 생성**

  (앞의 코드와 동일하게 이클립스에서 기존 소스파일을 복사(ctrl+c)해서 붙여 넣으면(ctrl+v) 편하게 생성 가능)



서버

클라이언트

```
받은 메시지: 서버 연결 완료
```

- **서버/클라이언트 구현 - 쉬운 이해를 위해 매우 간결한 코드 작성(가장 심플하게 구현)**

  ☞ 서버는 (클라이언트와 연결이 되면) "서버 연결 완료" 메시지를 한 번만 자동으로 보내고 연결 종료

  클라이언트는 (서버와 연결 후) 서버의 메시지를 한 번만 받아서 출력하고 종료

TestServer3.java

(e-class에 올려드린 실습 파일)

```java
//(클라이언트와 연결이 되면) "서버 연결 완료" 메시지를 한 번만 자동으로 보내고 종료(가장 심플하게 구현)
//앞의 코드를 다음과 같이 수정, 이후 Ctrl+Shift+O 눌러서 import 파일 정리

import java.io.BufferedWriter;..

public class TestServer3 {
    public static void main(String[] args) {
        try {

            ServerSocket listener = new ServerSocket(9999); // 서버 소켓 생성
            Socket socket = listener.accept(); // 클라이언트로부터 연결 요청 대기

            BufferedWriter out = new BufferedWriter(new OutputStreamWriter(socket.getOutputStream()));
            out.write("서버 연결 완료"+"\n"); // 연결이 되면 클라이언트로 "서버 연결 완료" 메시지를 보냄
            out.flush();


            socket.close();
            listener.close();

        } catch (IOException e) {
            System.out.println("오류가 발생했습니다.");
        }
    }
}
```

서버

클라이언트

받은 메시지: 서버 연결 완료

TestClient3.java

(e-class에 올려드린 실습 파일)

```java
//(서버와 연결 후) 서버의 메시지를 한 번만 받아서 출력하고 종료(가장 심플하게 구현)
//앞의 코드를 다음과 같이 수정, 이후 Ctrl+Shift+O 눌러서 import 파일 정리

import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.net.Socket;

public class TestClient3 {
    public static void main(String[] args) {

        try {
            Socket socket = new Socket("localhost", 9999); // 클라이언트 소켓 생성. 서버에 바로 접속


            BufferedReader in = new BufferedReader(new InputStreamReader(socket.getInputStream()));

            String inputMessage = in.readLine(); // 서버로부터 한 행의 텍스트 받음
            System.out.println("받은 메시지: " + inputMessage); // 서버가 보낸 메시지 화면에 출력


            socket.close();

        } catch (IOException e) {
            System.out.println("오류가 발생했습니다.");
        }
    }
}
```

서버

클라이언트

받은 메시지: 서버 연결 완료

입출력 스트림

입출력 스트림

- (지금부터는 우리가 계속 사용하던) 다음 부분에 대한 자세한 설명

```
BufferedReader in = new BufferedReader(new InputStreamReader(socket.getInputStream()));
BufferedWriter out = new BufferedWriter(new OutputStreamWriter(socket.getOutputStream()));
```

- 위의 코드는 다음과 동일한 코드(풀어서 쓰면 다음과 같음)

```
InputStream is = socket.getInputStream();
InputStreamReader ir = new InputStreamReader(is);
BufferedReader in = new BufferedReader(ir);

OutputStream os = socket.getOutputStream();
OutputStreamWriter ow = new OutputStreamWriter(os);
BufferedWriter out = new BufferedWriter(ow);
```

☞ *이제부터는 서버(TestServer3.java)는 그대로 놓고 데이터를 받는 클라이언트만 수정해가면서 입력 스트림에 대해 살펴보도록 하겠습니다.*

☞ (앞서 설명한바와 같이) 다음과 같이 풀어서 작성해도 동일하게 동작

TestClient31.java

```java
//(서버와 연결 후) 서버의 메시지를 한 번만 받아서 출력하고 종료(가장 심플하게 구현)
//앞의 코드를 다음과 같이 수정, 이후 Ctrl+Shift+O 눌러서 import 파일 정리

import java.io.BufferedReader;

public class TestClient31 {
    public static void main(String[] args) {

        try {
            Socket socket = new Socket("localhost", 9999); // 클라이언트 소켓 생성. 서버에 바로 접속

            InputStream is = socket.getInputStream();
            InputStreamReader ir = new InputStreamReader(is);
            BufferedReader in = new BufferedReader(ir);

            String inputMessage = in.readLine(); // 서버로부터 한 행의 텍스트 받음
            System.out.println("받은 메시지: " + inputMessage); // 서버가 보낸 메시지 화면에 출력


            socket.close();

        } catch (IOException e) {
            System.out.println("오류가 발생했습니다.");
        }
    }
}
```

서버

클라이언트

```
받은 메시지: 서버 연결 완료
```

결과는 동일

InputStream 클래스

- 첫 부분 socket.getInputStream()함수가 만들어 주는 객체는 '*InputStream 클래스의 객체*'
  - 바이트 단위 입력, 1 바이트씩 읽어들임, read()
  - 우리가 계속 사용하던 *System.in*도 *InputStream 클래스의 객체*

※ `Scanner scanner = new Scanner(System.in);`

- *InputStream 클래스의 주요 메소드*

반환형	메소드	설명
int	read()	입력 스트림으로부터 1바이트를 읽고, 읽은 바이트를 반환한다.
int	read(byte[] b)	입력 스트림으로부터 읽은 바이트들을 매개값으로 주어진 바이트 배열 b에 저장하고 실제로 읽은 바이트 수를 반환
void	close()	사용한 시스템 자원을 반납하고, 입력 스트림을 닫는다.

[참고] InputStream 클래스 직접 활용(*소켓 입력 스트림*)

- 다음과 같이 1바이트씩 읽어들여서 배열에 저장 후 직접 문자열로 만들어야 함(번거로움)

TestClient32.java

```java
import java.io.IOException;

public class TestClient32 {
    public static void main(String[] args) {

        try {
            Socket socket = new Socket("localhost", 9999); // 클라이언트 소켓 생성. 서버에 바로 접속

            InputStream is = socket.getInputStream();

            byte[] byteData = new byte[1024];
            int bytesRead = is.read(byteData);

            String inputMessage = new String(byteData, 0, bytesRead);
            System.out.println("받은 메시지: " + inputMessage);


            socket.close();

        } catch (IOException e) {
            System.out.println("오류가 발생했습니다.");
        }
    }
}
```

결과는 동일

서버

클라이언트

```
받은 메시지: 서버 연결 완료
```

[참고] InputStream 클래스 직접 활용(*system.in*)

- System 클래스의 static 맴버인 system.in 참조변수가 참조하는 객체도 방금 전과 똑같이 사용 가능

TestKeyboard.java

☞ *키보드로부터 데이터를 읽어들이는 코드*

```java
import java.io.IOException;

public class TestKeyboard {
    public static void main(String[] args) {

        try {
            InputStream is = System.in;

            System.out.print("텍스트 입력: ");

            byte[] byteData = new byte[1024];
            int bytesRead = is.read(byteData);

            String inputMessage = new String(byteData, 0, bytesRead);
            System.out.print(inputMessage);

        } catch (IOException e) {
            System.out.println("오류가 발생했습니다.");
        }
    }
}
```

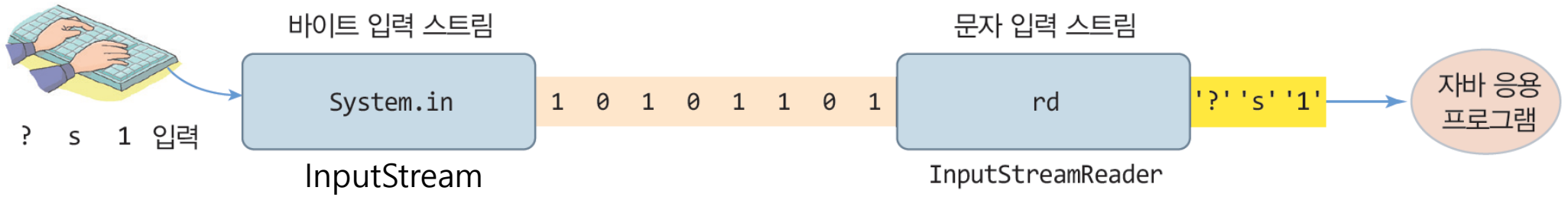
결과는 동일

```
텍스트 입력: 안녕하세요.
안녕하세요.
```

InputStreamReader (문자 변환 보조 스트림)

- 문자 단위 입력이 가능
- 보조 스트림 : 다른 스트림과 연결되어 여러 가지 편리한 기능을 제공해주는 스트림
  자체적으로 입출력을 수행할 수 없기 때문에 입력 소스와 직접 연결되는
  InputStream 등에 연결해서 사용

* 표준 입력 스트림 System.in에 InputStreamReader 스트림을 연결한 사례



InputStreamReader (문자 변환 보조 스트림)

- **InputStreamReader의 주요 메소드**

반환형	메소드	설명
int	read()	한 개의 문자를 읽어서 정수형으로 반환
int	read(char[] cbuf)	문자들을 읽어서 cbuf 배열에 저장하고 읽은 개수 반환
void	close()	입력 스트림을 닫고 관련된 시스템 자원 해제

* 표준 입력 스트림 System.in에 InputStreamReader 스트림을 연결한 사례

```
InputStreamReader rd = new InputStreamReader(System.in);
int c = rd.read(); // 키보드에서 문자 읽음
```

☞ ***InputStreamReader를 활용하는 경우에도 문자로 읽어들일수는 있으나 직접 연결해서 문자열을 만들어야하기 때문에 번거로움***

[참고] InputStreamReader 직접 활용(system.in)

- 키보드로부터 문자열을 읽어들이는 코드

TestKeyboard2.java

```java
import java.io.IOException;

public class TestKeyboard2 {
    public static void main(String[] args) {

        try {
            InputStream is = System.in;
            InputStreamReader rd = new InputStreamReader(is);

            System.out.print("텍스트 입력: ");

            char[] cbuf = new char[1024];
            int charRead = rd.read(cbuf);

            String inputMessage = new String(cbuf, 0, charRead);
            System.out.print(inputMessage);

        } catch (IOException e) {
            System.out.println("오류가 발생했습니다.");
        }
    }
}
```

결과는 동일

```
텍스트 입력: 안녕하세요.
안녕하세요.
```

[참고] InputStreamReader 직접 활용(소켓 입력 스트림)

- 서버로부터 전송된 문자열을 읽어들이는 코드

TestClient33.java

```java
import java.io.IOException;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.net.Socket;

public class TestClient33 {
    public static void main(String[] args) {

        try {
            Socket socket = new Socket("localhost", 9999); // 클라이언트 소켓 생성. 서버에 바로 접속

            InputStream is = socket.getInputStream();
            InputStreamReader rd = new InputStreamReader(is);

            char[] cbuf = new char[1024];
            int charRead = rd.read(cbuf);

            String inputMessage = new String(cbuf, 0, charRead);
            System.out.println("받은 메시지: " + inputMessage);

            socket.close();

        } catch (IOException e) {
            System.out.println("오류가 발생했습니다.");
        }
    }
}
```

결과는 동일

서버

클라이언트

```
받은 메시지: 서버 연결 완료
```

BufferedReader (성능 향상 보조 스트림)

- 우리가 그동안 사용하던 형태
- 추가적으로 버퍼를 사용함으로써 입출력 속도 향상
- *readLine()* 메소드를 추가적으로 가지고 있어서 라인(문자열) 단위로 읽어 올 수 있는 장점

  - 캐리지리턴(₩r) 또는 라인피트(₩n)로 구분된 행단위의 문자열을 한꺼번에 읽을 수 있음

키보드로부터 문자열을 읽어들이는 코드

```
BufferedReader in = new BufferedReader(new InputStreamReader(System.in));
String inputMessage = in.readLine();
```

서버로부터 전송된 문자열을 읽어들이는 코드

```
BufferedReader in = new BufferedReader(new InputStreamReader(socket.getInputStream()));
String inputMessage = in.readLine();
```

BufferedReader 활용(system.in)

- 키보드로부터 문자열을 읽어들이는 코드

TestKeyboard3.java

```java
import java.io.BufferedReader;

public class TestKeyboard3 {
    public static void main(String[] args) {

        try {
            BufferedReader in = new BufferedReader(new InputStreamReader(System.in));

            System.out.print("텍스트 입력: ");

            String inputMessage = in.readLine(); // 서버로부터 한 행의 텍스트 받음
            System.out.println(inputMessage); // 서버가 보낸 메시지 화면에 출력

        } catch (IOException e) {
            System.out.println("오류가 발생했습니다.");
        }
    }
}
```

결과는 동일

```
텍스트 입력: 안녕하세요.
안녕하세요.
```

BufferedReader 활용(소켓 입력 스트림)

- 서버로부터 전송된 문자열을 읽어들이는 코드

TestClient34.java

```java
import java.io.BufferedReader;

public class TestClient34 {
    public static void main(String[] args) {

        try {
            Socket socket = new Socket("localhost", 9999); // 클라이언트 소켓 생성. 서버에 바로 접속

            BufferedReader in = new BufferedReader(new InputStreamReader(socket.getInputStream()));

            String inputMessage = in.readLine(); // 서버로부터 한 행의 텍스트 받음
            System.out.println("받은 메시지: " + inputMessage); // 서버가 보낸 메시지 화면에 출력

            socket.close();

        } catch (IOException e) {
            System.out.println("오류가 발생했습니다.");
        }
    }
}
```

결과는 동일

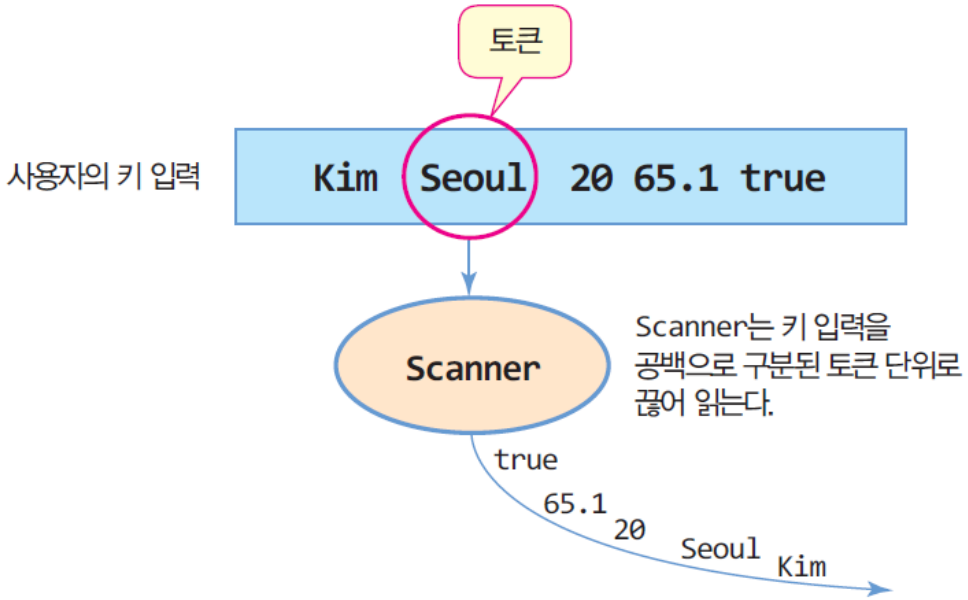
서버

클라이언트

받은 메시지: 서버 연결 완료

Scanner 클래스를 활용하는 것도 가능

- Scanner는 입력되는 키 값을 공백으로 구분되는 아이템 단위로 읽음
- 공백 문자 : '₩t', '₩f', '₩r', ' ', '₩n'



메소드	설명
String next()	다음 토큰을 문자열로 리턴
byte nextByte()	다음 토큰을 byte 타입으로 리턴
short nextShort()	다음 토큰을 short 타입으로 리턴
int nextInt()	다음 토큰을 int 타입으로 리턴
long nextLong()	다음 토큰을 long 타입으로 리턴
float nextFloat()	다음 토큰을 float 타입으로 리턴
double nextDouble()	다음 토큰을 double 타입으로 리턴
boolean nextBoolean()	다음 토큰을 boolean 타입으로 리턴
String nextLine()	'\n'을 포함하는 한 라인을 읽고 '\n'을 버린 나머지 문자열 리턴
void close()	Scanner의 사용 종료
boolean hasNext()	현재 입력된 토큰이 있으면 true, 아니면 입력 때까지 무한정 대기, 새로운 입력이 들어올 때 true 리턴. crtl-z 키가 입력되면 입력 끝이므로 false 리턴

Scanner 클래스 활용(system.in)

- 키보드로부터 문자열을 읽어들이는 코드

TestKeyboard4.java

```java
import java.util.Scanner;

public class TestKeyboard4 {
    public static void main(String[] args) {


        Scanner scanner = new Scanner(System.in);

        System.out.print("텍스트 입력: ");

        String inputMessage = scanner.nextLine(); // 서버로부터 한 행의 텍스트 받음
        System.out.println(inputMessage); // 서버가 보낸 메시지 화면에 출력

        scanner.close();

    }
}
```

결과는 동일

```
텍스트 입력: 안녕하세요.
안녕하세요.
```

Scanner 클래스 활용(소켓 입력 스트림)

- 서버로부터 전송된 문자열을 읽어들이는 코드

TestClient35.java

```java
import java.io.IOException;
import java.net.Socket;
import java.util.Scanner;

public class TestClient35 {
    public static void main(String[] args) {

        try {
            Socket socket = new Socket("localhost", 9999); // 클라이언트 소켓 생성. 서버에 바로 접속

            Scanner scanner = new Scanner(socket.getInputStream());

            String inputMessage = scanner.nextLine(); // 서버로부터 한 행의 텍스트 받음
            System.out.println("받은 메시지: " + inputMessage); // 서버가 보낸 메시

            scanner.close();
            socket.close();

        } catch (IOException e) {
            System.out.println("오류가 발생했습니다.");
        }
    }
}
```

스캐너 클래스는 다양한 유형의 데이터를 편하게 읽어들일 수 있지만 입력 데이터가 많을 경우 느려질 수 있어서 입력데이터가 많을 경우 BufferedReader를 사용하는 게 좋음

결과는 동일

서버

클라이언트

받은 메시지: 서버 연결 완료

정리

- 지금까지 InputStream, InputStreamReader, BufferedReader, Scanner 클래스에 대해 정리해 보았습니다(입력 스트림).
- 출력 스트림도 다음과 같이 동일한 구조로 사용 가능

```
BufferedReader in = new BufferedReader(new InputStreamReader(socket.getInputStream()));
BufferedWriter out = new BufferedWriter(new OutputStreamWriter(socket.getOutputStream()));
```

- 위의 코드는 다음과 동일한 코드(풀어서 쓰면 다음과 같음)

```
InputStream is = socket.getInputStream();
InputStreamReader ir = new InputStreamReader(is);
BufferedReader in = new BufferedReader(ir);

OutputStream os = socket.getOutputStream();
OutputStreamWriter ow = new OutputStreamWriter(os);
BufferedWriter out = new BufferedWriter(ow);
```

정리

- **출력 스트림 사용 예**

```
BufferedWriter out = new BufferedWriter(new OutputStreamWriter(socket.getOutputStream()));

out.write("서버 연결 완료"+"\n"); // 연결이 되면 클라이언트로 "서버 연결 완료" 메시지를 보냄

out.flush();

out.close();
```

기타 출력 스트림

PrintWriter/PrintStream (프린터 보조 스트림)

- 그동안 빈번히 사용하던 System.out 객체가 PrintStream 클래스의 객체 (System.out.println())
- PrintStream과 PrintWriter는 거의 같은 기능을 가지고 있지만 PrintWriter가 보다 다양한 언어의 문자를 처리하는데 적합하므로 (System.out 객체를 사용하는 경우가 아니라면) PrintWriter 사용 권장

☞ *PrintWriter/PrintStream 클래스에서 제공하는 주요 메소드(다음과 같이 println(), print() 메소드가 오버로딩 되어 있음)*

PrintWriter/PrintStream			
void	print(boolean b)	void	println(boolean b)
void	print(char c)	void	println(char c)
void	print(double d)	void	println(double d)
void	print(float f)	void	println(float f)
void	print(int i)	void	println(int i)
void	print(long l)	void	println(long l)
void	print(Object obj)	void	println(Object obj)
void	print(String s)	void	println(String s)
		void	println()

PrintWriter 활용(소켓 입력 스트림)

- 서버에서 문자열 전송

TestServer4.java

```java
//(클라이언트와 연결이 되면) "연결 완료" 메시지를 한 번만 자동으로 보내고 종료(가장 심플하게 구현)

import java.io.IOException;
import java.io.PrintWriter;
import java.net.ServerSocket;
import java.net.Socket;

public class TestServer4 {
    public static void main(String[] args) {
        try {

            ServerSocket listener = new ServerSocket(9999); // 서버 소켓 생성
            Socket socket = listener.accept(); // 클라이언트로부터 연결 요청 대기

            PrintWriter out = new PrintWriter(socket.getOutputStream());
            out.print("서버 연결 완료"+"\n"); // 연결이 되면 클라이언트로 "서버 연결 완료" 메시지를 보냄
            out.flush();

            out.close();
            socket.close();
            listener.close();

        } catch (IOException e) {
            System.out.println("오류가 발생했습니다.");
        }
    }
}
```

서버

클라이언트

받은 메시지: 서버 연결 완료

DataInputStream (기본 자료형 입출력 보조 스트림)

- **DataInputStream과 DataOutputStream 보조 스트림을 연결하면 8가지 기본 자료형의 단위로 읽고 쓸 수 있는 장점이 있음**
- **기본 자료형인 boolean, byte, char, short, int, long, float, double 형태로 입출력 할 수 있음**

DataInputStream과 DataOutputStream 클래스에서 제공하는 메소드

DataInputStream		DataOutputStream	
boolean	readBoolean()	void	writeBoolean(Boolean v)
byte	readByte()	void	writeByte(int v)
char	readChar()	void	writeChar(int v)
double	readDouble()	void	writeDouble(double v)
float	readFloat()	void	writeFloat(float v)
int	readInt()	void	writeInt(int v)
long	readLong()	void	writeLong(long v)
short	readShort()	void	writeShort(int v)
String	readUTF()	void	writeUTF(String str)

반드시 짝 이룸

길이 정보 포함

DataInputStream 활용(서버)

- (클라이언트와 연결이 되면) "연결 완료" 메시지를 한 번만 자동으로 보내고 종료

TestServer5.java

```java
//(클라이언트와 연결이 되면) "연결 완료" 메시지를 한 번만 자동으로 보내고 종료(가장 심플하게 구현)
//앞의 코드를 다음과 같이 수정, 이후 Ctrl+Shift+O 눌러서 import 파일 정리

import java.io.DataOutputStream;
import java.io.IOException;
import java.net.ServerSocket;
import java.net.Socket;

public class TestServer5 {
    public static void main(String[] args) {
        try {

            ServerSocket listener = new ServerSocket(9999); // 서버 소켓 생성
            Socket socket = listener.accept(); // 클라이언트로부터 연결 요청 대기

            DataOutputStream os = new DataOutputStream(socket.getOutputStream());
            os.writeUTF("서버 연결 완료");


            socket.close();
            listener.close();

        } catch (IOException e) {
            System.out.println("오류가 발생했습니다.");
        }
    }
}
```

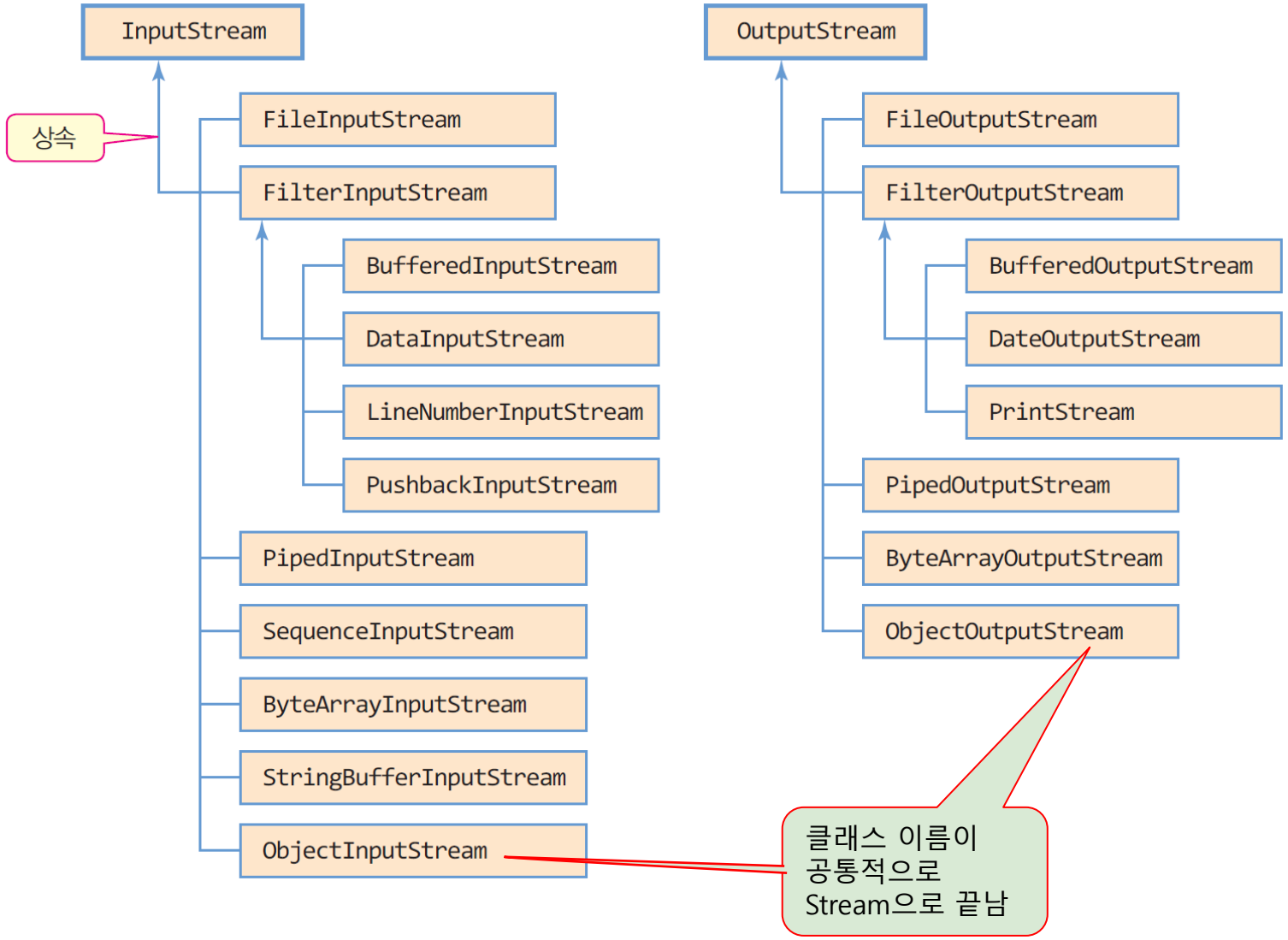
DataInputStream 활용(클라이언트)

- 키보드로부터 문자열을 읽어들이는 코드

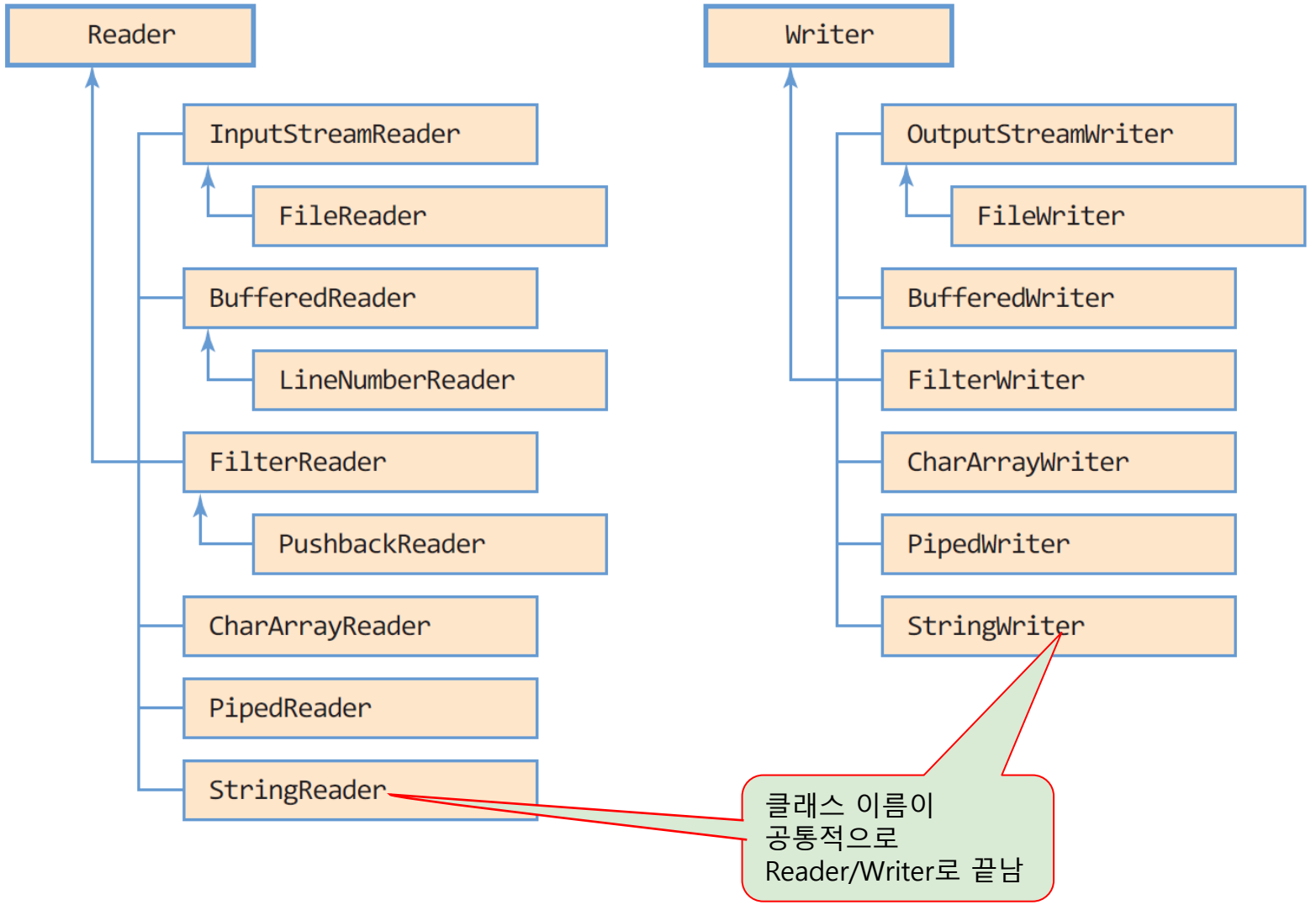
TestServer5.java

```java
import java.io.DataInputStream;
import java.io.IOException;
import java.net.Socket;

public class TestClient36 {
    public static void main(String[] args) {

        try {
            Socket socket = new Socket("localhost", 9999); // 클라이언트 소켓 생성. 서버에 바로 접속

            DataInputStream is = new DataInputStream(socket.getInputStream());

            String inputMessage = is.readUTF();
            System.out.println("받은 메시지: " + inputMessage); // 서버가 보낸 메시지 화면에 출력

            socket.close();

        } catch (IOException e) {
            System.out.println(e.getMessage());
        }
    }
}
```

[참고] JDK의 바이트 스트림 클래스 계층 구조



[참고] JDK의 문자 스트림 클래스 계층 구조



Q&A

담당교수 : 신성

E-mail : sihns@hansung.ac.kr

연구실 : 우촌관 702호

휴대폰 번호 : 010-8873-8353