

# 16장. 입출력 스트림

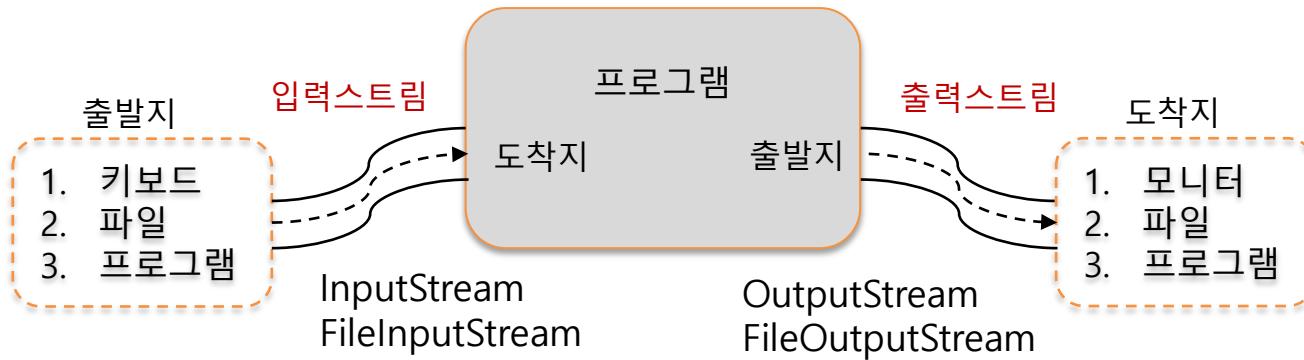


입출력 / 보조스트림



# 입, 출력 스트림

- **스트림**이란? 자료흐름이 물의 흐름과 같다는 뜻이다. 입출력 장치는 매우 다양하기 때문에 프로그램 호환성이 떨어짐
- 입출력 장치와 무관하고 일관성 있게 프로그램을 구현할 수 있도록 일종의 가상통로인 스트림을 제공
- 자료를 읽어 들이려는 소스(source)와 자료를 쓰려는 대상(target)에 따라 각각 다른 **스트림 클래스**를 제공
  - ✓ 입력 스트림 – 어떤 동영상을 재생하기 위해 동영상 파일에서 자료를 읽을 때 사용함
  - ✓ 출력 스트림 – 편집 화면에 사용자가 쓴 글을 파일에 저장할 때는 출력 스트림 사용함



# 입, 출력 스트림

## 바이트 단위 스트림과 문자 단위 스트림

- 바이트 단위 스트림 - 그림, 동영상, 음악 파일등 대부분 파일은 바이트 단위로 읽거나 쓴다.
- 문자 단위 스트림 – 문자만 받고 보낼 수 있도록 특화되어 있다.

구분	바이트 기반 스트림		문자 기반 스트림	
	입력 스트림	출력 스트림	입력 스트림	출력 스트림
최상위 클래스	InputStream	OutputStream	Reader	Writer
하위 클래스	FileInputStream	FileOutputStream	FileReader	FileWriter



# 표준 입출력

## 표준 입출력

- **System 클래스**는 3개의 변수를 가지고 있는데, System.out은 표준 출력용, System.in은 표준 입력용, 빨간색으로 오류를 표시할 때는 System.err을 사용한다.
- out, in, err 모두 정적 변수이다.
- 그 외 java.util 패키지에 있는 **Scanner 클래스** – 문자, 정수, 실수 등을 읽을 수 있다.

자료형	변수 이름	설명
static PrintStream	out	표준 출력 스트림
static InputStream	in	표준 입력 스트림
static OutputStream	err	표준 오류 출력 스트림



# 표준 입출력

## System.in으로 화면에서 문자 1개 입력 받기

```
package fileio.inputstream;

import java.io.IOException;

public class SystemInTest1 {

    public static void main(String[] args){
        System.out.println("한 문자를 입력하고 [Enter]를 누르세요");

        int readByte;
        try {
            readByte = System.in.read();
            System.out.println(readByte);
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

한 문자를 입력하고 [Enter]를 누르세요  
가  
176



# 표준 입출력

## System.in으로 화면에서 문자 여러개 입력 받기

```
public class SystemInTest2 {  
  
    public static void main(String[] args) {  
        System.out.println("여러 개의 문자를 입력하고 [Enter]를 누르세요");  
  
        int readByte;  
        try {  
            while((readByte = System.in.read()) != -1) {  
                //읽을 문자가 없을 때 -1을 반환함.  
                System.out.print((char)readByte);  
            }  
        } catch (IOException e) {  
            e.printStackTrace();  
        }  
    }  
}
```

여러 개의 문자를 입력하고 [Enter]를 누르세요  
corona2019  
corona2019



# 바이트 단위 스트림 - OutputStream

## ➤ OutputStream

- 주요 하위 클래스

스트림 클래스	설명
FileOutputStream	바이트 단위로 파일에 자료를 씁니다.
BufferedOutputStream	기반 스트림에서 자료를 쓸 때 추가 기능을 제공하는 보조 스트림의 상위 클래스이다.

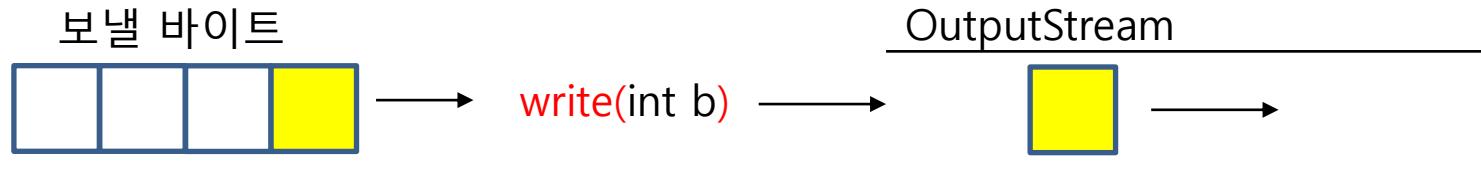
- 주요 메소드

리턴타입	메소드	설명
void	write(int b)	한 바이트를 출력한다.
void	write(byte[ ] b)	b[ ] 배열에 있는 자료를 출력한다.
void	write(byte b[ ], int off, int len)	b[ ] 배열에 off 위치부터 len 개수 만큼 출력
void	close()	출력 스트림과 연결된 대상 리소스를 닫는다.



# FileOutputStream

## ➤ 파일에 바이트 자료 쓰기



```
package outputstream;

import java.io.FileOutputStream;
import java.io.IOException;
import java.io.OutputStream;

public class WriteSample {

    public static void main(String[] args) throws IOException {

        //데이터가 test1.db인 바이트 기반 파일 출력 스트림 생성
        OutputStream os = new FileOutputStream("C:/File/test1.db");

        byte a = 10;
        byte b = 20;
        byte c = 30;

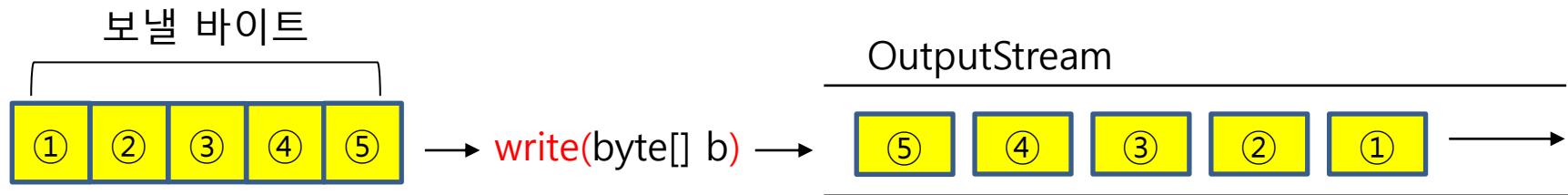
        os.write(a); //파일 쓰기
        os.write(b);
        os.write(c);

        os.flush(); //출력 버퍼에 잔류하는 모든 바이트를 출력
        os.close();
    }
}
```



# FileOutputStream

- **write(byte[] b)** – 파일에 바이트 배열로 쓰기



```
public class WriteSample2 {  
  
    public static void main(String[] args) throws IOException {  
  
        //데이터가 test2.db인 바이트 기반 파일 출력 스트림 생성  
        OutputStream os = new FileOutputStream("C:/File/test2.db");  
  
        byte[] array = new byte[] {10, 20, 30};  
  
        os.write(array);  
  
        os.flush();  
        os.close();  
    }  
}
```



# FileOutputStream

## ➤ write(byte[] b, int off, int len) 메소드

배열 b로 부터 len 개의 바이트를 출력 스트림으로 보낸다.

```
public class WriteSample3 {  
    public static void main(String[] args) throws IOException {  
        //데이터가 test3.db인 바이트 기반 파일 출력 스트림 생성  
        OutputStream os = new FileOutputStream("C:/File/test3.db");  
  
        byte[] array = new byte[] {10, 20, 30, 40, 50};  
  
        os.write(array, 1, 3); //1번 인덱스부터 3개 쓰기  
  
        os.flush();  
        os.close();  
    }  
}
```



# 바이트 단위 스트림 – InputStream

## ➤ **InputStream**

### ▪ 주요 하위 클래스

스트림 클래스	설명
FileInputStream	파일에서 바이트 단위로 자료를 읽는다.
BufferedInputStream	기반 스트림에서 자료를 읽을 때 추가 기능을 제공하는 보조 스트림의 상위 클래스이다.

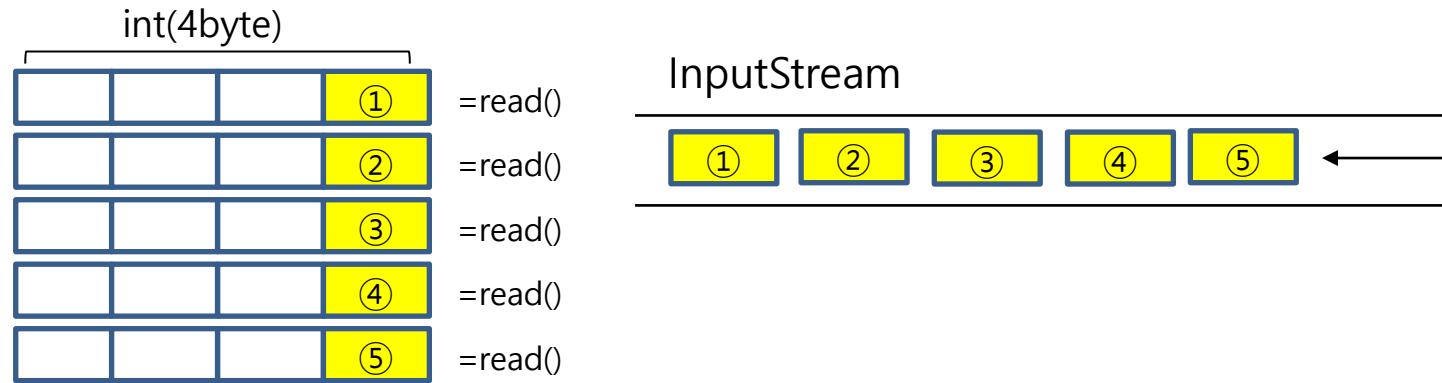
### ▪ 주요 메소드

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



# FileInputStream

## ▶ 파일에서 자료 읽기 – 1byte



# FileInputStream

## ➤ 파일에서 자료 읽기 – 1byte

```
package inputstream;

import java.io.FileInputStream;
import java.io.IOException;
import java.io.InputStream;

public class ReadExample {

    public static void main(String[] args) throws IOException {

        //데이터가 test1.db인 바이트 기반 파일 입력 스트림 생성
        InputStream is = new FileInputStream("C:/file/test1.db");

        int data;
        while((data = is.read()) != -1) { //1byte 씩 읽어 파일 끝에 도달하면
            System.out.println(data);
        }
        is.close();
    }
}
```



# FileInputStream

```
public class ReadExample2 {  
  
    public static void main(String[] args) throws IOException {  
  
        //데이터가 test2.db인 바이트 기반 파일 입력 스트림 생성  
        InputStream is = new FileInputStream("C:/file/test2.db");  
  
        byte[] buffer = new byte[100];  
  
        int data;  
        while((data = is.read(buffer)) != -1) { //배열 길이 만큼 읽어 파일 끝에 도달하면  
            for(int i = 0; i < data; i++) { //읽은 바이트 수만큼 반복  
                System.out.println(buffer[i]);  
            }  
        }  
        is.close();  
    }  
}
```



# FileInputStream

```
public class ReadExample3 {

    public static void main(String[] args) throws IOException {

        //데이터가 test3.db인 바이트 기반 파일 입력 스트림 생성
        InputStream is = new FileInputStream("C:/file/test3.db");

        byte[] buffer = new byte[5];

        int data;
        while((data = is.read(buffer, 2, 3)) != -1) {
            //3byte를 읽고 buffer[2], buffer[3], buffer[4]에 저장
            for(int i = 0; i < buffer.length; i++) { //배열의 전체를 반복
                System.out.println(buffer[i]);
            }
        }
        is.close();
    }
}
```



# 이미지 파일 복사

- **FileInputStream과 FileOutputStream을 이용하여 파일 복사하기**

로컬 디스크 (C:) > File			
	이름	수정한 날짜	유형
			크기
▶	bg0.jpg	2021-02-15 오전 11:13	알씨 JPG 파일
▶	bg1.jpg	2022-07-15 오전 6:01	알씨 JPG 파일
▶	bg2.jpg	2022-07-15 오전 6:06	알씨 JPG 파일
▶	test1.db	2022-07-14 오전 7:28	Data Base File
▶	test2.db	2022-07-09 오후 1:42	Data Base File
▶	test3.db	2022-07-09 오후 1:56	Data Base File



# 이미지 파일 복사

- **FileInputStream과 FileOutputStream을 이용하여 파일 복사하기**

```
public class FileCopyExample {  
  
    public static void main(String[] args) {  
  
        String originFile = "C:/File/bg0.jpg"; //원본 파일  
        String copyFile = "C:/File/bg1.jpg"; //복사 파일  
        FileInputStream fis = null;  
        FileOutputStream fos = null;  
        try {  
            fis = new FileInputStream(originFile);  
            fos = new FileOutputStream(copyFile);  
            int i;  
            while((i=fis.read()) != -1) { //이미지 읽기  
                fos.write(i); //이미지 쓰기  
            }  
        } catch (FileNotFoundException e) {  
            e.printStackTrace();  
        } catch (IOException e) {  
            e.printStackTrace();  
        }finally {  
            try {  
                fis.close();  
                fos.close();  
            } catch (IOException e) {  
                e.printStackTrace();  
            }  
        }  
    }  
}
```



# 이미지 파일 복사

## try ~ with ~ resource 문

close() 메서드를 명시적으로 호출하지 않아도 try 블록내에서 열린 리소스를 자동으로 닫도록 만들수 있다.

try~with~resource 문을 사용하려면 해당 리소스가 AutoCloseable 인터페이스를 구현해야 한다.

java.io

### Class FileInputStream

java.lang.Object  
  java.io.InputStream  
    java.io.FileInputStream

#### All Implemented Interfaces:

Closeable, AutoCloseable



# 이미지 파일 복사

## try ~ with ~ resource 문

```
public class FileCopyExample2 {  
  
    public static void main(String[] args) {  
        //try ~ with ~ resource문 : close()를 명시하지 않아도 됨.  
        long start = 0, end = 0;  
        String originFile = "C:/File/bg0.jpg"; //원본 파일  
        String copyFile = "C:/File/bg2.jpg"; //복사 파일  
  
        try(FileInputStream fis = new FileInputStream(originFile);  
            FileOutputStream fos = new FileOutputStream(copyFile)){  
            start = System.currentTimeMillis(); //수행 전 현재 시간  
            int i;  
            while((i = fis.read()) != -1) {  
                fos.write(i);  
            }  
            end = System.currentTimeMillis(); //수행 후 현재 시간  
        }catch(IOException e) {  
            e.printStackTrace();  
        }  
        System.out.println("파일 복사 소요시간: " + (end-start) + "ms");  
    }  
}
```

파일 복사 소요시간: 2007ms



# 문자 단위 스트림 – Writer

## ❖ Writer

- 주요 하위 클래스

스트림 클래스	설명
FileWriter	문자 단위로 파일에 자료를 씁니다.
OutputStreamWriter	파일에 바이트 단위로 출력한 자료를 문자로 변환해 주는 보조 스트림이다.
BufferedWriter	문자로 쓸 때 배열을 제공하여 한꺼번에 쓸 수 있는 기능을 제공해 주는 보조 스트림이다.

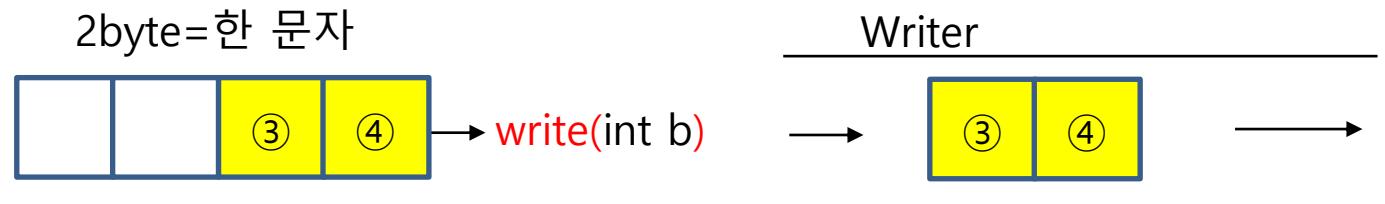
- 주요 메소드

리턴타입	메소드	설명
void	write(int c)	한 문자를 파일에 출력한다.
void	write(char[] b)	문자 배열 buf의 내용을 파일에 출력한다.
void	flush()	파일에 출력하기 전에 자료가 있는 공간(출력버퍼)을 비워 출력
void	close()	출력 스트림과 연결된 대상 리소스를 닫는다.

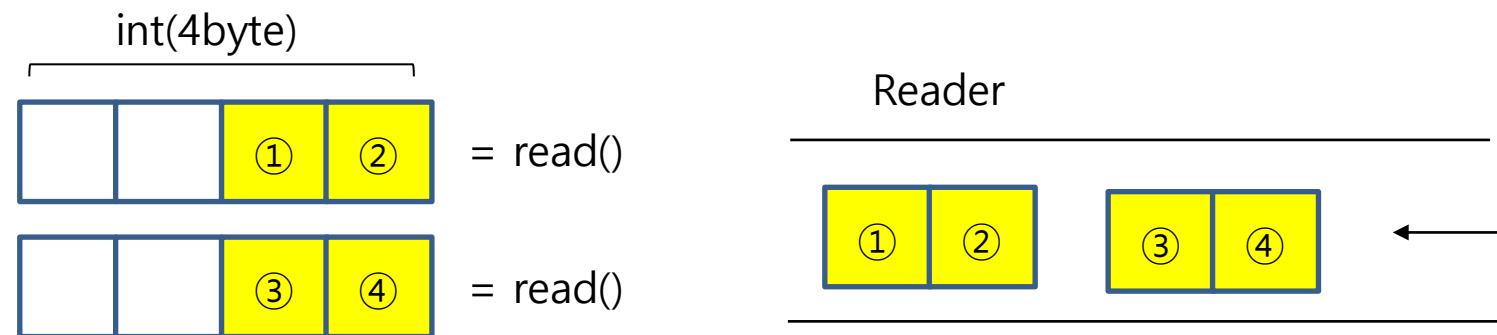


# 문자 단위 스트림 – Writer

## FileWriter 클래스로 쓰기

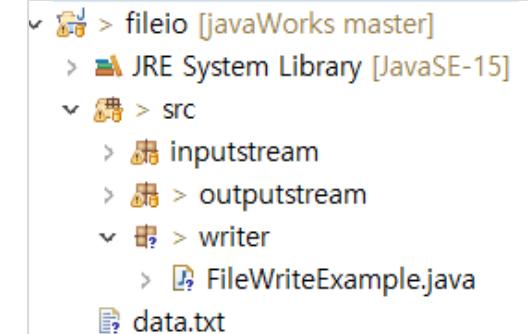


## FileReader 클래스로 읽기



# 문자 단위 스트림 – Writer

```
public class FileWriteExample {  
    public static void main(String[] args) {  
        //문자 파일 기반 출력 스트림 생성  
        try(Writer fw = new FileWriter("data.txt")){  
            fw.write("Hello Java!\\n");      //문자열  
            fw.write("만나서 반갑습니다\\n");  
            //정수가 아닌 아스키 코드, 정수, 실수는 사용할 수 없음  
            fw.write(48);    //문자 0  
            fw.write(10);   //개행 문자(줄바꿈)  
  
            char[] buf = {'s', 'm', 'i', 'l', 'e' };  //배열 출력  
            fw.write(buf);  
            fw.write(10);  
            fw.write(buf, 1, 4);    //1번 인덱스부터 2개 출력  
        }catch(IOException e) {  
            e.printStackTrace();  
        }  
        System.out.println("수행 완료!");  
    }  
}
```



# File 클래스

## ● File 클래스

- 파일 개념을 추상화한 클래스
- 파일의 속성, 경로, 이름 등을 알 수 있음.

```
package writer;

import java.io.File;
import java.io.IOException;

public class FileExample {

    public static void main(String[] args) throws IOException {
        File file = new File("C:/File/newFile.txt");
        file.createNewFile();

        System.out.println(file.isFile());
        System.out.println(file.isDirectory());
        System.out.println(file.getName());          //파일 이름
        System.out.println(file.getPath());          //경로

        file.delete();
    }
}
```

```
true
false
newFile.txt
C:\File\newFile.txt
```



# 문자 단위 스트림 – Reader

## ❖ Reader

- 주요 하위 클래스

스트림 클래스	설명
FileReader	파일에서 문자 단위(2바이트)로 읽는 스트림 클래스이다.
InputStreamReader	바이트 단위로 읽은 자료를 문자로 변환해 주는 보조 스트림
BufferedReader	문자로 읽을 때 배열을 제공하여 한꺼번에 읽을 수 있는 기능을 제공해 주는 보조 스트림이다.

- 주요 메소드

리턴타입	메소드	설명
int	read()	파일로부터 한 문자를 읽는다.
int	read(char[] buf)	파일로부터 buf 배열에 문자를 읽는다.
void	close()	스트림과 연결된 파일 리소스를 닫는다.



# 문자 단위 스트림 – Reader

## ❖ FileReader 클래스로 읽기

```
public class FileReaderExample {  
    public static void main(String[] args) {  
        //문자 파일 기반 입력 스트림 생성  
        try(Reader fr = new FileReader("data.txt")){  
            int i;  
            while((i = fr.read()) != -1) {  
                System.out.print((char)i);  
            }  
        }catch(IOException e) {  
            e.printStackTrace();  
        }  
    }  
}
```



# 보조 스트림

보조 스트림

- 실제 읽고 쓰는 스트림이 아닌 보조적인 기능을 추가하는 스트림.
  - 생성자의 매개변수로 또 **기반스트림**을 가짐
  - 데코레이터(decorator) 패턴이라고 함



# 보조 스트림 – InputStreamReader

## InputStreamReader와 OutputStreamWriter

- 바이트 자료만 입력되는 스트림에서 문자로 변환해 준다.
- **System.in**이나 네트워크 **socket** 통신을 할 때 쓰인다.
- 입출력 기능이 없으므로 다른 입출력 스트림을 포함한다.

생성자	설명
InputStreamReader (FileInputStream in)	생성자의 매개변수로 FileInputStream을 받는다.
OutputStreamReader (FileOutputStream out)	생성자의 매개변수로 FileOutputStream을 받는다.



# 보조 스트림 – InputStreamReader

## InputStreamReader 클래스로 문자 읽기

```
package iostream.decorator;

import java.io.FileInputStream;

public class InputStreamReaderTest {

    public static void main(String[] args) {
        //보조스트림은 기반스트림을 매개변수로 사용
        try(InputStreamReader isr =
            new InputStreamReader(new FileInputStream("reader.txt"))){
            int readByte;
            while((readByte=isr.read()) != -1) {
                System.out.print((char)readByte);
            }
        }catch(Exception e) {
            e.printStackTrace();
        }
    }
}
```

Module [java.base](#)

Package [java.io](#)

Class **InputStreamReader**

[java.lang.Object](#)

[java.io.Reader](#)

[java.io.InputStreamReader](#)

All Implemented Interfaces:

[Closeable](#), [AutoCloseable](#), [Readable](#)

새해 복많이 받으세요!!



# 보조 스트림 – InputStreamReader

## OutputStreamWriter 클래스로 문자 쓰기

- 바이트 자료만 출력되는 스트림에서 문자로 변환해 파일을 생성.

```
public class OutputStreamWriterTest {  
  
    public static void main(String[] args) {  
        //보조스트림과 기반스트림 함께 사용하기2  
        try(FileOutputStream fos = new FileOutputStream("writer.txt");  
            OutputStreamWriter osw = new OutputStreamWriter(fos)){  
            osw.write("지금까지 정말 재미있게 자바 공부했어요!!");  
        }catch(Exception e){  
            e.printStackTrace();  
        }  
        System.out.println("Done");  
    }  
}
```

세미콜론으로 구분

지금까지 정말 재미있게 자바 공부했어요!!



# 보조 스트림 – Buffered 스트림

## Buffered 스트림

- 입출력이 한 바이트나 문자 단위로 이루어지면 그만큼 프로그램 수행 속도가 느려진다.
- Buffered 스트림은 내부적으로 8,192바이트 크기의 배열을 가지고 있으며 더 빠르게 입출력을 수행하는 버퍼링 기능을 제공한다.

스트림 클래스	설명
BufferedInputStream (InputStream in)	바이트 단위로 읽는 스트림에 버퍼링 기능을 제공
BufferedOutputStream (FileOutputStream out)	바이트 단위로 출력하는 스트림에 버퍼링 기능을 제공
BufferedReader	문자 단위로 읽는 스트림에 버퍼링 기능을 제공
BufferedWriter	문자 단위로 출력하는 스트림에 버퍼링 기능을 제공



# 보조 스트림 – Buffered 스트림

## BufferedInputStream과 BufferedOutputStream으로 파일 복사하기

```
public class BufferedStreamTest {  
  
    public static void main(String[] args) {  
        long start = 0;  
        long end = 0;  
        String originFile = "C:/javaDev/day12/feature.png";  
        String copyFile = "C:/ncsTest/feature3.png";  
  
        try(FileInputStream fis = new FileInputStream(originFile);  
            FileOutputStream fos = new FileOutputStream(copyFile);  
            BufferedInputStream bis = new BufferedInputStream(fis);  
            BufferedOutputStream bos = new BufferedOutputStream(fos)){  
            start = System.currentTimeMillis();  
            int i;  
            while((i=bis.read()) != -1) {  
                bos.write(i);  
            }  
            end = System.currentTimeMillis();  
        }catch(IOException e) {  
            e.printStackTrace();  
        }  
        System.out.println("파일 복사 소요시간 : " + (end-start) + "milliseconds");  
    }  
}
```

파일 복사 소요시간 : 23 milliseconds



# 보조 스트림 – Buffered 스트림

## BufferedReader 클래스로 파일 읽기

```
public class BufferedReaderTest {  
    public static void main(String[] args) {  
        String fileName = "C:/JavaWork4/Chapter17/reader.txt";  
        try(FileReader fr = new FileReader(fileName);  
            BufferedReader br = new BufferedReader(fr)){  
            //System.out.println((char)br.read()); //read() - int형 반환  
            System.out.println(br.readLine()); // readLine()-string형 반환  
  
            String line = null;  
            while((line=br.readLine()) != null) {  
                System.out.println(line);  
            }  
        }catch(Exception e) {  
            e.printStackTrace();  
        }  
    }  
}
```

새해 복많이 받으세요!!  
2021년은 소띠해(신축년)  
Corona를 잘 피하자...

### readLine

```
public String readLine() throws IOException
```

Reads a line of text. A line is considered to be terminated by any one of a line feed ('\n'), a carriage return ('\r'), a carriage return followed immediately by a line feed, or by reaching the end-of-file (EOF).

줄바꿈문자(\n, \r)를 만날때까지 읽을 수 있다.



# 보조 스트림 – BufferedWriter

## BufferedWriter 클래스로 파일 쓰기

```
public class BufferedWriterTest {  
    public static void main(String[] args) {  
        try(FileWriter fr = new FileWriter("vegetable.txt"));  
            BufferedWriter bw = new BufferedWriter(fr)){  
                bw.write("Onion");  
                bw.newLine();  
                bw.write("Potato");  
                bw.newLine();  
                bw.write("Garlic");  
                bw.newLine();  
                bw.write(65); //65 - 아스키 코드값  
                int num = 66;  
                bw.write(num);  
            }catch(Exception e) {  
                e.printStackTrace();  
            }  
            System.out.println("Done");  
        }  
    }
```

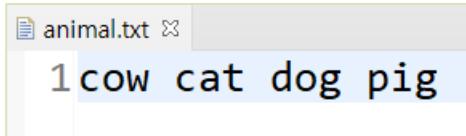
Module java.base  
Package java.io  
**Class BufferedWriter**  
java.lang.Object  
    java.io.Writer  
        java.io.BufferedWriter  
**All Implemented Interfaces:**  
Closeable, Flushable, Appendable, AutoCloseable  
  
public class BufferedWriter  
extends Writer

Onion  
Potato  
Garlic  
AB



# 보조 스트림 – BufferedReader

BufferedReader 로  
배열을 이용하여 읽어오기



```
public class BufferedReaderTest2 {  
    public static void main(String[] args) {  
        try(FileReader fr = new FileReader("animal.txt"));  
            BufferedReader br = new BufferedReader(fr)){  
                String line = null;  
                String[] animal = null;  
                while((line=br.readLine()) != null) {  
                    animal = line.split(" "); //공백문자로 구분  
                }  
                System.out.println(animal[0]);  
  
                //전체 데이터 가져오기  
                for(int i=0; i<animal.length; i++) {  
                    System.out.println(animal[i]);  
                }  
  
                //랜덤하게 가져오기  
                int rand = (int)(Math.random()*animal.length);  
                System.out.println(animal[rand]);  
            }catch(Exception e) {  
                e.printStackTrace();  
            }  
    }  
}
```



# 보조 스트림 – BufferedReader

## BufferedReader와 BufferedWriter로 성적 처리하기

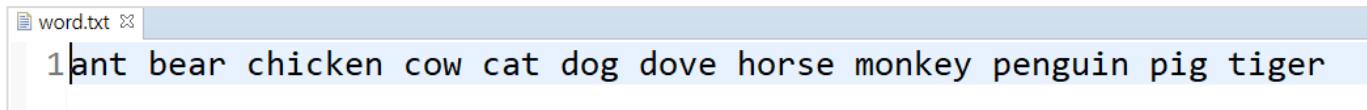
```
public class BufferedReaderScore {
    public static void main(String[] args) {
        try(FileReader fr = new FileReader("score.txt");
            FileWriter fw = new FileWriter("score2.txt"));
        BufferedReader reader = new BufferedReader(fr);
        BufferedWriter writer = new BufferedWriter(fw)){
            String line = null;
            int sum = 0; //총점
            double avg = 0.0; //평균
            while((line=reader.readLine()) != null) {
                String[] data = line.split(" "); //공백문자로 분리
                sum = Integer.parseInt(data[1]) + Integer.parseInt(data[2]);
                avg = (double)sum / 2;
                line = line + " " + sum + " " + String.format("%.1f", avg);
                //System.out.println(line); - 콘솔에 쓰기
                writer.write(line); //파일에 쓰기
                writer.newLine();
            }
        }catch(Exception e) {
            System.out.println(e.getMessage());
        }
        System.out.println("처리 완료!!!");
    }
}
```

	1	kim	91	80	171	85.5
2	lee	70	59	129	64.5	
3	han	60	85	145	72.5	



# 보조 스트림 – BufferedReader

## 영어 타자 연습 게임



```
public class EnglishTypingGame {  
  
    public static void main(String[] args) {  
        try(FileReader fr = new FileReader("word.txt");  
            BufferedReader br = new BufferedReader(fr)){  
            String line;  
            String[] word = null;  
            while((line=br.readLine()) != null) {  
                word = line.split(" ");  
            }  
            int n= 0; //정답 개수  
            long milliseconds = 0; //소요 시간
```

타자 연습 게임 - 준비되면 [Enter]  
문제 1  
horse  
horse  
통과!!  
문제 2  
penguin  
penguin  
통과!!  
문제 3  
dog



# 보조 스트림 – BufferedReader

```
System.out.println("타자 연습 게임 - 준비되면 [Enter]");
Scanner sc = new Scanner(System.in);
sc.nextLine();
milliseconds = System.currentTimeMillis(); //시작 시간

for(int i=1; i<11; i++) {
    System.out.println("문제 " + i);
    int rand = (int) (Math.random()*word.length);
    String question = word[rand];
    System.out.println(question); //문제

    String answer = sc.nextLine(); //단어 입력

    if(answer.equals(question)) {
        System.out.println("통과!!");
        n++;
    }else {
        System.out.println("오타! 다시 도전!");
    }
}
milliseconds = System.currentTimeMillis() - milliseconds;
sc.close();
System.out.println("정답 개수 : " + n);
System.out.println("게임 소요 시간 " + (float)milliseconds/1000 + "초");
br.close();
}catch(IOException e) {
    e.printStackTrace();
}
```



# 보조스트림 – DataStream

## DataInputStream과 DataOutputStream

지금까지의 공부한 스트림은 사람이 읽고 쓰는 텍스트 또는 이미지 형식의 자료를 다루었으나, DataStream은 메모리에 저장된 0, 1상태를 그대로 읽거나 쓴다.  
즉, 자료형의 크기가 그대로 보존된다.

메서드	설명
byte readByte()	1바이트를 읽어 반환
char readChar()	한 문자를 읽어 반환
int readInt()	4바이트를 읽어 정수값 반환
double readDouble()	8바이트를 읽어 실수값 반환
String readUTF()	문자열을 읽어 반환

## DataInputStream

## DataOutputStream

메서드	설명
void writeByte(int v)	1바이트의 자료 출력
void writeChar(int v)	2바이트의 자료 출력
void writeInt(int v)	4바이트의 자료 출력
void writeDouble(double v)	8바이트의 자료 출력
void readUTF(String str)	문자열을 출력



# DataStream

```
public class DataOutputStreamTest {  
  
    public static void main(String[] args) {  
        //자료형이 그대로 유지됨 - 0과 1인 기계어  
        //보조스트림(기반스트림) - 기반스트림이 보조스트림의 생성자임  
        try(FileOutputStream fos = new FileOutputStream("data2.txt");  
            DataOutputStream dos = new DataOutputStream(fos)){  
            dos.writeByte('A');           //영문 한글자  
            dos.writeChar('가');          //한글 한글자  
            dos.writeInt(48);            //정수  
            dos.writeFloat(2.54F);        //실수  
            dos.writeUTF("감사합니다.");   //문자열  
  
        }catch(IOException e) {  
            e.printStackTrace();  
        }  
    }  
}
```

A♦ 0@"\ 埃 맷궗♦ 빙♦ 뜻♦ 땅.



# DataStream

```
public class DataInputStreamTest {  
  
    public static void main(String[] args) {  
        //자료형이 그대로 유지됨  
        //보조스트림(기반스트림) - 기반스트림이 보조스트림의 생성자임  
        try(FileInputStream fis = new FileInputStream("data2.txt");  
            DataInputStream dis = new DataInputStream(fis)){  
            System.out.println(dis.readByte());  
            System.out.println(dis.readChar());  
            System.out.println(dis.readInt());  
            System.out.println(dis.readFloat());  
            System.out.println(dis.readUTF());  
        }catch(IOException e) {  
            e.printStackTrace();  
        }  
    }  
}
```



# 직렬화(Serialization)

## Serialization(직렬화)

- 인스턴스의 어느 순간 상태를 그대로 저장하거나, 네트워크를 통해 전송하기 위해 연속 스트림으로 만드는 것을 직렬화라 한다.
- 역직렬화는 저장된 내용이나 전송받은 내용을 다시 복원하는 것이다.
- 보조 스트림인 **ObjectInputStream**과 **ObjectOutputStream** 사용한다.
- 주요 메서드로는 `writeObject()`와 `readObject()`가 있다.
- `serialVersinUID`를 사용하여 버전 관리 (객체를 역직렬화할 때 직렬화할 때의 클래스 상태가 다르면 오류가 발생.)

생성자	설명
<code>ObjectInputStream(InputStream in)</code>	<code>InputStream</code> 을 생성자의 매개변수로 받아 <code>ObjectInputStream</code> 을 생성합니다.
<code>ObjectOutputStream(OutputStream out)</code>	<code>OutputStream</code> 을 생성자의 매개변수로 받아 <code>ObjectOutputStream</code> 을 생성합니다.



# 직렬화(Serialization)

## Serialization 예제

```
package iostream.serialization;
import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.ObjectInputStream;
import java.io.ObjectOutputStream;
import java.io.Serializable;

class Person implements Serializable{
    private static final long serialVersionUID = 12345L;

    String name;
    String job;

    public Person() {}

    public Person(String name, String job) {
        this.name = name;
        this.job = job;
    }

    public String toString() {
        return name + "," + job;
    }
}
```

Serializable 인터페이스 구현

java.io.NotSerializableException: iostream.serialization.Person  
at java.base/java.io.ObjectOutputStream.writeObject0(Obj...  
at java.base/java.io.ObjectOutputStream.writeObject(Obj...



# 직렬화(Serialization)

## 직렬화와 역직렬화

```
public class SerializationTest {
    public static void main(String[] args) {
        //직렬화
        Person personSon = new Person("손정의", "대표이사");
        Person personJang = new Person("장그래", "부장");

        try(FileOutputStream fos = new FileOutputStream("serial.out");
            ObjectOutputStream oos = new ObjectOutputStream(fos)){
            oos.writeObject(personSon); //객체를 파일에 씀
            oos.writeObject(personJang);
        }catch(Exception e) {
            e.printStackTrace();
        }
    }

    //복원 - 역직렬화
    try(FileInputStream fis = new FileInputStream("serial.out");
        ObjectInputStream ois = new ObjectInputStream(fis)){
        Person p1 = (Person) ois.readObject(); //Object형 -> Person 변환
        Person p2 = (Person) ois.readObject();

        System.out.println(p1);
        System.out.println(p2);
    }catch(Exception e) {
        e.printStackTrace();
    }
}
```

↳ sr iostream.serialization.Person  
09 ↳ L ↳ jobt ↳ java/lang/String;L

손정의, 대표이사  
장그래, 부장

