

# Java



# 파일 입출력에 사용되는 자바 클래스들

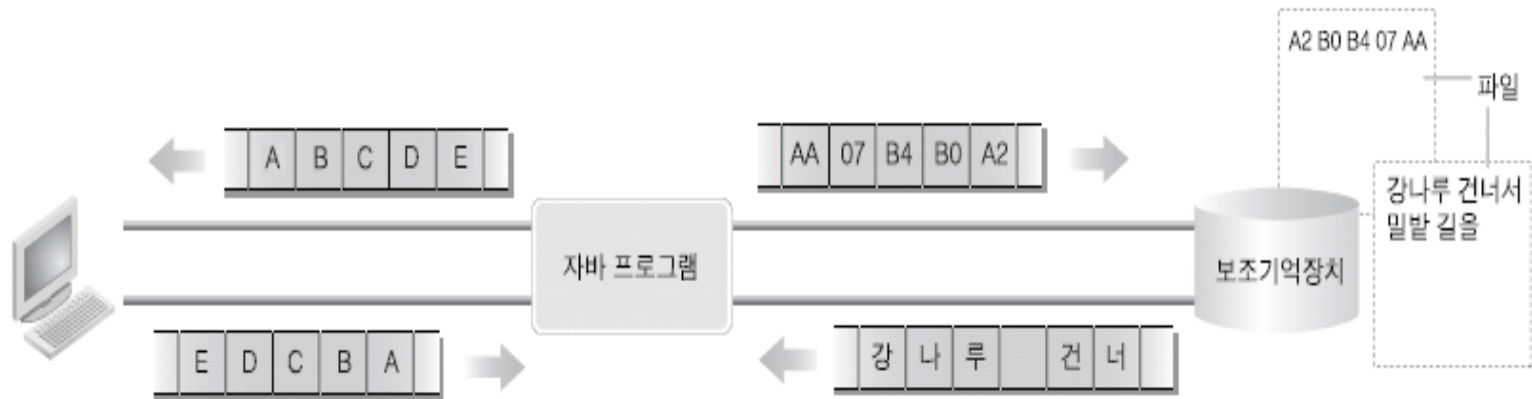
JDK 라이브러리의 파일을 다루는 클래스들  
파일의 내용을 읽고 쓰는 클래스들  
입출력 기능/성능을 향상시키는 클래스들  
데이터를 포맷해서 출력하는 클래스들  
파일 관리에 사용되는 클래스



# 파일 입출력에 사용되는 자바 클래스들

## ● 스트림이란?

• 일차원적인 데이터의 흐름



### • 흐름의 방향에 따른 분류

- 입력 스트림(input stream)
- 출력 스트림(output stream)

### • 데이터의 형태에 따른 분류

- 문자 스트림(character stream)
- 바이트 스트림(byte stream)

# Stream Introduction

## ● Stream?

- ▶ Data를 목적지로 Input/Output하기 위한 방법이다.
- ▶ Stream에 Data를 쓸 수 있고, Stream에서 Data를 읽을 수 있다.
- ▶ Stream에 Data를 쓸 경우, 이러한 Stream을 output stream이라고 한다.
- ▶ Stream에서 Data를 읽을 경우, 이러한 Stream을 input stream이라고 한다.

# Stream Introduction

## ● Stream의 특징

- ➡ Stream은 FIFO 구조이다. - FIFO구조란 먼저 들어간 것이 먼저 나오는 형태로서 Data의 순서가 바뀌지 않는다는 특징이 있다.
- ➡ Stream은 단방향이다. - Java에서 Stream은 읽기, 쓰기가 동시에 되지 않는다. 따라서 읽기, 쓰기가 필요하다면 읽는 Stream과 쓰는 Stream을 하나씩 열어 사용해야 한다.
- ➡ Stream은 지연될 수 있다. - Stream은 넣어진 Data가 처리되기 전까지는 Stream에 사용되는 Thread는 지연상태에 빠진다. 따라서 Network 내에서는 Data가 모두 전송되기 전까지 Network Thread는 지연상태가 된다.

# File class

## File class

- ▶ System에 있는 file이나 directory를 추상화한 class이다.
- ▶ File class를 이용하면 file의 크기, 생성, 삭제, 변경 및 마지막 수정날짜 등 다양한 정보를 알 수 있는 method를 제공하고 있다.

## File class Constructor

※ File 클래스의 주요 생성자

생성자	설명
File(String pathname)	문자열 pathname을 가지고 경로를 생성하여 File 객체를 생성한다.
File(String parent, String child)	Parent와 child 문자열을 연결한 문자열로 경로를 생성하여 File 객체를 생성한다.
File(File parent, String child)	Parent의 파일 객체와 child 문자열로 경로를 생성하여 File 객체를 생성한다.

# File class

## File class의 주요 method

※ File 클래스의 주요 메서드

반환형	메서드	설명
boolean	canRead( )	파일을 읽을 수 있으면 true, 그렇지 않으면 false다.
	canWrite( )	파일을 쓸 수 있으면 true, 그렇지 않으면 false다.
	createNewFile( )	파일을 새로 생성하면 true, 그렇지 않으면 false다.
	delete( )	파일을 지우면 true, 그렇지 않으면 false다.
	exists( )	파일이나 디렉토리가 존재하면 true, 그렇지 않으면 false다.
String	getAbsolutePath( )	파일의 절대 경로를 반환한다.
	getCanonicalPath( )	파일의 정규 경로를 반환한다.
	getName( )	파일명을 반환한다.
boolean	isDirectory( )	디렉토리면 true, 그렇지 않으면 false다.
	isFile( )	파일이면 true, 그렇지 않으면 false다.
long	lastModified( )	1970년 1월 1일부터 현재까지의 시간을 밀리세컨드 초로 반환한다.
	length( )	파일의 크기를 바이트로 반환한다.
String[ ]	list( )	특정 디렉토리의 모든 파일과 자식 디렉토리를 스트링 배열로 반환한다.
boolean	mkdir( )	디렉토리를 생성하면 true, 디렉토리가 있어서 생성하지 못하면 false다.
	renameTo(File dest)	dest 파일 객체로 이름을 바꾸면 true, 그렇지 않으면 false다.

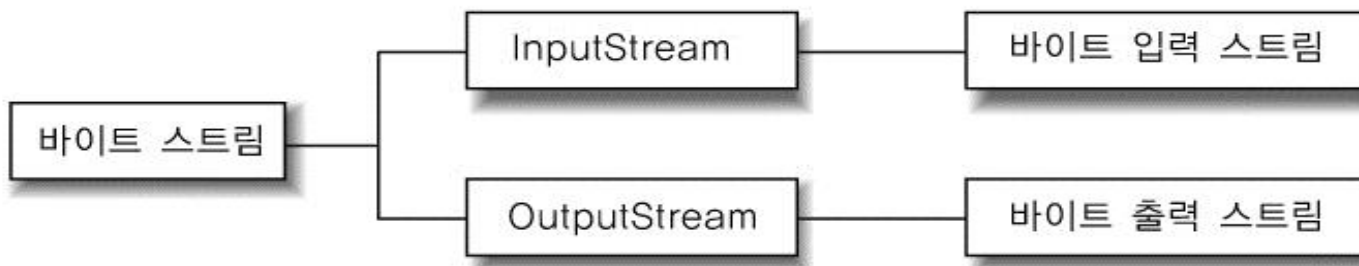
# Byte Stream

## Byte Stream

- ▶ Byte Stream은 1byte를 Input/Output 할 수 있는 Stream이다,
- ▶ 일반적으로 Byte로 구성된 file, 즉 동영상 file, Image file, 음악 file을 처리하기에 적합한 Stream이다,

## Byte Stream의 종류

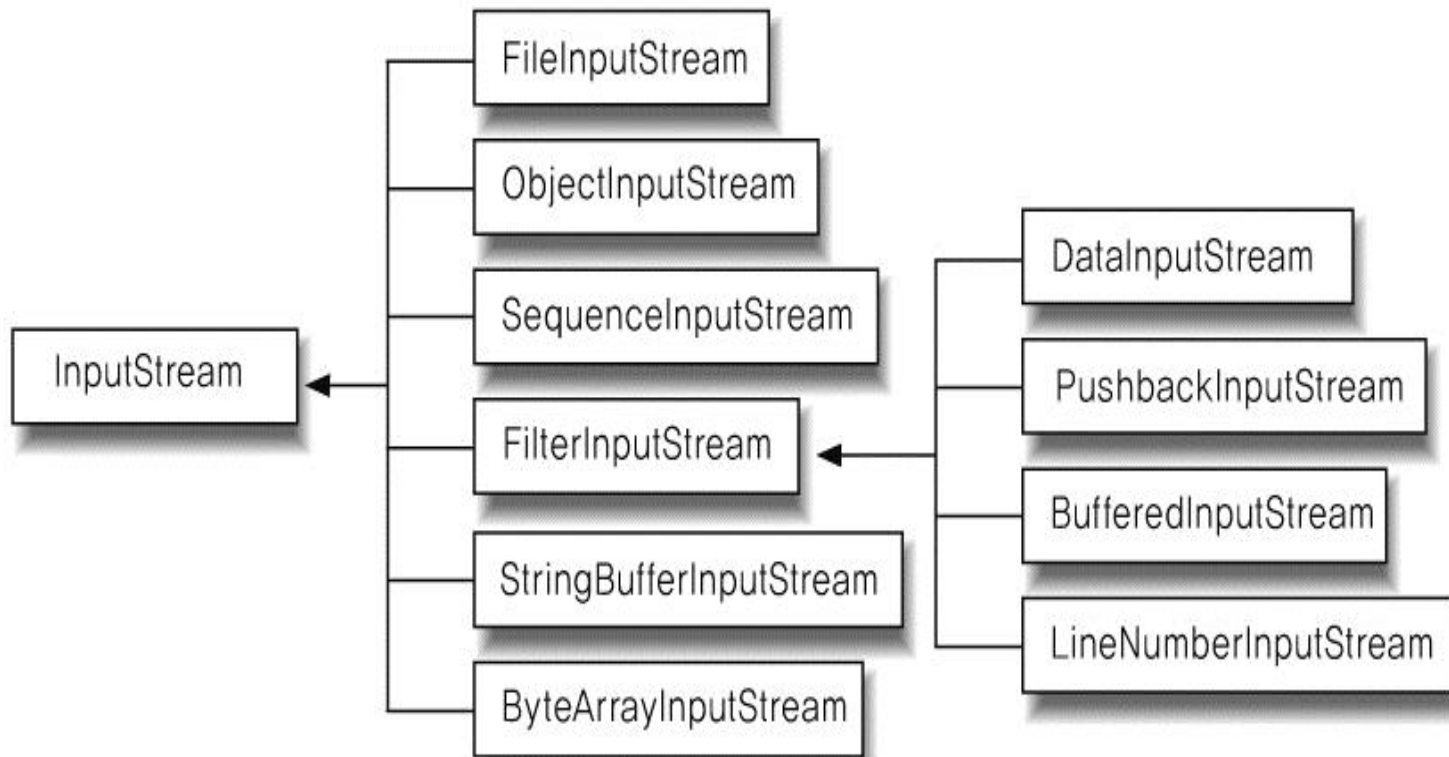
- ▶ InputStream과 OutputStream으로 구성되어 있다.





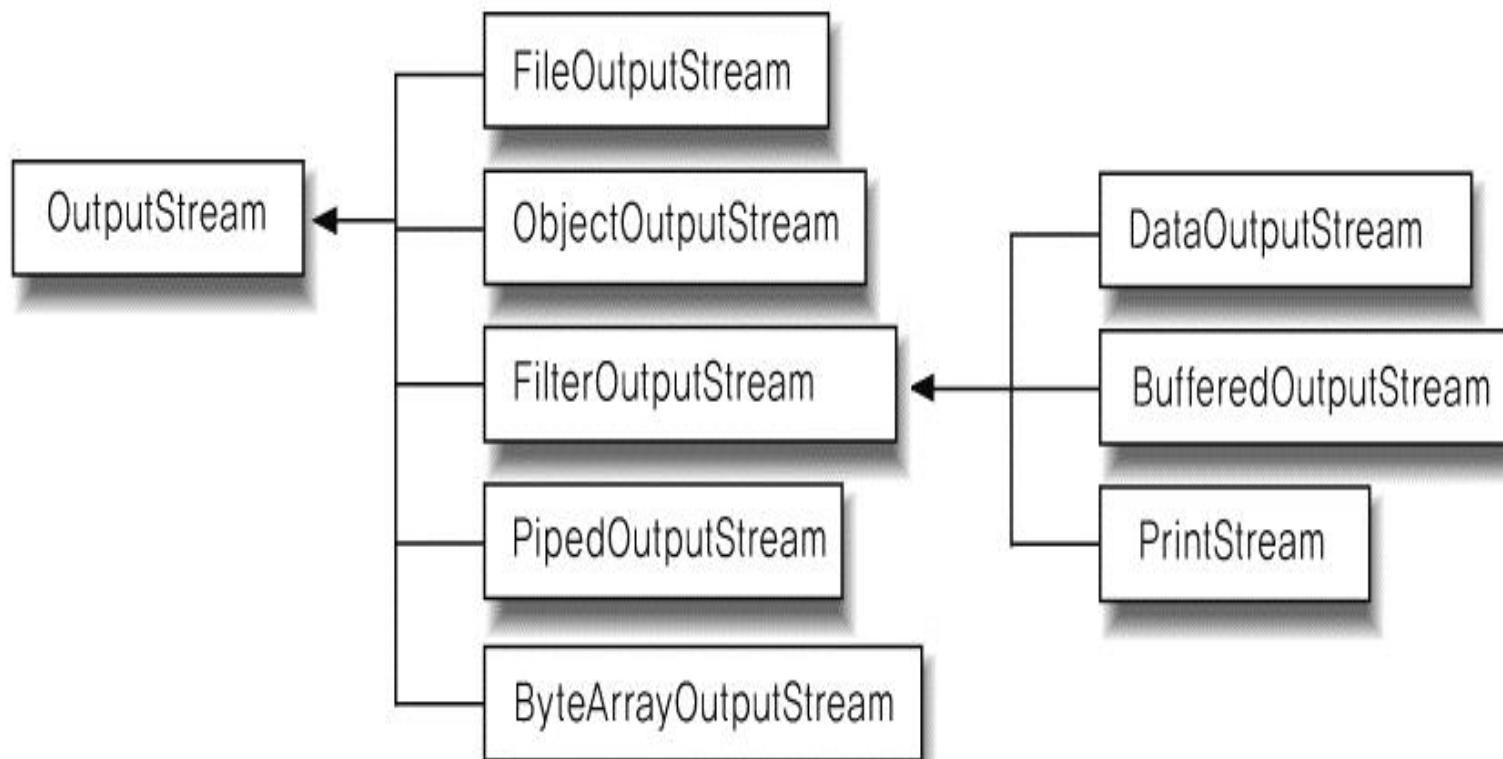
# Byte Stream

## Byte Input Stream의 구조



# Byte Stream

## ● Byte OutputStream의 구조



# Byte Input Stream

## Byte Input Stream(InputStream)

- ▶ InputStream은 Byte Input을 수행하는 데 필요한 method를 정의하는 추상 class이다.
- ▶ Java Program은 Object를 생성하고 생성된 Object와 Byte Stream과 연결함으로써 file을 연다.
- ▶ Java는 다른 장치들과도 Byte Stream을 연결할 되어 있고 Program이 시작되면 장치들과 연결된 세 개의 Object(System.in, System.out, System.err)를 생성한다.
- ▶ System.in Object는 Keyboard로 Byte를 Input할 수 있는 InputStream Object이다.

# Byte Input Stream

## ● InputStream의 주요 method

※ InputStream의 주요 메서드

반환형	메서드	설명
abstract int	read( )	스트림 데이터 1바이트를 읽어온다. 반환값은 0~255의 아스키코드 값이기 때문에 문자로 나타내려면 char로 캐스팅해야 한다. 더 이상 읽을 수 없을 때는 -1을 반환한다.
int	read(byte b[ ])	스트림 데이터 1바이트를 읽어 바이트 배열에 저장하고, 읽은 수만큼 반환한다.
	read(byte b[ ], int start, int length)	스트림 데이터를 length만큼 읽어 바이트 배열 b의 start 위치에 저장하고, 읽을 수만큼 반환한다.
	available( )	읽을 수 있는 바이트 수를 반환한다.
long	skip(long n)	읽을 수 있는 바이트에서 n만큼 건너뛴다.
void	close( )	입력 스트림을 닫는다.

# Byte Input Stream

## FileInputStream

- ➡ FileInputStream은 System에 있는 모든 file을 읽을 수 있는 기능을 제공한다.
- ➡ file을 읽을 때는 file의 경로와 file Object를 Constructor의 parameter로 설정할 수 있다.
- ➡ 만약, file이 존재하지 않으면 FileNotFoundException을 발생하게 된다.

※ FileInputStream의 주요 생성자

생성자	설명
<code>FileInputStream(String name)</code>	name이 의미하는 것은 파일 시스템의 실제 경로를 의미하고, 이것을 매개변수로 FileInputStream 객체를 생성한다.
<code>FileInputStream(File file)</code>	File 객체를 이용하여 FileInputStream 객체를 생성한다.

# Byte Input Stream

## ● DataInputStream

- ➡ DataInput Interface는 Input Stream으로부터 기본형 Data를 읽기 위한 method를 정의한다.
- ➡ DataInput Interface는 Primitive Data Type을 읽을 수 있는 각종 method와 Character를 읽을 수 있는 method를 정의 하고 있다.
- ➡ DataInputStream class의 Constructor는 한 개로 구성되어 있으며, 어떠한 예외 처리도 되어 있지 않다.

※ DataInputStream의 주요 생성자

생성자	설명
DataInputStream (InputStream in)	매개변수인 InputStream 객체로 DataInputStream 객체를 생성한다.

# Byte Input Stream

## ● BufferedInputStream

- ➡ Bufferring은 Input/Output 수행을 향상 시킨 기술이다.
- ➡ Bufferring이란 논리적 Data 덩어리들이 하나의 큰 물리적 Input 연산으로서 file로부터 읽어서 Buffer로 Input 하는 것을 말한다.
- ➡ Bufferring을 이용하면 Data를 읽어서 Buffer를 꽉 채우고, 연속된 read() method 호출의 경우는 단지 memory Buffer로부터 Data를 읽어 내는 것일뿐으므로 훨씬 효율적이다.
- ➡ 또한 mark 기능과 reset 기능을 구현 추가로 구현하였다.

※ BufferedInputStream의 주요 생성자

생성자	설명
BufferedInputStream (InputStream in)	매개변수인 InputStream 객체로 BufferedInputStream 객체를 생성한다.
BufferedInputStream (InputStream in, int size)	매개변수인 InputStream 객체로 BufferedInputStream 객체를 생성하고 size는 버퍼의 용량을 정하는 부분인데, 만약 지정하지 않으면 8192byte가 정해져 있다.

# Byte Output Stream

## Byte Output Stream(OutputStream)

- OutputStream은 Byte Output을 수행하는 필요한 method를 정의한 abstract class이다.
- Program이 시작 되면 장치와 연결된 두 개의 Output Stream은 System.out, System.err를 생성한다.
- System.out Object는 화면에 Data를 Output 한다.
- System.err Object는 화면에 error Message를 Output하게 된다.

### ※ OutputStream의 주요 메서드

반환형	메서드	설명
abstract void	write(int b)	출력 스트림으로 b의 값을 바이트로 변환하여 쓰기한다.
void	write(byte[ ] b)	출력 스트림으로 바이트 배열 b를 쓰기한다.
	write(byte[ ] b, int start, int length)	출력 스트림으로 바이트 배열 b를 start부터 length만큼 쓰기한다.
	flush( )	출력 스트림을 통하여 쓰기를 할 때 일반적으로 버퍼에 가득차게 되면 한꺼번에 보내게 되는데, 이 메서드를 사용하게 되면 버퍼에 가득 차 있지 않더라도 버퍼의 내용을 바로 보내게 된다.
	close( )	모든 자원을 반납한다.



# Byte Output Stream

## FileOutputStream

- ▶ `FileOutputStream`은 System에 있는 모든 file에 쓸 수 있는 기능을 제공한다.
- ▶ 만약 `Object`를 생성할 때, file이 존재하지 않으면 `FileNotFoundException`을 발생하게 된다.
- ▶ `Object`가 생성되면 file이 있는 경우에는 file을 생성하지 않으면 file이 없는 경우에는 file을 생성하게 된다.
- ▶ `FileNotFoundException`의 의미는 경로가 일치하지 않을 때 발생하는 예외이다. 즉 경로는 일치하고 file이 없는 경우에는 예외가 발생하지 않고 file 생성하게 된다.

# Byte Output Stream

## ※ FileOutputStream의 주요 생성자

생성자	설명
<code>FileOutputStream</code> (String name)	name이 의미하는 것은 파일 시스템의 실제 경로를 의미하고, 이것을 매개변수로 <code>FileOutputStream</code> 객체를 생성한다.
<code>FileOutputStream</code> (String name, boolean append)	name이 의미하는 것은 파일 시스템의 실제 경로를 의미하고, <code>append</code> 가 <code>true</code> 이면 이어쓰기의 기능을 하고, <code>false</code> 이면 덮어쓰기를 한다. 이 두 개의 매개변수로 <code>FileOutputStream</code> 객체를 생성한다.
<code>FileOutputStream</code> (File file)	File 객체를 이용하여 <code>FileOutputStream</code> 객체를 생성한다.
<code>FileOutputStream</code> (File file, boolean append)	이 두 개의 매개변수로 <code>FileOutputStream</code> 객체를 생성한다. <code>append</code> 의 의미는 두 번째 생성자와 동일하다.

# Byte Output Stream

## ■ DataOutputStream

- ▶ DataOutput Interface는 Output Stream으로부터 기본형 Data를 쓰기 위한 method를 정의한다.
- ▶ DataOutput Interface는 Primitive Data Type을 쓸 수 있는 각종 method와 Character를 쓸 수 있는 method를 정의 하고 있다.
- ▶ DataOutputStream class의 Constructor는 한 개 로 구성되어 있으며, 어떠한 예외 처리도 되어 있지 않다.

※ DataOutputStream의 주요 생성자

생성자	설명
DataOutputStream (OutputStream out)	매개변수인 OutputStream 객체로 DataOutputStream 객체를 생성한다.

# Byte Output Stream

## ● BufferedOutputStream

- ➡ 이 class를 사용하면 Buffer가 채워질 때마다 한번에 대량으로 Output장치로의 실제 전송이 수행된다.
- ➡ OutputStream은 Output 속도의 향상을 위해서 flush() method를 정의하고 있다, 하지만 실제로는 구현되지 않았다.
- ➡ flush란 Buffer가 다 차지 않더라도 Buffer를 비워주는 기능.
- ➡ flush 기능을 구현한 class가 바로 BufferedOutputStream class가 된다.

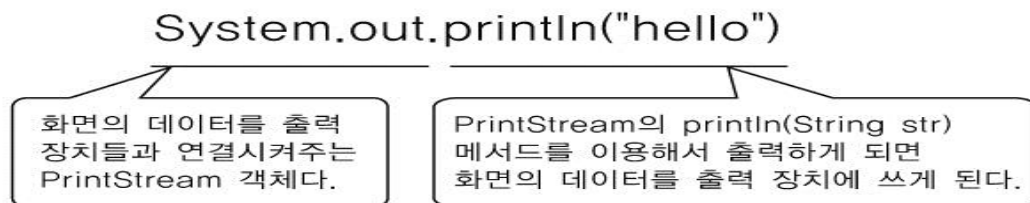
※ BufferedOutputStream의 주요 생성자

생성자	설명
BufferedOutputStream (OutputStream out)	매개변수인 OutputStream 객체로 BufferedOutputStream 객체를 생성한다.
BufferedOutputStream (OutputStream out, int size)	매개변수인 OutputStream 객체로 BufferedOutputStream 객체를 생성하고 size는 버퍼의 용량을 정하는 부분인데, 만약 지정하지 않으면 8192byte로 정해진다.

# Byte Output Stream

## PrintStream

- ▶ PrintStream은 모든 Data Type을 Output할 수 있는 `print()`, `println()` method가 Overloading 되어 있다.
- ▶ Program이 시작되면 장치와 연결된 OutputStream인 `System.out`, `System.err` Object가 PrintStream Object다.



- ▶ Java 5.0에서는 `PrintStream`의 `format()` method와 `printf()` method가 추가되어 있기 때문에 이전의 `System.out.printf()` 나 `System.out.format()`을 이용해서 Output문을 작성할 수 있었다.

# Byte Output Stream

## PrintStream

- ▶ PrintStream은 두 가지 중요한 특징을 가진다.
- ▶ 첫번째, 다른 Stream과는 다르게 flush 기능을 자동으로 처리할 수 있는 Constructor를 가지고 있다.
- ▶ 두번째, 모든 method의 예외처리를 하지 않았다는 점이다.

# 파일 입출력에 사용되는 자바 클래스들

## ● 스트림을 다루는 클래스들

### ● 파일 입출력 스트림을 다루는 클래스들

The screenshot shows the Java 2 Platform SE 5.0 API documentation for the `java.io` package. The browser window title is "java.io (Java 2 Platform SE 5.0) - Microsoft Internet Explorer". The address bar shows "http://java.sun.com/j2se/1.5.0/docs/api/index.html". The left sidebar lists various Java packages, with `java.io` selected. Below the package list, the `java.io` package is expanded, showing a list of interfaces and classes. A callout box labeled "java.io 선택" points to the `java.io` package in the sidebar. Another callout box labeled "java.io 패키지에 대한 개략적인 설명이 있는 홈페이지" points to the main content area. The main content area displays the "Package java.io" information, including a description: "Provides for system input and output through data streams, serialization and the file system." Below this, there is an "Interface Summary" table and a "Class Summary" table.

Package java.io

Provides for system input and output through data streams, serialization and the file system.

See: [Description](#)

**Interface Summary**

Interface	Description
<a href="#">Closeable</a>	A <code>Closeable</code> is a source or destination of data that can be closed.
<a href="#">DataInput</a>	The <code>DataInput</code> interface provides for reading bytes from a binary stream and reconstructing from them data in any of the Java primitive types.
<a href="#">DataOutput</a>	The <code>DataOutput</code> interface provides for converting data from any of the Java primitive types to a series of bytes and writing these bytes to a binary stream.
<a href="#">Externalizable</a>	Only the identity of the class of an <code>Externalizable</code> instance is written in the serialization stream and it is the responsibility of the class to save and restore the contents of its instances.
<a href="#">FileFilter</a>	A filter for abstract pathnames.
<a href="#">FilenameFilter</a>	Instances of classes that implement this interface are used to filter filenames.
<a href="#">Flushable</a>	A <code>Flushable</code> is a destination of data that can be flushed.
<a href="#">ObjectInput</a>	<code>ObjectInput</code> extends the <code>DataInput</code> interface to include the reading of objects.
<a href="#">ObjectInputValidation</a>	Callback interface to allow validation of objects within a graph.
<a href="#">ObjectOutput</a>	<code>ObjectOutput</code> extends the <code>DataOutput</code> interface to include writing of objects.
<a href="#">ObjectStreamConstants</a>	Constants written into the <code>Object</code> Serialization Stream.
<a href="#">Serializable</a>	Serializability of a class is enabled by the class implementing the <code>java.io.Serializable</code> interface.

**Class Summary**

Class	Description
<a href="#">BufferedInputStream</a>	A <code>BufferedInputStream</code> adds functionality to another input stream—namely, the ability to buffer the input and to support the <code>mark</code> and <code>reset</code> methods.
<a href="#">BufferedOutputStream</a>	The class implements a buffered output stream.
<a href="#">BufferedReader</a>	Read text from a character-input stream, buffering characters so as to provide for

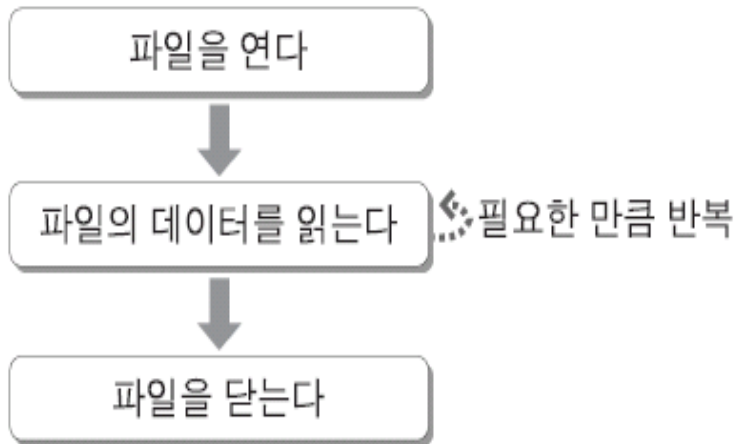
파일 입출력 클래스들

# 파일 입출력에 사용되는 자바 클래스들

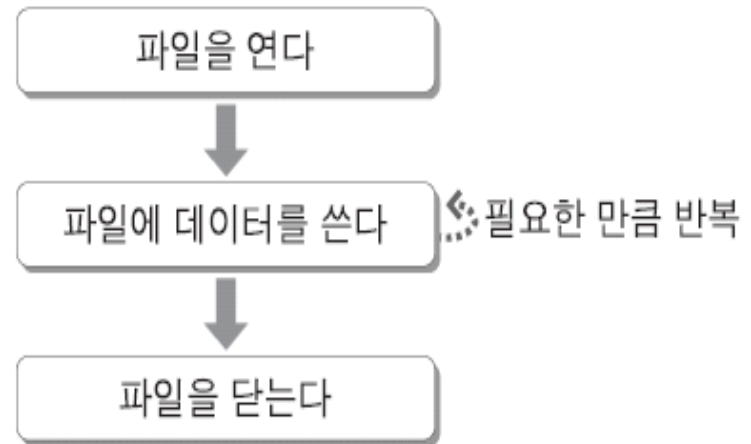
## ● 파일 입출력의 과정

• 3단계로 이루어짐

a) 파일로부터 데이터를 읽는 3단계



b) 파일에 데이터를 쓰는 3단계





# 파일 입출력에 사용되는 자바 클래스들

## ● FileReader 클래스

- **FileReader** 클래스 : 텍스트 파일을 읽는 클래스
- 사용 방법
  - 1) 1단계: 파일을 엽니다

```
FileReader reader = new FileReader("poem.txt");
```

↑  
생성자 안에서 현재 디렉토리의  
poem.txt 파일을 엽니다.

# 파일 입출력에 사용되는 자바 클래스들

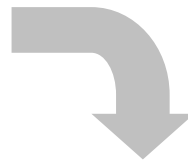
## ● FileReader 클래스

### •• 사용 방법

- 2) 2단계: 파일을 읽습니다

data = reader.read();

↑  
이 메서드는 파일에 있는  
문자 하나를 읽어서 리턴합니다.



```
while (true) {  
    int data = reader.read();  
    if (data < 0) }  
        break;  
    char ch = (char) data;  
              
}
```

데이터를 읽어서

마이너스 값이면 반복을 중단하고,

아니면 char 타입으로 캐스트합니다.

데이터 처리 로직이 들어가는 부분

# 파일 입출력에 사용되는 자바 클래스들

## ● FileReader 클래스

### •• 사용 방법

- 3) 3단계: 파일을 닫습니다

```
reader.close();
```

파일을 닫는 메소드

# 파일 입출력에 사용되는 자바 클래스들

## ● FileReader 클래스

- [예제 10-1] 텍스트 파일을 읽는 프로그램 - 미완성 (1)

```
1  import java.io.*;
2  class ReaderExample1 {
3      public static void main(String args[]) {
4          FileReader reader = new FileReader("poem.txt"); ----- 파일을 여는 부분
5          while (true) {
6              int data = reader.read();
7              if (data == -1)
8                  break;
9              char ch = (char) data;
10             System.out.print(ch);
11         }
12         reader.close(); ----- 파일을 닫는 부분
13     }
14 }
```



명령 프롬프트

```
E:\work\chap10\10-2-1>javac ReaderExample1.java
ReaderExample1.java:4: unreported exception java.io.FileNotFoundException; must
be caught or declared to be thrown
    FileReader reader = new FileReader("poem.txt");
                        ^
ReaderExample1.java:6: unreported exception java.io.IOException; must be caught
or declared to be thrown
    int data = reader.read();
                    ^
ReaderExample1.java:12: unreported exception java.io.IOException; must be caught
or declared to be thrown
    reader.close();
    ^
3 errors
E:\work\chap10\10-2-1>_
```

# 파일 입출력에 사용되는 자바 클래스들

## ● FileReader 클래스

### • [예제 10-2] 텍스트 파일을 읽는 프로그램 - 미완성 (2)

```
1  import java.io.*;
2  class ReaderExample1 {
3      public static void main(String args[]) {
4          try {
5              FileReader reader = new FileReader("poem.txt");
6              while (true) {
7                  int data = reader.read();
8                  if (data == -1)
9                      break;
10                 char ch = (char) data;
11                 System.out.print(ch);
12             }
13             reader.close();
14         }
15         catch (FileNotFoundException fnfe) {
16             System.out.println("파일이 존재하지 않습니다.");
17         }
18         catch (IOException ioe) {
19             System.out.println("파일을 읽을 수 없습니다.");
20         }
21     }
22 }
```

익셉션이 발생하는 부분을  
try 문으로 묶었습니다.

FileReader의 생성자가  
발생하는 익셉션을 처리

FileReader의 read, close 메소드가  
발생하는 익셉션을 처리

# 파일 입출력에 사용되는 자바 클래스들

## FileReader 클래스

### • [예제 10-3] 텍스트 파일을 읽는 프로그램 - 완성

```
1  import java.io.*;
2  class ReaderExample1 {
3      public static void main(String args[]) {
4          FileReader reader = null;
5          try {
6              reader = new FileReader("poem.txt");
7              while (true) {
8                  int data = reader.read();
9                  if (data == -1)
10                     break;
11                  char ch = (char) data;
12                  System.out.print(ch);
13              }
14          }
15          catch (FileNotFoundException fnfe) {
16              System.out.println("파일이 존재하지 않습니다.");
17          }
18          catch (IOException ioe) {
19              System.out.println("파일을 읽을 수 없습니다.");
20          }
21          finally {
22              try {
23                  reader.close();
24              }
25              catch (Exception e) {
26              }
27          }
28      }
29  }
```

poem.txt - 메모장

강나루 건너서  
말발 길을  
구름에 달 가듯이  
가는 나그네.

프로그램을 실행하기 전에 실행 디렉토리에

명령 프롬프트

E:\work\ch10\10-2-1>java ReaderExample1 <---  
강나루 건너서  
말발 길을  
구름에 달 가듯이  
가는 나그네.

그리고 나서 프로그램을 실행하면  
poem.txt 파일의 내용이 출력됨

# 파일 입출력에 사용되는 자바 클래스들

## 02. 파일의 내용을 읽고 쓰는 클래스들

### FileReader 클래스

- 한꺼번에 여러 문자를 읽는 `read` 메소드

```
int num = reader.read(arr);
```



파일로부터 읽은 문자를 담을 char 배열

[올바른 예]

```
char arr[] = new char[100];  
int num = reader.read(arr);
```

[잘못된 예]

```
char arr[];  
int num = reader.read(arr);
```

# 파일 입출력에 사용되는 자바 클래스들

## 02. 파일의 내용을 읽고 쓰는 클래스들

### FileWriter 클래스

- **FileWriter** 클래스 : 텍스트를 파일에 쓰는 클래스
- 사용 방법
  - 1) 1단계: 파일을 엽니다

```
FileWriter writer = new FileWriter("output.txt");
```

현재 디렉토리에 output.txt라는  
파일을 새로 만들어서 엽니다.



# 파일 입출력에 사용되는 자바 클래스들

## 02. 파일의 내용을 읽고 쓰는 클래스들

### FileWriter 클래스

#### •• 사용 방법

- 2) 2단계: 파일에 문자를 씁니다

```
writer.write(ch);
```

↑  
이 문자를 파일에 씁니다.

- 3) 3단계: 파일을 닫습니다

```
writer.close();
```

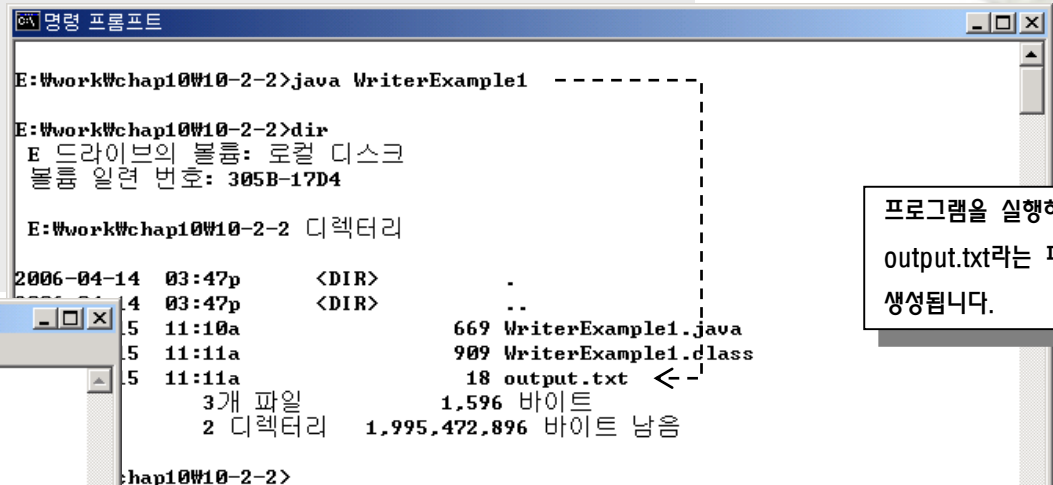
↑  
파일을 닫는 메소드

# 파일 입출력에 사용되는 자바 클래스들

## ● FileWriter 클래스

- [예제 10-4] FileWriter 클래스로 문자 데이터를 파일에 쓰는 프로그램

```
1  import java.io.*;
2  class WriterExample1 {
3      public static void main(String args[]) {
4          FileWriter writer = null;
5          try {
6              writer = new FileWriter("output.txt"); ----- 파일을 엽니다.
7              char arr[] = { '뇌', '를', ' ', '자', '극', '하', '는', ' ', 'J', 'a', 'v', 'a' };
8              for (int cnt = 0; cnt < arr.length; cnt++) }----- 파일에 반복해서 문자들을 씁니다.
9                  writer.write(arr[cnt]);
10             }
11         catch (IOException ioe) {
12             System.out.println("파일로 출력할 수 없습니다.");
13         }
14         finally {
15             try {
16                 writer.close();
17             }
18             catch (Exception e) {
19             }
20         }
21     }
22 }
```



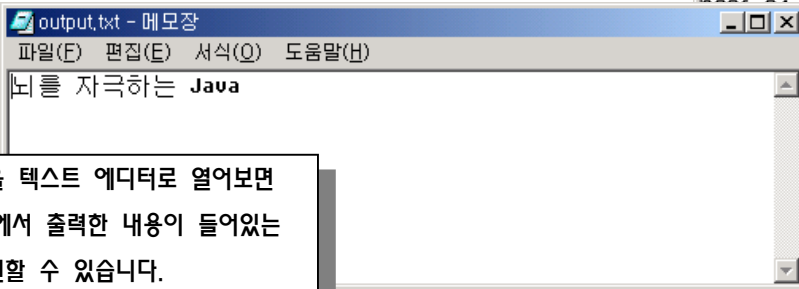
```
E:\work\chap10\10-2>java WriterExample1
E:\work\chap10\10-2>dir
E 드라이브의 볼륨: 로컬 디스크
볼륨 일련 번호: 305B-17D4

E:\work\chap10\10-2 디렉터리

2006-04-14  03:47p    <DIR>          .
              4 03:47p    <DIR>          ..
              5 11:10a             669 WriterExample1.java
              5 11:11a             909 WriterExample1.class
              5 11:11a              18 output.txt  <-
              3개 파일              1,596 바이트
              2 디렉터리    1,995,472,896 바이트 남음

E:\work\chap10\10-2>
```

프로그램을 실행하면  
output.txt라는 파일이  
생성됩니다.



그 파일을 텍스트 에디터로 열어보면  
프로그램에서 출력한 내용이 들어있는  
것을 확인할 수 있습니다.

# 파일 입출력에 사용되는 자바 클래스들

## 02. 파일의 내용을 읽고 쓰는 클래스들

### FileWriter 클래스

- 한꺼번에 여러 문자를 출력하는 write 메소드

```
writer.write(arr);
```

↑  
char 배열

arr 배열에 있는 모든 문자들을 파일로 출력

# 파일 입출력에 사용되는 자바 클래스들

## 02. 파일의 내용을 읽고 쓰는 클래스들

### FileOutputStream 클래스

•• FileOutputStream 클래스 : 바이트 데이터를 파일에 쓰는 클래스

•• 사용 방법

- 1) 1단계: 파일을 엽니다
  - \* 방법은 FileWriter 클래스와 동일

- 2) 2단계: 파일에 데이터를 씁니다

```
outputStream.write(71);
```

71의 비트 패턴인 01000111을 갖는 하나의 바이트를 파일로 출력

- 3) 3단계: 파일을 닫습니다
  - \* 방법은 FileWriter 클래스와 동일

# 파일 입출력에 사용되는 자바 클래스들

## FileOutputStream 클래스

- [예제 10-5] FileOutputStream 클래스로 바이트 데이터를 파일에 쓰는 프로그램

```
1  import java.io.*;
2  class OutputStreamExample1 {
3      public static void main(String args[]) {
4          FileOutputStream out = null;
5          try {
6              out = new FileOutputStream("output.dat");----- 파일을 엽니다.
7              byte arr[] = { 0, 1, 2, 3, 4, 5, 6, 7, 8, 9,
8                           10, 11, 12, 13, 14, 15, 16, 17, 18, 19 };
9              for (int cnt = 0; cnt < arr.length; cnt++) }----- 파일에 반복해서 byte 타입 데이터를 씁니다.
10                 out.write(arr[cnt]);
11             }
12             catch (IOException ioe) {
13                 System.out.println("파일로 출력할 수 없습니다.");
14             }
15             finally {
16                 try {
17                     out.close();
18                 }
19                 catch (Exception e) {
20                 }
21             }
22         }
23     }
```

```
E:\work\chap10\10-2-3>java OutputStreamExample1
E:\work\chap10\10-2-3>dir
E 드라이브의 볼륨: 로컬 디스크
볼륨 일련 번호: 305B-17D4

E:\work\chap10\10-2-3 디렉터리

2006-04-14  05:19p      <DIR>          .
2006-04-14  05:19p      <DIR>          ..
2006-04-15  12:37p                939 OutputStreamExample1.class
2006-04-15  12:37p                20 output.dat
2006-04-15  12:24p                696 OutputStreamExample1.java
3개 파일                1,655 바이트
2 디렉터리            1,991,655,424 바이트 남음

E:\work\chap10\10-2-3>
```

프로그램을 실행하면  
output.dat라는 파일이  
생깁니다.

# 파일 입출력에 사용되는 자바 클래스들

## 02. 파일의 내용을 읽고 쓰는 클래스들

### FileInputStream 클래스

• FileInputStream 클래스 : 파일로부터 바이트 단위로 데이터를 읽는 클래스

• 사용 방법

- 1) 1단계: 파일을 엽니다
  - \* 방법은 FileReader 클래스와 동일
- 2) 2단계: 파일로부터 데이터를 읽습니다

```
while (true) {  
    int data = inputStream.read(); — 데이터를 읽어서  
    if (data < 0) } — -1이면 반복을 중단하고  
        break;  
    byte b = (byte) data; — 아니면 byte 타입으로 캐스트  
    ...  
}
```

- 3) 3단계: 파일을 닫습니다
  - \* 방법은 FileReader 클래스와 동일

# 파일 입출력에 사용되는 자바 클래스들

## 02. 파일의 내용을 읽고 쓰는 클래스들

### FileInputStream 클래스

- 한꺼번에 여러 바이트를 읽는 `read` 메소드

```
byte arr = new byte[16];
```

byte 타입의 배열을 생성해서  
넘겨줘야 합니다.

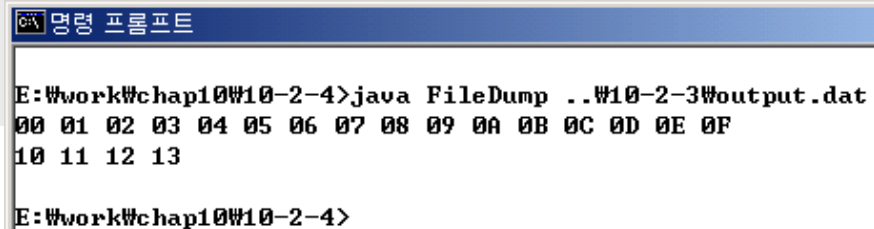
```
int num = inputStream.read(arr);
```

# 파일 입출력에 사용되는 자바 클래스들

## FileInputStream 클래스

- [예제 10-6] 파일의 내용을 읽어서 16진수로 출력하는 프로그램

```
1  import java.io.*;
2  class FileDump {
3      public static void main(String args[]) {
4          if (args.length < 1) {
5              System.out.println("Usage: java FileDump <filename>");
6              return;
7          }
8          FileInputStream in = null;
9          try {
10             in = new FileInputStream(args[0]); ----- 파일을 엽니다.
11             byte arr[] = new byte[16];
12             while (true) {
13                 int num = in.read(arr); ----- 파일로부터 16바이트씩 읽습니다.
14                 if (num < 0)
15                     break;
16                 for (int cnt = 0; cnt < num; cnt++)
17                     System.out.printf("%02X ", arr[cnt]); ----- 읽어들이는 바이트 데이터를 16진수로 출력합니다.
18                 System.out.println();
19             }
20         }
21         catch (FileNotFoundException fnfe) {
22             System.out.println(args[0] + " 파일이 존재하지 않습니다.");
23         }
24         catch (IOException ioe) {
25             System.out.println(args[0] + " 파일을 읽을 수 없습니다.");
26         }
27         finally {
28             try {
29                 in.close(); ----- 파일을 닫습니다.
30             }
31             catch (Exception e) {
32             }
33         }
34     }
35 }
```



```
E:\work\chap10\10-2-4>java FileDump ..\10-2-3\output.dat
00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F
10 11 12 13

E:\work\chap10\10-2-4>
```



# 파일 입출력에 사용되는 자바 클래스들

## 03. 입출력 기능/성능 향상 클래스들

### 입출력 기능/성능을 향상시키는 클래스들

클래스 이름	설명
DataInputStream	프리미티브 타입의 데이터를 입출력하는 클래스
DataOutputStream	
ObjectInputStream	프리미티브 타입과 레퍼런스 타입의 데이터를 입출력하는 클래스
ObjectOutputStream	
BufferedReader	데이터를 한꺼번에 읽어서 버퍼에 저장해두는 클래스
BufferedInputStream	
BufferedWriter	데이터를 버퍼에 저장해두었다가 한꺼번에 출력하는 클래스
BufferedOutputStream	
LineNumberReader	텍스트 파일의 각 행에 번호를 붙여가면서 읽는 클래스

- 이 클래스들은 모두 `java.io` 패키지에 속함
- 이 클래스들은 단독으로는 사용될 수 없음

# 파일 입출력에 사용되는 자바 클래스들

## 03. 입출력 기능/성능 향상 클래스들

### DataOutputStream 클래스

- **DataOutputStream** 클래스 : 프리미티브 타입 데이터를 출력하는 클래스
- 사용 방법
  - 1) 다음과 같은 방법으로 객체를 생성합니다.

```
FileOutputStream out1 = new FileOutputStream("output.dat");
```

FileOutputStream 객체를 생성해서  
DataOutputStream 생성자의 파라미터로 사용합니다.

```
DataOutputStream out2 = new DataOutputStream(out1);
```

# 파일 입출력에 사용되는 자바 클래스들

## ● DataOutputStream 클래스

### ● 사용 방법

- 2) 프리미티브 타입 데이터를 출력하는 메소드를 호출합니다.

메소드	설명
void writeByte(int value)	value의 마지막 1바이트 출력
void writeShort(int value)	value의 마지막 2바이트 출력
void writeChar(int value)	value의 마지막 2바이트 출력
void writeInt(int value)	value 출력
void writeLong(long value)	value 출력
void writeFloat(float value)	value 출력
void writeDouble(double value)	value 출력
void writeBoolean(boolean value)	value 출력

[예]

```
out2.writeInt(14);
```

14를 표현하는 00 00 00 0E 을 00, 00, 00, 0E 로  
나누어서 FileOutputStream의 write 메소드를 4번 호출합니다.

# 파일 입출력에 사용되는 자바 클래스들

## 03. 입출력 기능/성능 향상 클래스들

### DataOutputStream 클래스

- 사용 방법

- 3) 파일을 닫습니다.

```
out2.close();
```



이 메소드 내부에서는 FileOutputStream의  
close 메소드를 호출합니다.

# 파일 입출력에 사용되는 자바 클래스들

## ● DataOutputStream 클래스

### • [예제 10-7] DataOutputStream 클래스의 사용 예

```
1  import java.io.*;
2  class DataOutputExample1 {
3      public static void main(String args[]) {
4          DataOutputStream out = null;
5          try {
6              out = new DataOutputStream(new FileOutputStream("output.dat")); ---- 파일을 엽니다.
7              int arr[] = { 0, 1, 2, 3, 4, 5, 6, 7, 8, 9 };
8              for (int cnt = 0; cnt < arr.length; cnt++)
9                  out.writeInt(arr[cnt]); ----- 파일에 int 타입 데이터를 씁니다.
10         }
11         catch (IOException ioe) {
12             System.out.println("파일로 출력할 수 없습니다.");
13         }
14         finally {
15             try {
16                 out.close(); ---- 파일을 닫습니다.
17             }
18             catch (Exception e) {
19             }
20         }
21     }
22 }
```

# 파일 입출력에 사용되는 자바 클래스들

## 03. 입출력 기능/성능 향상 클래스들

### InputStream 클래스

- **InputStream** 클래스 : 프리미티브 타입 데이터를 읽는 클래스
- 사용 방법
  - 1) 다음과 같은 방법으로 객체를 생성합니다.

```
FileInputStream in1 = new FileInputStream("input.dat");
```

FileInputStream 객체를 생성해서  
InputStream 생성자의 파라미터로 사용합니다.

```
InputStream in2 = new InputStream(in1);
```

# 파일 입출력에 사용되는 자바 클래스들

## ● DataInputStream 클래스

### •• 사용 방법

- 2) 프리미티브 타입의 데이터를 읽는 메소드를 호출합니다.

메소드	설명
byte readByte()	1바이트를 읽어서 byte 타입으로 리턴
short readShort()	2바이트를 읽어서 short 타입으로 리턴
char readChar()	2바이트를 읽어서 char 타입으로 리턴
int readInt()	4바이트를 읽어서 int 타입으로 리턴
long readLong()	8바이트를 읽어서 long 타입으로 리턴
float readFloat()	4바이트를 읽어서 float 타입으로 리턴
double readDouble()	8바이트를 읽어서 double 타입으로 리턴
boolean readBoolean()	1바이트를 읽어서 boolean 타입으로 리턴 (0이면 false, 아니면 true)

[예]

```
try {  
    while (true) {  
        int data = in.readInt();  
        /* data 처리부분 */  
    }  
} catch (EOFException e) {  
    System.out.println("끝");  
}
```

데이터를 읽다가 파일에 끝에 도달하면  
EOFException이 발생합니다.

그러면 실행의 흐름은  
이곳으로 이동합니다.

# 파일 입출력에 사용되는 자바 클래스들

## 03. 입출력 기능/성능 향상 클래스들

### InputStream 클래스

#### •• 사용 방법

- 3) 파일을 닫습니다.
- \* close 메소드 호출 방법은 다른 클래스와 동일




# 파일 입출력에 사용되는 자바 클래스들

## ◆ DataInputStream 클래스

- [예제 10-8] DataInputStream 클래스의 사용 예

```
1  import java.io.*;
2  class DataInputExample1 {
3      public static void main(String args[]) {
4          DataInputStream in = null;
5          try {
6              in = new DataInputStream(new FileInputStream("output.dat")); ----- 파일을 엽니다.
7              while (true) {
8                  int data = in.readInt();
9                  System.out.println(data); }
10             }
11         }
12         catch (FileNotFoundException fnfe) {
13             System.out.println("파일이 존재하지 않습니다.");
14         }
15         catch (EOFException eofe) {
16             System.out.println("끝");
17         }
18         catch (IOException ioe) {
19             System.out.println("파일을 읽을 수 없습니다.");
20         }
21         finally {
22             try {
23                 in.close(); ----- 파일을 닫습니다.
24             }
25             catch (Exception e) {
26             }
27         }
28     }
29 }
```



```
명령 프롬프트
E:\work\chap10\10-3-1>java DataInputExample1
0
1
2
3
4
5
6
7
8
9
E:\work\chap10\10-3-1>
```

# 파일 입출력에 사용되는 자바 클래스들

## 03. 입출력 기능/성능 향상 클래스들

### 문자열을 읽고 쓰는 메소드

- `DataOutputStream` 클래스의 문자열을 쓰는 메소드

```
out.writeUTF("Hello, java");
```

↑                    ↑                    ↑  
이 메소드를 호출하면 이 문자열을 출력합니다.

↑  
DataOutputStream 객체

- `DataInputStream` 클래스의 문자열을 읽는 메소드

```
String str = in.readUTF();
```

↑                    ↑  
이 메소드를 호출하면  
문자열을 읽어서 리턴합니다.

↑  
DataInputStream 객체

# 파일 입출력에 사용되는 자바 클래스들

## 03. 입출력 기능/성능 향상 클래스들

### ObjectOutputStream 클래스

- • ObjectOutputStream 클래스 : 객체를 출력하는 클래스
- • 사용 방법
  - 1) 다음과 같은 방법으로 ObjectOutputStream 객체를 생성합니다.

```
FileOutputStream out1 = new FileOutputStream("output.dat");
```

FileOutputStream 객체를 생성해서  
ObjectOutputStream 생성자의 파라미터로 사용합니다.

```
ObjectOutputStream out2 = new ObjectOutputStream(out1);
```

# 파일 입출력에 사용되는 자바 클래스들

## 03. 입출력 기능/성능 향상 클래스들

### ObjectOutputStream 클래스

#### •• 사용 방법

- 2) 객체를 출력하는 `writeObject` 메소드를 호출합니다.

```
out2.writeObject(obj);
```

↑  
객체를 출력하는 메소드

이렇게 객체를 스트림으로 만드는 것을  
직렬화(serialization)라고 합니다.

- 3) 파일을 닫습니다.
- \* `close` 메소드 호출 방법은 다른 클래스와 동일

# 파일 입출력에 사용되는 자바 클래스들

## 03. 입출력 기능/성능 향상 클래스들

### ObjectOutputStream 클래스

• 직렬화 가능 클래스와 직렬화 불가능 클래스

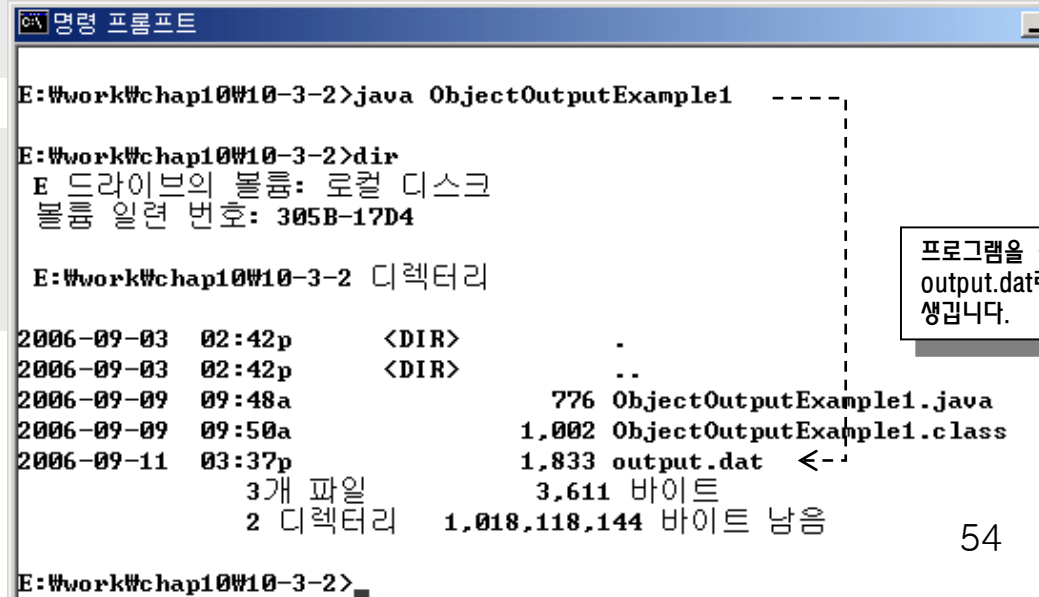
The image shows two side-by-side screenshots of the Java API documentation in a web browser. The left screenshot displays the documentation for the `java.util.GregorianCalendar` class. A red box highlights the 'All Implemented Interfaces' section, which lists `Serializable`, `Cloneable`, and `Comparable<Calendar>`. A text box overlay on this screenshot states: "java.io.Serializable를 구현하므로 직렬화 가능 클래스". The right screenshot displays the documentation for the `java.util.StringTokenizer` class. A text box overlay on this screenshot states: "java.io.Serializable를 구현하지 않으므로 직렬화 불가능 클래스". Both screenshots show the class hierarchy and implemented interfaces, with the left one specifically highlighting the `Serializable` interface.

# 파일 입출력에 사용되는 자바 클래스들

## ObjectOutputStream 클래스

- [예제 10-9] 객체를 파일에 저장하는 프로그램

```
1 import java.io.*;
2 import java.util.GregorianCalendar;
3 class ObjectOutputStreamExample1 {
4     public static void main(String args[]) {
5         ObjectOutputStream out = null;
6         try {
7             out = new ObjectOutputStream(new FileOutputStream("output.dat")); ----- 파일을 엽니다.
8             out.writeObject(new GregorianCalendar(2006, 0, 14));
9             out.writeObject(new GregorianCalendar(2006, 0, 15)); } GregorianCalendar 객체를
10            out.writeObject(new GregorianCalendar(2006, 0, 16)); 생성해서 파일에 씁니다.
11        }
12        catch (IOException ioe) {
13            System.out.println("파일로 출력할 수 없습니다.");
14        }
15        finally {
16            try {
17                out.close(); ----- 파일을 닫습니다.
18            }
19            catch (Exception e) {
20            }
21        }
22    }
23 }
```



```
E:workchap10work10-3-2>java ObjectOutputStreamExample1 -----
E:workchap10work10-3-2>dir
E 드라이브의 볼륨: 로컬 디스크
볼륨 일련 번호: 305B-17D4

E:workchap10work10-3-2 디렉터리

2006-09-03  02:42p        <DIR>          .
2006-09-03  02:42p        <DIR>          ..
2006-09-09  09:48a             776 ObjectOutputStreamExample1.java
2006-09-09  09:50a             1,002 ObjectOutputStreamExample1.class
2006-09-11  03:37p             1,833 output.dat  <-
                               3개 파일             3,611 바이트
                               2 디렉터리          1,018,118,144 바이트 남음

E:workchap10work10-3-2>
```

프로그램을 실행하면  
output.dat에  
생깁니다.

# 파일 입출력에 사용되는 자바 클래스들

## 03. 입출력 기능/성능 향상 클래스들

### ObjectInputStream 클래스

- ObjectInputStream 클래스 : 객체를 읽는 클래스
- 사용 방법
  - 1) 다음과 같은 방법으로 ObjectInputStream 객체를 생성합니다.

```
FileInputStream in1 = new FileInputStream("input.dat");
```

FileInputStream 객체를 생성해서  
ObjectInputStream 생성자의 파라미터로 사용합니다.

```
ObjectInputStream in2 = new ObjectInputStream(in1);
```

# 파일 입출력에 사용되는 자바 클래스들

## 03. 입출력 기능/성능 향상 클래스들

### ObjectInputStream 클래스

#### •• 사용 방법

- 2) 객체를 읽는 `readObject` 메소드를 호출합니다.

```
GregorianCalendar obj = (GregorianCalendar) in2.readObject();
```

↑  
캐스트 연산이 필요

↑  
객체를 읽어서 리턴하는 메소드

- 3) 파일을 닫습니다.
- \* `close` 메소드 호출 방법은 다른 클래스와 동일



# 파일 입출력에 사용되는 자바 클래스들

## ObjectInputStream 클래스

- [예제 10-10] 객체를 파일로부터 읽는 프로그램

```
1  import java.io.*;
2  import java.util.GregorianCalendar;
3  import java.util.Calendar;
4  class ObjectInputExample1 {
5      public static void main(String args[]) {
6          ObjectInputStream in = null;
7          try {
8              in = new ObjectInputStream(new FileInputStream("output.dat")); ----- 파일을 엽니다.
9              while (true) {
10                 GregorianCalendar calendar = (GregorianCalendar) in.readObject();
11                 int year = calendar.get(Calendar.YEAR);
12                 int month = calendar.get(Calendar.MONTH) + 1;
13                 int date = calendar.get(Calendar.DATE);
14                 System.out.println(year + "/" + month + "/" + date);
15             }
16         }
17         catch (FileNotFoundException fnfe) {
18             System.out.println("파일이 존재하지 않습니다.");
19         }
20         catch (EOFException eofe) { ----- 파일로부터 더 이상 읽을 객체가 없을 때
21             System.out.println("끝"); ----- 발생하는 이벤션을 처리합니다.
22         }
23         catch (IOException ioe) {
24             System.out.println("파일을 읽을 수 없습니다.");
25         }
26         catch (ClassNotFoundException cnfe) {
27             System.out.println("해당 클래스가 존재하지 않습니다.");
28         }
29         finally {
30             try {
31                 in.close(); ----- 파일을 닫습니다.
32             }
33             catch (Exception e) {
34             }
35         }
36     }
37 }
```

명령 프롬프트

```
E:\work\chap10\10-3-2>java ObjectInputExample1
2006/1/14
2006/1/15
2006/1/16
??
E:\work\chap10\10-3-2>
```

# 파일 입출력에 사용되는 자바 클래스들

## 03. 입출력 기능/성능 향상 클래스들

### 버퍼를 이용해서 입출력 성능을 향상시키는 클래스들

• 스트림의 종류에 따라 4개의 클래스가 있음

클래스 이름	설명
BufferedInputStream	바이트 입력 스트림을 버퍼링하는 클래스
BufferedOutputStream	바이트 출력 스트림을 버퍼링하는 클래스
BufferedReader	문자 입력 스트림을 버퍼링하는 클래스
BufferedWriter	문자 출력 스트림을 버퍼링하는 클래스

# 파일 입출력에 사용되는 자바 클래스들

## 03. 입출력 기능/성능 향상 클래스들

### BufferedInputStream 클래스

•• **BufferedInputStream** 클래스 : 바이트 입력 스트림의 성능을 향상시키는 클래스

•• 사용 방법

- 1) 다음과 같은 방법으로 객체를 생성합니다.

```
FileInputStream in1 = new FileInputStream("input.dat");
```

FileInputStream 객체를 생성해서  
BufferedInputStream 생성자의 파라미터로 사용합니다.

```
BufferedInputStream in2 = new BufferedInputStream(in1);
```

- 2) `read` 메소드를 호출하여 데이터를 읽습니다.
  - \* `FileInputStream` 클래스의 `read` 메소드 호출 방법과 동일
- 3) 파일을 닫습니다.
  - \* `FileInputStream` 클래스의 `close` 메소드 호출 방법과 동일

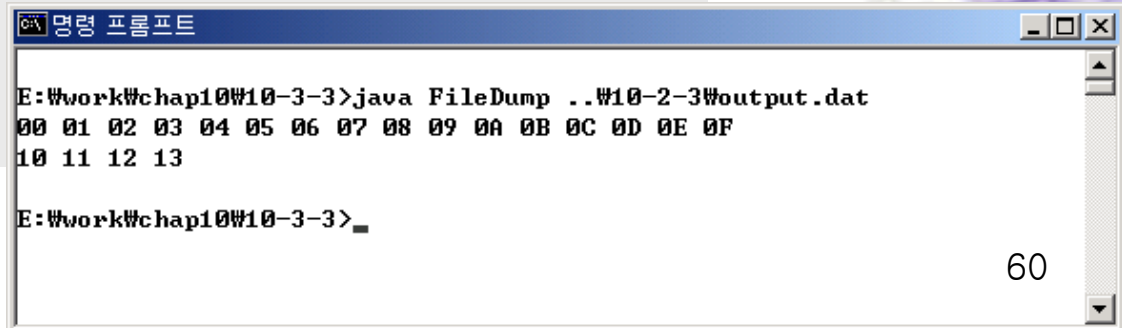
# 파일 입출력에 사용되는 자바 클래스들

## ● BufferedInputStream 클래스

- [예제 10-11] BufferedInputStream로 성능을 향상시킨 FileDump 프로그램

```
1  import java.io.*;
2  class FileDump {
3      public static void main(String args[]) {
4          if (args.length < 1) {
5              System.out.println("Usage: java FileDump <filename>");
6              return;
7          }
8          BufferedInputStream in = null;-----
9          try {
10             in = new BufferedInputStream(new FileInputStream(args[0]));-----
11             byte arr[] = new byte[16];
12             while (true) {
13                 int num = in.read(arr);
14                 if (num < 0)
15                     break;
16                 for (int cnt = 0; cnt < num; cnt++)
17                     System.out.printf("%02X ", arr[cnt]);
18                 System.out.println();
19             }
20         }
21         catch (FileNotFoundException fnfe) {
22             System.out.println(args[0] + " 파일이 존재하지 않습니다.");
23         }
24         catch (IOException ioe) {
25             System.out.println(args[0] + " 파일을 읽을 수 없습니다.");
26         }
27         finally {
28             try {
29                 in.close();
30             }
31             catch (Exception e) {
32             }
33         }
34     }
35 }
```

이 두 부분을 제외한 나머지 부분은 [예제 10-6]과 동일합니다.



```
명령 프롬프트
E:\work\chap10\10-3-3>java FileDump ..\10-2-3\output.dat
00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F
10 11 12 13
E:\work\chap10\10-3-3>
```

# 파일 입출력에 사용되는 자바 클래스들

## 03. 입출력 기능/성능 향상 클래스들

### BufferedInputStream 클래스

•• 버퍼의 크기를 지정하는 방법

```
BufferedInputStream in2 = new BufferedInputStream(in1, 1024);
```

↑                      ↑  
FileInputStream 객체    버퍼의 크기

# 파일 입출력에 사용되는 자바 클래스들

## 03. 입출력 기능/성능 향상 클래스들

### BufferedWriter 클래스

• **BufferedWriter** 클래스 : 문자 출력 스트림의 성능을 향상시키는 클래스

• 사용 방법

- 1) 다음과 같은 방법으로 객체를 생성합니다.

```
FileWriter writer1 = new FileWriter("output.txt");
```

FileWriter 객체를 생성해서  
BufferedWriter 생성자의 파라미터로 사용합니다.

```
BufferedWriter writer2 = new BufferedWriter(writer1);
```

- 2) write 메소드를 호출하여 데이터를 출력합니다.
  - \* FileWriter 클래스의 write 메소드 호출 방법과 동일
- 3) 파일을 닫습니다.
  - \* FileWriter 클래스의 close 메소드 호출 방법과 동일

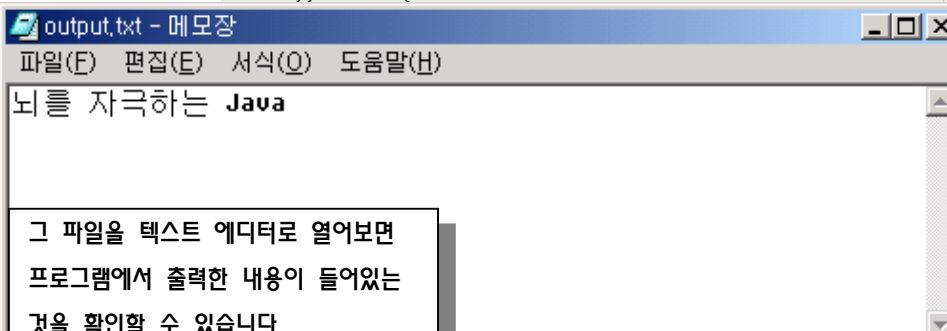
# 파일 입출력에 사용되는 자바 클래스들

## ● BufferedWriter 클래스

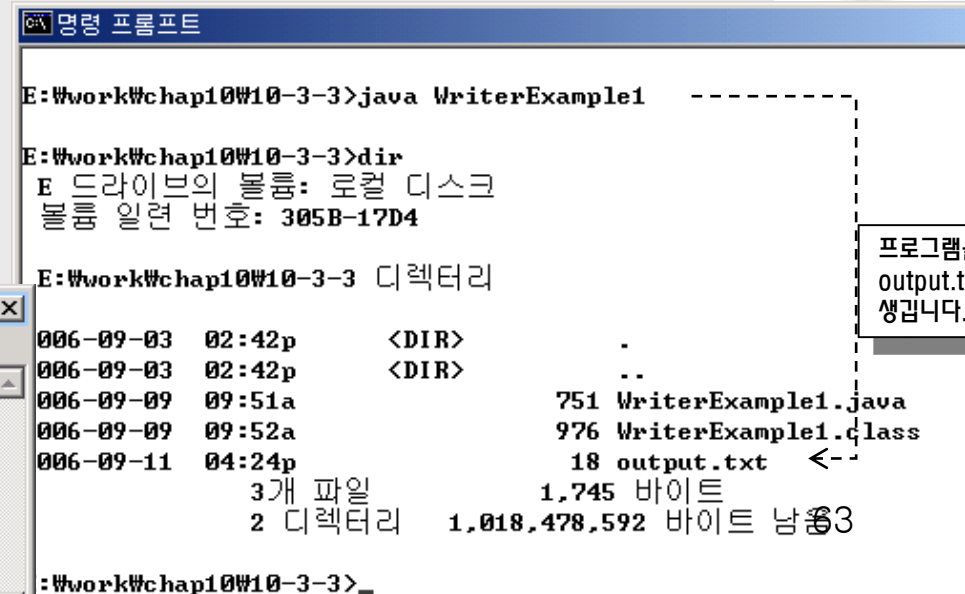
- [예제 10-12] BufferedWriter 클래스로 데이터 출력 성능을 향상시킨 프로그램

```
1 import java.io.*;
2 class WriterExample1 {
3     public static void main(String args[]) {
4         BufferedWriter writer = null;
5         try {
6             writer = new BufferedWriter(new FileWriter("output.txt"));
7             char arr[] = { '뇌', '를', ' ', '자', '극', '하', '는', ' ', 'J', 'a', 'v', 'a' };
8             for (int cnt = 0; cnt < arr.length; cnt++)
9                 writer.write(arr[cnt]);
10        }
11        catch (IOException ioe) {
12            System.out.println("파일로 출력할 수 없습니다.");
13        }
14        finally {
15            try {
16                writer.close();
17            }
18            catch (Exception e) {
19            }
20        }
21    }
22 }
```

이 두 부분을 제외한 나머지 부분은  
[예제 10-4]와 동일합니다.



그 파일을 텍스트 에디터로 열어보면  
프로그램에서 출력한 내용이 들어있는  
것을 확인할 수 있습니다



프로그램을  
output.txt  
생깁니다.

# 파일 입출력에 사용되는 자바 클래스들

## 03. 입출력 기능/성능 향상 클래스들

### BufferedWriter 클래스

- 버퍼에 있는 데이터를 파일에 즉시 출력하는 flush 메소드

```
writer.flush();
```

호출되는 즉시 버퍼의 데이터를  
모두 출력하는 메소드



# 파일 입출력에 사용되는 자바 클래스들

## 03. 입출력 기능/성능 향상 클래스들

### LineNumberReader 클래스

- LineNumberReader 클래스 : 텍스트를 한 줄씩 번호를 매기면서 읽는 클래스
- 사용 방법
  - 1) 다음과 같은 방법으로 객체를 생성합니다.

```
FileReader reader1 = new FileReader("input.txt");
```

FileReader 객체를 생성해서 LineNumberReader 생성자의 파라미터로 사용합니다.

```
LineNumberReader reader2 = new LineNubmerReader(reader1);
```

# 파일 입출력에 사용되는 자바 클래스들

## 03. 입출력 기능/성능 향상 클래스들

### LineNumberReader 클래스

#### •• 사용 방법

- 2) `readLine` 메소드를 호출해서 텍스트를 읽습니다.

```
String str = reader2.readLine();
```

↑  
텍스트 한 줄을 읽어서  
리턴하는 메소드

- 3) 읽어들이는 행의 번호는 `getLineNumber` 메소드를 호출해서 가져옵니다.

```
int num = reader2.getLineNumber();
```

↑  
바로 전에 읽은 행 번호를  
리턴하는 메소드

# 파일 입출력에 사용되는 자바 클래스들

## ● LineNumberReader 클래스

- [예제 10-13] 텍스트 파일에 행 번호를 붙이면서 읽는 프로그램

```
1 import java.io.*;
2 class LineNumberExample1 {
3     public static void main(String args[]) {
4         LineNumberReader reader = null;
5         try {
6             reader = new LineNumberReader(new FileReader("poem.txt")); ----- 파일을 엽니다.
7             while (true) {
8                 String str = reader.readLine();
9                 if (str == null)
10                     break;
11                 int lineNo = reader.getLineNumber();
12                 System.out.println(lineNo + ": " + str);
13             }
14         }
15         catch (FileNotFoundException fnfe) {
16             System.out.println("파일이 존재하지 않습니다.");
17         }
18         catch (IOException ioe) {
19             System.out.println("파일을 읽을 수 없습니다.");
20         }
21         finally {
22             try {
23                 reader.close(); ----- 파일을 닫습니다.
24             }
25             catch (Exception e) {
26             }
27         }
28     }
29 }
```

} 파일로부터 한 행씩 읽어서  
행 번호와 함께 출력합니다.

프로그램을 실행하기 전에 실행 디렉토리에  
poem.txt라는 파일을 만들어 놓으십시오.

poem.txt - 메모장  
파일(F) 편집(E) 서식(O) ...  
거울속에는소리가없소  
저렇게까지조용한세상은참없을것이오  
거울속에도내게귀가있소  
내말을못알아듣는딴한귀가두개나있소  
거울속의나는왼손잡이오

명령 프롬프트

E:\work\chap10\10-3-4>java LineNumberExample1

1: 거울속에는소리가없소  
2: 저렇게까지조용한세상은참없을것이오  
3:  
4: 거울속에도내게귀가있소  
5: 내말을못알아듣는딴한귀가두개나있소  
6:  
7: 거울속의나는왼손잡이오  
8: 내악수를받을줄모르는-악수를모르는왼손잡이오  
9:  
10: 거울때문에나는거울속의나를만져보지를못하는구료마는  
11: 거울이아니었던들내가어찌거울속의나를만나보기라도했겠소  
12:  
13: 나는지금거울을안가졌소마는거울속에는늘거울속의내가있소  
14: 잘은모르지만외로워서일에몰두할께요  
15:  
16: 거울속의나는참나와는반대요마는  
17: 또래잖았소  
18: 나는거울속의나를근심하고진찰할수없으니떡섭섭하오

그리고 나  
poem.txt  
행번호와

# 데이터를 포맷해서 출력하는 클래스들

## PrintWriter 클래스와 PrintStream 클래스

[PrintWriter 클래스의 API 규격서]

The screenshot shows the Java API specification for the `PrintWriter` class in the Java 2 Platform SE 5.0. The left sidebar lists various classes, with `PrintWriter` selected. The main area displays the 'Method Summary' for `PrintWriter`. The methods listed are:

- `append(char c)`: Appends the specified character to this writer.
- `append(CharSequence csq)`: Appends the specified character sequence to this writer.
- `append(CharSequence csq, int start, int end)`: Appends a subsequence of the specified character sequence to this writer.
- `checkError()`: Flush the stream if it's not closed and check its error state.
- `close()`: Close the stream.
- `flush()`: Flush the stream.
- `format(Locale l, String format, Object... args)`: Writes a formatted string to this writer using the specified format string and arguments.
- `format(String format, Object... args)`: Writes a formatted string to this writer using the specified format string and arguments.
- `print(boolean b)`: Print a boolean value.
- `print(char c)`: Print a character.
- `print(char[] s)`: Print an array of characters.
- `print(double d)`: Print a double-precision floating-point number.
- `print(float f)`: Print a floating-point number.
- `print(int i)`: Print an integer.
- `print(long l)`: Print a long integer.
- `print(Object obj)`: Print an object.
- `print(String s)`: Print a string.
- `printf(Locale l, String format, Object... args)`: A convenience method to write a formatted string to this writer using the specified format string and arguments.

[PrintStream 클래스의 API 규격서]

The screenshot shows the Java API specification for the `PrintStream` class in the Java 2 Platform SE 5.0. The left sidebar lists various classes, with `PrintStream` selected. The main area displays the 'Method Summary' for `PrintStream`. The methods listed are:

- `append(char c)`: Appends the specified character to this output stream.
- `append(CharSequence csq)`: Appends the specified character sequence to this output stream.
- `append(CharSequence csq, int start, int end)`: Appends a subsequence of the specified character sequence to this output stream.
- `checkError()`: Flush the stream and check its error state.
- `close()`: Close the stream.
- `flush()`: Flush the stream.
- `format(Locale l, String format, Object... args)`: Writes a formatted string to this output stream using the specified format string and arguments.
- `format(String format, Object... args)`: Writes a formatted string to this output stream using the specified format string and arguments.
- `print(boolean b)`: Print a boolean value.
- `print(char c)`: Print a character.
- `print(char[] s)`: Print an array of characters.
- `print(double d)`: Print a double-precision floating-point number.
- `print(float f)`: Print a floating-point number.
- `print(int i)`: Print an integer.
- `print(long l)`: Print a long integer.
- `print(Object obj)`: Print an object.
- `print(String s)`: Print a string.
- `printf(Locale l, String format, Object... args)`: A convenience method to write a formatted string to this output stream using the specified format string and arguments.

메소드의 기능과 사용 방법이 거의 동일합니다.

# 입출력 기능/성능 향상 클래스들



.

## ● **PrintWriter** 클래스

- **PrintWriter** 클래스 : 데이터를 포맷해서 파일로 출력하는 클래스
- 사용 방법
  - 1) 다음과 같은 방법으로 **PrintWriter** 객체를 생성해서 파일을 엽니다.

```
PrintWriter writer = new PrintWriter("output.txt");
```

↑  
생성자 안에서 이 파일을 엽니다.

# 입출력 기능/성능 향상 클래스들

## PrintWriter 클래스

### •• 사용 방법

- 2) print, println, printf 메소드를 호출합니다.

```
writer.print(12);           // "12"라는 문자열을 출력  
writer.print(10000L);       // "10000"이라는 문자열을 출력
```

```
writer.print(12.5);         // "12.5"라는 문자열과 줄바꿈 문자를 출력  
writer.print(12.5, "e");    // "12.5e"라는 문자열과 줄바꿈 문자를 출력
```

```
writer.printf("%d년 %d월 %d일", 2006, 4, 19);  
// "2006년 4월 19일"이라는 문자열을 출력  
writer.printf("파이=%4.2f", Math.PI);  
// "파이=3.14"라는 문자열을 출력
```

포맷 명세자(format specifier)

포맷 문자열(format string)

# 입출력 기능/성능 향상 클래스들

## PrintWriter 클래스

- 아규먼트 인덱스(argument index)

\\ "J00등 J00등 J00등"이 모든 문자열을 출력  
writer.println("J00등 J00등 J00등" ' J' J00' 3):

아규먼트 인덱스(argument index)

\\ "J00등 J00등 J00등"이 모든 문자열을 출력  
writer.println("J00등 J00등 J00등" ' J00):

\\ 문자열 "J00등 J00등 J00등"을 출력  
writer.println("J00등 J00등 J00등" ' new StringBuilder()):

# 데이터를 포맷해서 출력하는 클래스들

## PrintWriter 클래스

### • [예제 10-14] PrintWriter 클래스의 사용 예

```
1 import java.io.*;
2 import java.util.*;
3 class PrintWriterExample1 {
4     public static void main(String args[]) {
5         PrintWriter writer = null;
6         try {
7             writer = new PrintWriter("output.txt"); ----- 파일을 엽니다.
8             writer.println("    *** 성적표 ***    ");
9             writer.printf("%3s : %3d %3d %3d %5.1f %n", "김지영", 92, 87, 95, 91.3f);
10            writer.printf("%3s : %3d %3d %3d %5.1f %n", "박현식", 100, 90, 88, 92.7f);
11            writer.printf("%3s : %3d %3d %3d %5.1f %n", "최민재", 75, 88, 85, 82.7f);
12            writer.printf("작성일자 : %1$tY년 %1$tm월 %1$td일", new GregorianCalendar());
13        }
14        catch (IOException ioe) {
15            System.out.print("파일로 출력할 수 없습니다.");
16        }
17        finally {
18            try {
19                writer.close(); ----- 파일을 닫습니다.
20            }
21            catch (Exception e) {
22            }
23        }
24    }
25 }
```

데이터를 포맷해서  
출력합니다.

output.txt - 메모장

파일(F) 편집(E) 서식(O) 도움말(H)

\*\*\* 성적표 \*\*\*

김지영 : 92 87 95 91.3

박현식 : 100 90 88 92.7

최민재 : 75 88 85 82.7

작성일자 : 2006년 04월 19일

그 파일을 열어보면  
프로그램에서 출력한  
내용이 있습니다.

명령 프롬프트

E:\work\chap10\10-4>java PrintWriterExample1 -----

E:\work\chap10\10-4>dir

E 드라이브의 볼륨: 로컬 디스크

볼륨 일련 번호: 305B-17D4

E:\work\chap10\10-4 디렉터리

2006-04-19	09:40a	<DIR>	..
2006-04-19	09:40a	<DIR>	.
2006-04-19	09:41a		1,378 PrintWriterExample1.class
2006-04-19	09:41a		146 output.txt
2006-04-19	09:40a		909 PrintWriterExample1.java
		3개 파일	2,433 바이트
		2 디렉터리	1,972,101,120 바이트 남음

E:\work\chap10\10-4>

프로그램을 실행하  
output.txt라는 파  
생깁니다.



# 입출력 기능/성능 향상 클래스들

## PrintStream 클래스

•• **PrintStream 클래스** : 데이터를 포맷해서 파일로 출력하는 클래스

- 사용 방법은 **PrintWriter** 클래스와 동일
- JDK 구버전부터 있던 클래스
- **System.out.println, System.out.print, System.out.printf** 메소드가 속하는 클래스

System (Java 2 Platform SE 5.0) - Microsoft Internet Explorer

주소(D) http://java.sun.com/j2se/1.5.0/docs/api/index.html

Java™ 2 Platform Standard Ed. 5.0

Overview Package **Class** Use Tree Deprecated Index Help

PREV CLASS NEXT CLASS SUMMARY: NESTED | FIELD | CONSTR | METHOD FRAMES NO FRAMES DETAIL: FIELD | CONSTR | METHOD

java.lang  
Class System

java.lang.Object  
↳ java.lang.System

public final class **System**  
extends [Object](#)

The System class contains several useful class fields and methods. It cannot be instantiated.

Among the facilities provided by the System class are standard input, standard output, and error output streams; access to externally defined properties and environment variables; a means of loading files and libraries; and a utility method for quickly copying a portion of an array.

Since:  
JDK1.0

**Field Summary**

static <a href="#">PrintStream</a>	<b>err</b>	The "standard" error output stream.
static <a href="#">InputStream</a>	<b>in</b>	The "standard" input stream.
static <a href="#">PrintStream</a>	<b>out</b>	The "standard" output stream.

**Method Summary**

static void	<b>arraycopy</b> ( <a href="#">Object</a> src, int srcPos, <a href="#">Object</a> dest, int destPos, int length)	Copies an array from the specified source array, beginning at the specified position, to the specified position of the destination array.
static <a href="#">String</a>	<b>clearProperty</b> ( <a href="#">String</a> key)	Removes the system property indicated by the specified key.
static long	<b>currentTimeMillis</b> ()	

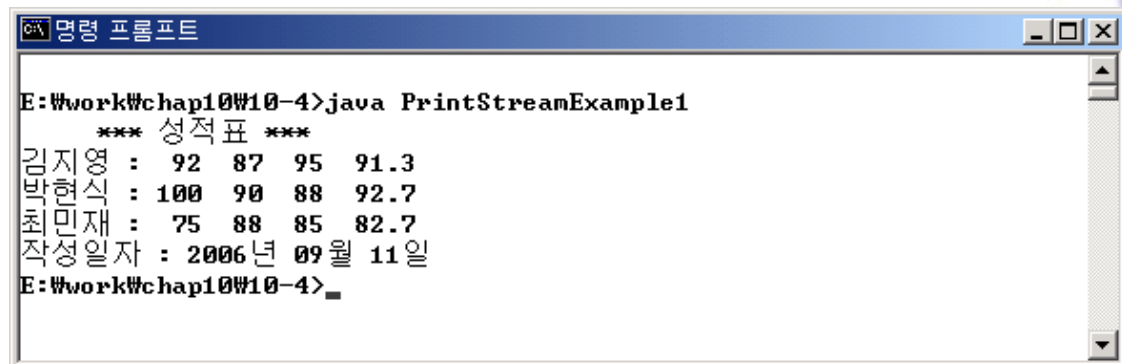
PrintStream 타입으로 선언되어 있는 System 클래스의 out 필드

# 데이터를 포맷해서 출력하는 클래스들

## PrintStream 클래스

- [예제 10-15] PrintStream 클래스의 사용 예

```
1    import java.util.*;
2    class PrintStreamExample1 {
3        public static void main(String args[]) {
4            System.out.println("    *** 성적표 ***    ");
5            System.out.printf("%3s : %3d %3d %3d %5.1f %n", "김지영", 92, 87, 95, 91.3f);
6            System.out.printf("%3s : %3d %3d %3d %5.1f %n", "박현식", 100, 90, 88, 92.7f);
7            System.out.printf("%3s : %3d %3d %3d %5.1f %n", "최민재", 75, 88, 85, 82.7f);
8            System.out.printf("작성일자 : %1$tY년 %1$tm월 %1$td일", new GregorianCalendar());
9        }
10    }
```



```
명령 프롬프트

E:\work\chap10\10-4>java PrintStreamExample1
    *** 성적표 ***
김지영 : 92 87 95 91.3
박현식 : 100 90 88 92.7
최민재 : 75 88 85 82.7
작성일자 : 2006년 09월 11일
E:\work\chap10\10-4>
```

# 파일 관리에 사용되는 File 클래스

## ● File 클래스

- File 클래스 : (파일의 내용이 아니라) 파일 자체를 관리하는 클래스
- 다음과 같은 메소드 제공
  - 파일 정보를 가져오는 메소드
  - 파일 정보를 수정하는 메소드
  - 파일을 생성/삭제하는 메소드
- 디렉토리 관리에도 사용됨

↑  
특수한 형태의 파일이라고 할 수 있음

# . 파일 관리에 사용되는 File 클래스

## ● 파일/디렉토리 정보 가져오기

### •• 사용 방법

- 1) 다음과 같은 방법으로 File 객체를 생성합니다.

```
File file = new File("poem.txt");
```

현재 디렉토리에 있는 poem.txt에 대한  
File 객체를 생성합니다.

```
File file = new File("C:\\work\\chap10");
```

C 드라이브의 work 디렉토리 아래에 있는  
chap10에 대한 File 객체를 생성합니다.

# 파일 관리에 사용되는 File 클래스

## ● 파일/디렉토리 정보 가져오기

### ● 사용 방법

- 2) 파일/디렉토리 정보를 가져오는 메소드를 호출합니다.

```
Boolean isThere = file.exists();
```

↑  
파일 또는 디렉토리가 있으면 true,  
없으면 false를 리턴

```
Boolean isFile = file.isFile();
```

↑  
파일이면 true,  
아니면 false를 리턴

```
Boolean isDir = file.isDirectory();
```

↑  
디렉토리면 true,  
아니면 false를 리턴

# 파일 관리에 사용되는 File 클래스

## ● 파일/디렉토리 정보 가져오기

### ● 사용 방법

- 2) 파일/디렉토리 정보를 가져오는 메소드를 호출합니다. (계속)

<code>String name = file.getName();</code>	<code>// 이름을 리턴</code>
<code>long size = file.length();</code>	<code>// 크기를 리턴</code>
<code>long time = file.lastModified();</code>	<code>// 최종 수정일시를 리턴</code>
<code>boolean readMode = file.canRead();</code>	<code>// 읽기 가능 여부를 리턴</code>
<code>boolean writeMode = file.canWrite();</code>	<code>// 쓰기 가능 여부를 리턴</code>
<code>boolean hiddenMode = file.isHidden();</code>	<code>// 숨김 여부를 리턴</code>
<code>String parent = file.getParent();</code>	<code>// 부모 디렉토리 경로명을 리턴</code>

`Files childs[] = file.listFiles();`

↑  
서브디렉토리와 파일들의 목록을  
리턴하는 메소드

# 파일 관리에 사용되는 File 클래스

## ● 파일/디렉토리 정보 가져오기

- [예제 10-16] 현재 디렉토리의 서브디렉토리와 파일 목록을 출력하는 프로그램

```
1  import java.io.*;
2  import java.util.*;
3  class FileExample1 {
4      public static void main(String args[]) {
5          File file = new File(".");
6          File arr[] = file.listFiles();
7          for (int cnt = 0; cnt < arr.length; cnt++) {
8              String name = arr[cnt].getName();
9              if (arr[cnt].isFile())
10                 System.out.printf("%-25s %7d ", name, arr[cnt].length());
11             else
12                 System.out.printf("%-25s <DIR> ", name);
13             long time = arr[cnt].lastModified();
14             GregorianCalendar calendar = new GregorianCalendar();
15             calendar.setTimeInMillis(time);
16             System.out.printf("%1$tF %1$tT %n", calendar);
17         }
18     }
19 }
```

현재 디렉토리 경로명을 가지고 File 객체를 생성합니다.

서브디렉토리와 파일 목록을 가져옵니다.

가져온 서브디렉토리와 파일의 이름, 크기, 최종수정 일시를 출력합니다.

```
C:\ 명령 프롬프트
E:\work\chap10\10-5>java FileExample1
FileExample1.class          1035 2006-04-19 16:21:34
FileExample1.java          726 2006-04-19 16:21:22
E:\work\chap10\10-5>
```

```
C:\ 명령 프롬프트
E:\work\chap10\10-5>java FileExample1
FileExample1.class          1035 2006-04-19 16:21:
FileExample1.java          726 2006-04-19 16:21:
E:\work\chap10\10-5>mkdir kiwi
E:\work\chap10\10-5>java FileExample1
FileExample1.class          1035 2006-04-19 16:21:
FileExample1.java          726 2006-04-19 16:21:
kiwi                        <DIR> 2006-04-19 16:33:
```

테스트를 위해 디렉토리 생성

# 파일 관리에 사용되는 File 클래스

## ● 파일/디렉토리 생성/삭제하기

•• 파일을 생성하고 삭제하는 메소드

```
File file1 = new File("poem.txt");  
file1.createNewFile();  
File file2 = new File("C:\\doc\\회의록.hwp");  
file2.delete();
```

••

현재 디렉토리에 poem.txt라는 이름의 파일을 생성합니다.

C:\doc\회의록.hwp 파일을 삭제합니다.

```
File file1 = new File("C:\\올빼미");  
file1.mkdir();  
File file2 = new File("두루미");  
file2.delete();
```

C 드라이브의 루트 디렉토리에 “올빼미” 라는 디렉토리를 생성합니다.

현재 디렉토리에 있는 “두루미” 라는 디렉토리를 삭제합니다



# 파일 관리에 사용되는 File 클래스

## ● 임시 파일 생성하기

- 임시 파일을 생성하는 메소드

```
File tmpFile = file.createTempFile("tmp", ".txt", tmpDir);
```

↑  
tmp로 시작하고

↑  
.txt로 끝나는

↑  
임시파일을  
이 디렉토리에 생성합니다.

# 파일 관리에 사용되는 File 클래스

## 임시 파일 생성하기

•• 생성된 임시 파일의 사용 방법

Field Summary

Fields inherited from class [java.io.Writer](#)

[lock](#)

Constructor Summary

**[FileWriter](#)**([File](#) file)  
Constructs a [FileWriter](#) object given a [File](#) object.

**[FileWriter](#)**([File](#) file, boolean append)  
Constructs a [FileWriter](#) object given a [File](#) object.

**[FileWriter](#)**([FileDescriptor](#) fd)  
Constructs a [FileWriter](#) object associated with a file descriptor.

**[FileWriter](#)**([String](#) fileName)  
Constructs a [FileWriter](#) object given a file name.

**[FileWriter](#)**([String](#) fileName, boolean append)  
Constructs a [FileWriter](#) object given a file name with a boolean indicating whether or not to append the data written.

FileWriter 클래스에는 File 타입의 파라미터를 받는 생성자가 있습니다.

```
FileWriter writer = new FileWriter(tmpFile);
```

그 생성자를 이용하여 파일을 열 수 있습니다.

# 파일 관리에 사용되는 File 클래스

## ● 임시 파일 생성하기

- [예제 10-17] 임시 파일을 생성해서 사용하는 프로그램

```
1  import java.io.*;
2  class FileExample2 {
3      public static void main(String args[]) {
4          FileWriter writer = null;
5          try {
6              File file = File.createTempFile("tmp", ".txt", new File("C:\\\\wtemp"));
7              writer = new FileWriter(file);
8              writer.write('자');
9              writer.write('바');
10         }
11         catch (IOException ioe) {
12             System.out.println("임시 파일에 쓸 수 없습니다.");
13         }
14         finally {
15             try {
16                 writer.close();
17             }
18             catch (Exception e) {
19             }
20         }
21     }
22 }
```

임시 파일을 생성합니다.

임시 파일을 열어서 데이터를 씁니다.

임시 파일을 닫습니다.

