

컴퓨터 프로그래밍 (Computer Programming)

이 선 순



12. 파일 입출력



목차

01. 예외 처리

02. 표준입출력

03. 파일입출력

03

파일 입출력

03 파일 입출력

■ 파일 입출력

- 파일 입출력 메소드는 입력과 출력을 표준 입출력 장치가 아닌 파일로 처리하는 메소드
- 표준 입출력과 파일 입출력

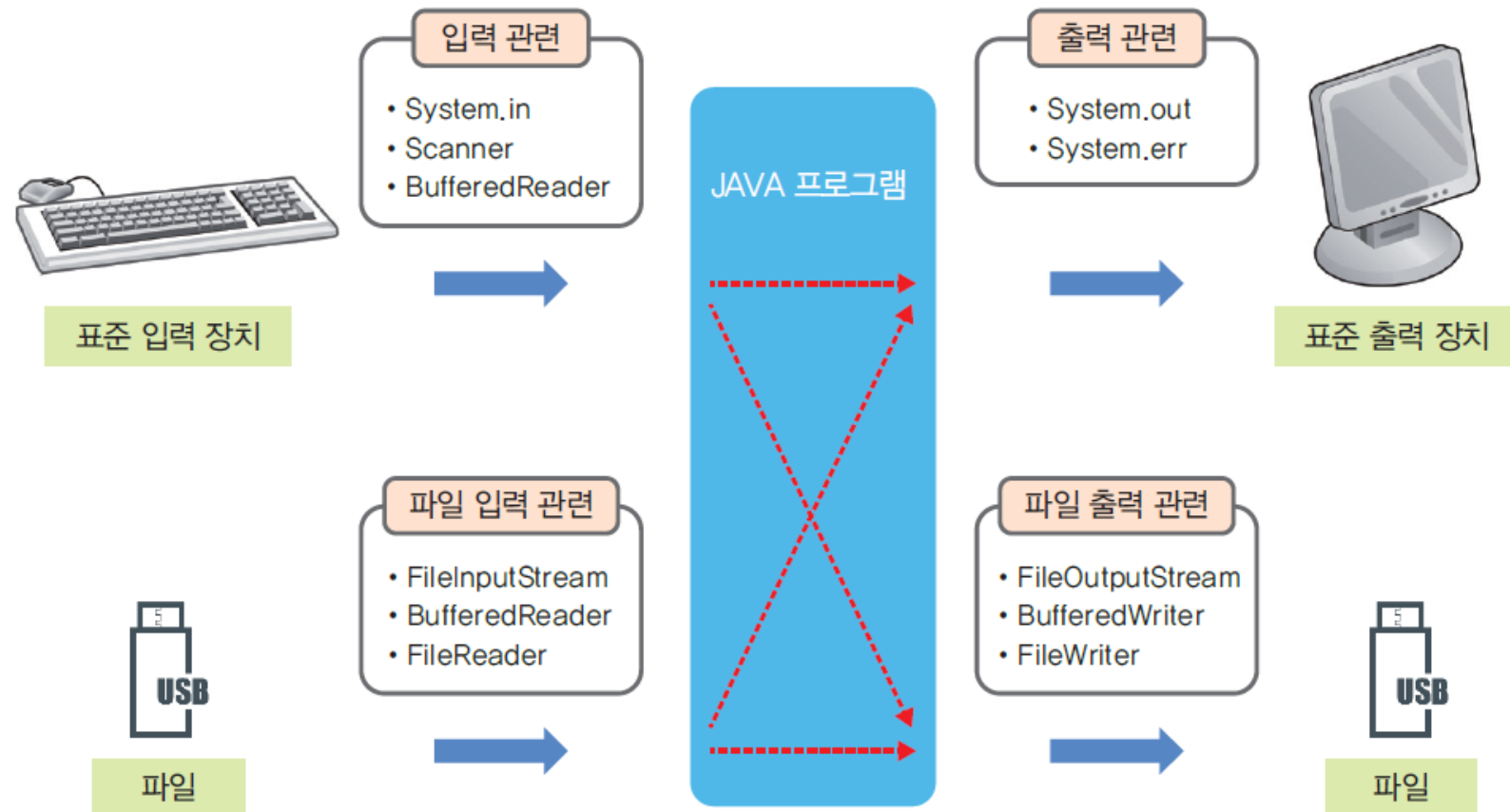


그림 10-13 표준 입출력과 파일 입출력

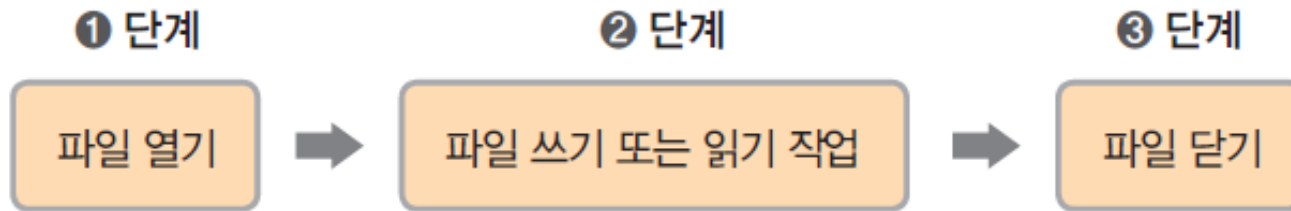
■ 스트림(stream)

- 스트림은 데이터를 송수신하기 위한 통로의 개념
- 입력 혹은 출력, 한쪽 방향으로만 진행됨
- 스트림은 1바이트를 처리하는 바이트 스트림과 2바이트를 처리하는 문자 스트림으로 나뉜다.
- [그림 10-13]에서 `FileInputStream`, `FileOutputStream`은 바이트 스트림에 해당하고, `BufferedReader`, `BufferedWriter`, `FileReader`, `FileWriter`는 문자 스트림에 해당한다.
- 한글은 2바이트이므로 문자 스트림을 사용하는 것이 더 편리하다.

03 파일 입출력

■ 파일 입출력의 기본 과정

- 표준입출력과 파일입출력의 차이점 : 사용하는 클래스와 메소드가 다름, 각 메소드의 사용법은 크게 다르지 않음
- 표준입출력 장치(키보드, 화면)은 항상 준비되어 있어 Scanner 클래스나 System.out.printf()와 같은 메소드로 바로 사용가능함
- 파일입출력을 위해서는 반드시 두가지 작업이 추가로 이루어져야 함
 - 파일을 사용하기 전의 '파일열기' 작업과 파일 사용이 끝난 후의 '파일닫기' 작업을 추가로 해야함
- 파일 입출력의 세단계



03 파일 입출력

■ 파일 입출력의 기본 과정

① 파일 열기(1단계)

- 파일을 열기 위해서는 관련 클래스에서 변수를 선언하고 파일명을 지정해야 한다.
- 파일명 지정시 경로도 정확하게 지정한다.

```
읽기용 : FileInputStream 변수명 = new FileInputStream("파일명");  
쓰기용 : FileOutputStream 변수명 = new FileOutputStream("파일명");
```

② 파일 처리(2단계)

- 데이터를 쓰거나 파일로부터 데이터를 읽어올 수 있는 상태

③ 파일 닫기(3단계)

- 파일과 관련된 모든 작업이 끝나면 파일을 정상적으로 닫아야 한다. 이때 닫는 변수는 1단계에서 선언한 변수 이름이다.

```
변수명.close( );
```


03 파일 입출력

■ 파일을 이용한 입력

- 파일 입력과 표준 출력
- 지금까지 입력방식은 키보드 입력이었음. 데이터가 간단하면 사용에 어려움이 없음
- 입력해야 할 데이터가 수십페이지라면? 여러 사람에게 입력데이터를 나눠주고 프로그램 실행 시 직접 입력한다면?
→ 시간도 많이 소요되고 사람마다 입력하는 방법이 달라서 동일한 결과가 나오지 않음. 파일에 데이터를 담아서 필요할 때 불러서 사용하는 것이 훨씬 효율적임



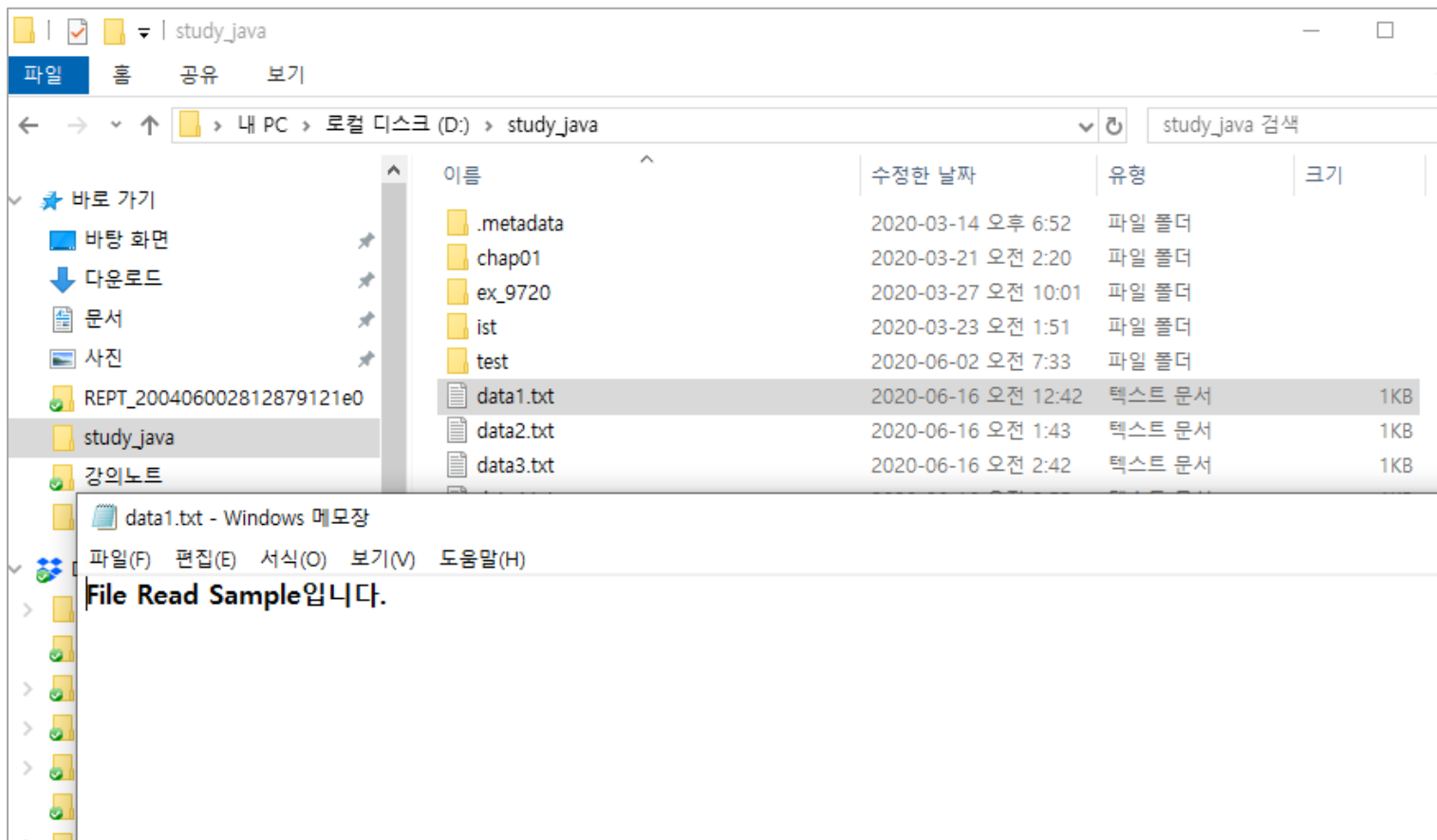
그림 10-14 파일 입력과 표준 출력

03 파일 입출력

■ 파일을 이용한 입력

■ FileInputStream을 이용하여 1바이트씩 읽어들이기

- FileInputStream 클래스를 사용하면 파일의 내용을 1바이트씩 읽음. 1바이트씩 읽어오는 메소드는 read()
- 메모장을 실행하여 'File Read Sample입니다.'라는 문장을 한 줄 쓰고 파일명을 'd:\study_java\data1.txt'로 하여 저장



03 파일 입출력

■ 파일을 이용한 입력

- [실습10-9] FileInputStream을 이용하여 1바이트씩 읽어들이기

```
1 import java.io.FileInputStream;
2
3 public class Ex10_09 {                               FileInputStream의 예외를 처리하기 위해 throws Exception문을 추가함
4     public static void main(String[] args) throws Exception {
5
6         FileInputStream fis = new FileInputStream("d:/study_java/data1.txt");
7                                     FileInputStream을 준비하고 파일을 연다
8         int ch;
9
10        while ((ch = fis.read()) != -1) {               파일에서 문자 하나를 read()로 읽어온다. 파일의 끝일 경우 -1을 반환한다
11
12            System.out.print((char) ch);               문자를 출력한다
13
14            fis.close();                               파일의 모든 내용을 처리한 후 파일을 닫는다
15        }
16    }
17 }
18
```

Problems @ Javadoc Declaration Console

<terminated> Ex10_09 [Java Application] C:\Program Files\Java\jdk-
File Read Sample?????????.

03 파일 입출력

■ 파일을 이용한 입력

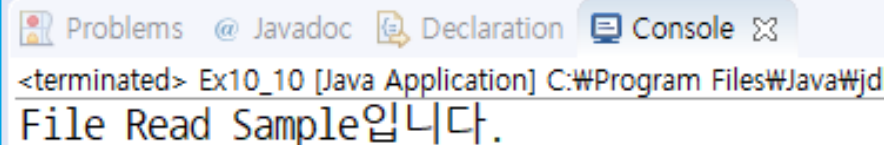
- [실습 10-9]는 파일 처리의 핵심을 알려주는 예제이니 잘 익히기 바람.
- 4행 : 파일을 읽을 때 발생할 예외 처리를 위해 throws Exception 문을 추가.
- 6행 : 입력을 위한 FileInputStream형의 fis 변수를 선언하면서 동시에 d:\wstudy_java\data1.txt 파일을 오픈.
FileInputStream이 읽기 모드로 열림
- 10, 12행 : 파일의 끝까지 1바이트씩 읽음. 만약 파일의 끝을 만나면 read() 메소드가 -1을 반환하므로 while 문을 빠져
나옴
- 14행 : 파일 사용이 끝났으므로 파일 닫음.
read()는 1바이트씩 읽으므로 2바이트를 차지하는 한글은 잘라서 읽고 바로 출력했기 때문에 깨짐
- **TIP** : 파일 경로도 문자열이므로 폴더를 구분하기 위해 /를 사용한다면 하나만 넣어도 되지만 \를 사용하려면 \\와
같이 2개를 넣어야 함

03 파일 입출력

■ 파일을 이용한 입력

- [실습10-10] FileInputStream을 이용하여 1바이트씩 읽어들이기

```
1 import java.io.FileInputStream;
2
3 public class Ex10_10 {
4     public static void main(String[] args) throws Exception {
5         FileInputStream fis = new FileInputStream("d:/study_java/data1.txt");
6         int ch;
7
8         byte[] bt = new byte[1024]; 1024 크기의 배열 bt를 선언한다
9
10        int i = 0;  배열의 첨자로 사용할 i를 선언한다
11
12        while ((ch = fis.read()) != -1) {
13            bt[i] = (byte) ch;  읽어온 1바이트를 배열에 저장한다
14            i++;  배열의 첨자를 1씩 증가시킨다.
15        }
16
17        System.out.print(new String(bt)); Byte 형식의 배열을 string 형식으로 출력한다.
18
19        fis.close();
20    }
21 }
```



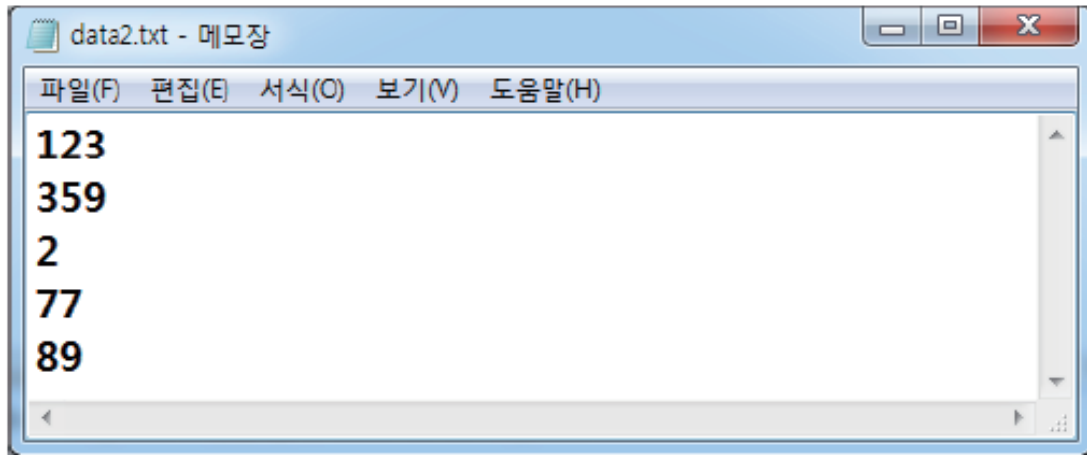
The screenshot shows the 'Console' tab of a Java IDE. It displays the output of the program: "<terminated> Ex10_10 [Java Application] C:\Program Files\Java\jdk... File Read Sample입니다."

- 8행에서 크기를 충분하게 1024로 설정한 배열을 준비하고, 13행에서 그 배열에 읽어온 1바이트를 차례대로 저장함
- 17행에서 문자열 형식으로 한꺼번에 출력했기 때문에 한글이 잘 출력됨

03 파일 입출력

■ Scanner를 활용한 파일 읽기

- 여러 줄에 숫자가 쓰인 파일의 합계를 내는 코드를 작성하기.
- 다음과 같이 다섯 줄의 숫자를 메모장에 쓰고 파일명을 'd:\Wstudy_java\data2.txt'로 하여 저장



03 파일 입출력

■ Scanner를 활용한 파일 읽기

■ [실습10-12] Scanner를 이용한 입력

```
1 import java.io.File;
2 import java.util.Scanner;
3
4 public class Ex10_12 {
5
6     public static void main(String[] args) throws Exception {
7
8         Scanner sc = new Scanner(new File("d:/study_java/data2.txt"));
9
10        int hap = 0;
11
12        while (sc.hasNextLine())
13            hap += sc.nextInt();
14
15        System.out.println("합계 : " + hap);
16
17        sc.close();
18    }
19 }
20 }
```

File을 지정해서 Scanner클래스의 변수를 선언한다

합계 변수를 선언하고 초기화한다

파일의 마지막까지 무한 반복한다
파일 내용을 정수형으로 읽어서 누적합계를 계산한다

누적한 합계를 출력한다

Problems @ Javadoc Declaration Console

<terminated> Ex10_12 [Java Application] C:\Program Files\Java\W

합계 : 650

■ Scanner를 활용한 파일 읽기

- [실습10-12] Scanner를 이용한 입력
- 8행 : Scanner 형의 변수를 선언할 때, 앞에서는 키보드로 입력받기 위해 System.in을 사용했으나 파일로 입력받기 위해서는 File 클래스를 사용함
- 12행 : while문의 hasNextLine() 은 다음 행이 있는지 미리 파악하여 다음 행이 있으면 true를 반환함.
다음 행이 있으면 14행을 실행하여 정수를 읽어들이고 파일에서 읽어들이는 값을 hap에 계속 누적해서 합계를 구함

03 파일 입출력

■ 파일을 이용한 출력

- 표준 입력과 파일 출력
- 화면의 출력결과는 한번 볼 수 있을 뿐 저장되지 않음
- 출력결과를 파일에 저장하는 방법 : FileOutputStream과 BufferedWriter클래스를 활용함

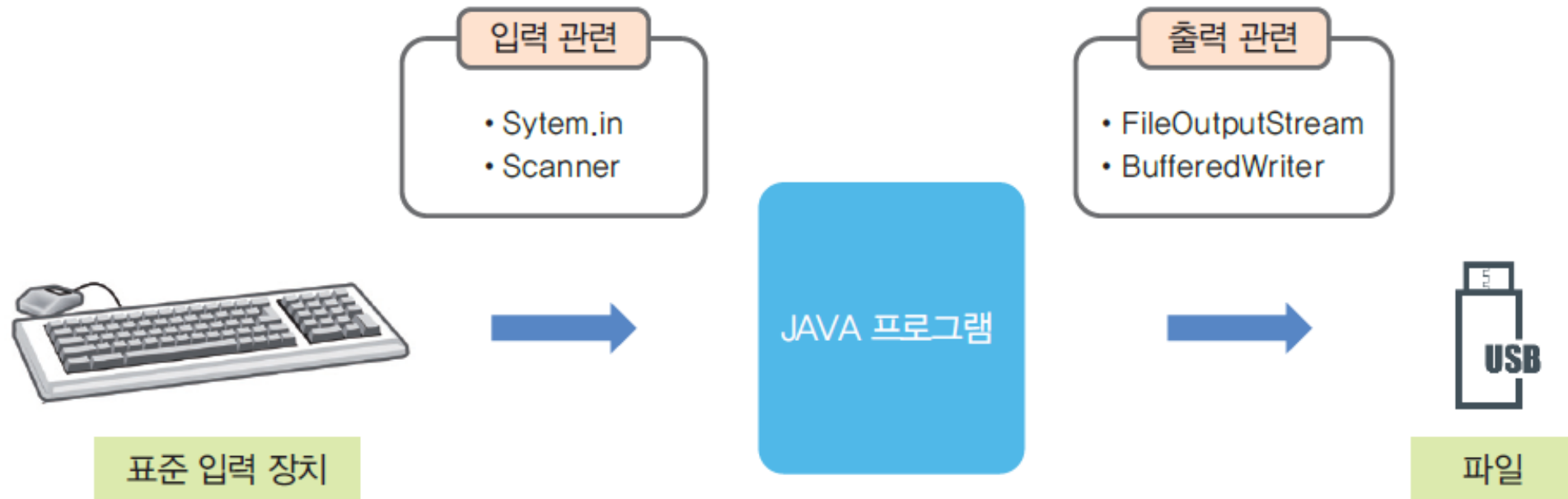


그림 10-19 표준 입력과 파일 출력

03 파일 입출력

■ 파일을 이용한 출력

- `FileOutputStream`을 이용하여 1바이트씩 파일에 쓰기 : `write()` 메소드 사용
- [실습10-13] 파일을 이용한 출력1

```
1 import java.io.FileOutputStream;
2
3 public class Ex10_13 {
4     public static void main(String[] args) throws Exception {
5         FileOutputStream fos = new FileOutputStream("d:/study_java/data3.txt");
6
7         int ch;
8
9         while ((ch = System.in.read()) != 13)
10             fos.write((byte) ch);
11
12         fos.close();
13     }
14 }
15
16
17
18
```

`FileOutputStream`의 예외를 처리하기 위해 `throws Exception`문을 추가함

`FileOutputStream`을 준비하고 파일을 연다

`Enter`(아스키코드 값 : 13)를 입력할때까지 키보드에서 문자하나를 `read()`로 읽어온다

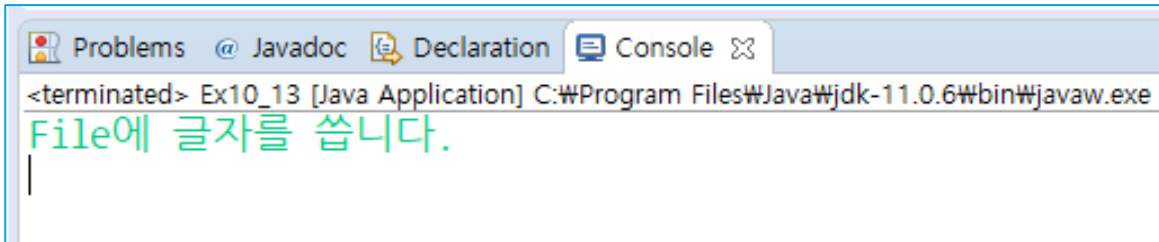
입력한 문자를 파일에 쓴다

파일의 모든 내용을 처리한 후 파일을 닫는다

03 파일 입출력

■ 파일을 이용한 출력

■ [실습10-13] 파일을 이용한 출력1



```
<terminated> Ex10_13 [Java Application] C:\Program Files\Java\jdk-11.0.6\bin\javaw.exe
File에 글자를 씁니다.
```

- 5행 : 입력을 위한 `FileOutputStream`형의 `fos` 변수를 선언하면서 동시에 `d:\wstudy_java\data3.txt` 파일 오픈.
`FileOutputStream`이 쓰기 모드로 열림
- 11, 12행 : 키보드로 1바이트씩 입력받는데, 만약 `[Enter]`의 아스키코드인 13을 만나면 `while` 문을 빠져나옴.
읽어온 문자는 `byte`형으로 변환해서 파일에 쓴다.
- 15행 : 파일의 사용이 끝났으므로 파일 닫음. 파일 탐색기에서 결과 파일을 확인해보면 잘 저장되어 있음

03 파일 입출력

■ 파일을 이용한 출력

■ [실습10-13] 파일을 이용한 출력1

data3.txt - Windows 메모장

파일(F) 편집(E) 서식(O) 보기(V) 도움말(H)

File에 글자를 씁니다.

내 PC > 로컬 디스크 (D:) > study_java

study_java 검색

이름	수정한 날짜	유형	크기
.metadata	2020-03-14 오후 6:52	파일 폴더	
chap01	2020-03-21 오전 2:20	파일 폴더	
ex_9720	2020-03-27 오전 10:01	파일 폴더	
ist	2020-03-23 오전 1:51	파일 폴더	
test	2020-06-02 오전 7:33	파일 폴더	
data1.txt	2020-06-16 오전 12:42	텍스트 문서	1KB
data2.txt	2020-06-16 오전 1:43	텍스트 문서	1KB
data3.txt	2020-06-16 오전 1:59	텍스트 문서	1KB
HelloJava.java	2020-03-21 오후 5:06	JAVA 파일	1KB
introduce.java	2020-03-21 오후 5:06	JAVA 파일	1KB
목차.txt	2020-03-30 오전 12:07	텍스트 문서	1KB

03 파일 입출력

■ 파일을 이용한 출력

- `FileWriter`를 이용하여 파일에 한 줄씩 쓰기 : `FileOutputStream` 클래스는 바이트 단위로 파일에 쓰기를 하지만 `FileWriter`는 문자열을 직접 파일에 쓸 수 있음

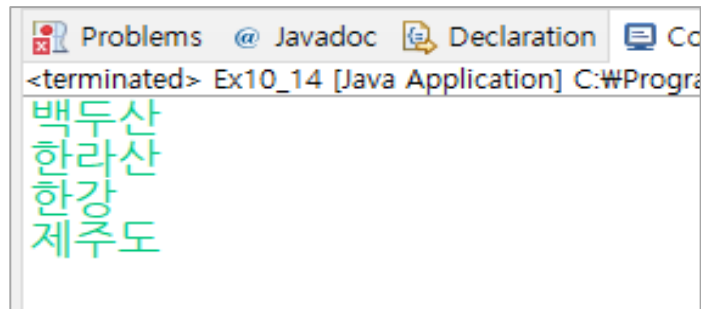
- [실습10-14] 파일을 이용한 출력2

```
1 import java.io.FileWriter;
2 import java.util.Scanner;
3
4 public class Ex10_14 {
5     public static void main(String[] args) throws Exception {
6
7         Scanner sc = new Scanner(System.in);   키보드로 입력받기 위해 Scanner 클래스를 준비함
8
9         FileWriter fw = new FileWriter("D:/study_java/data4.txt");   FileWriter 형식으로 fw변수를 준비함
10        String str;
11
12        while (!(str = sc.nextLine()).equals(""))   입력한 행이 비어 있는 행이 아니면 14행을 처리한다.
13            {                                       즉, 그냥 [Enter]를 누르면 입력을 종료한다.
14                fw.write(str + "\r\n");   입력한 문자열과 "WrWn"을 붙인다.
15            }
16        fw.close();
17    }
18 }
```

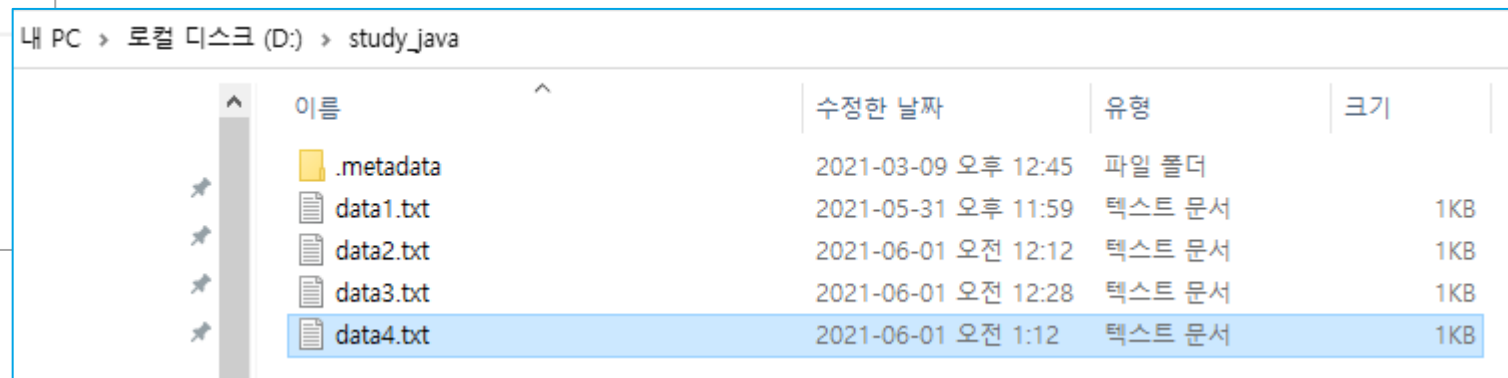
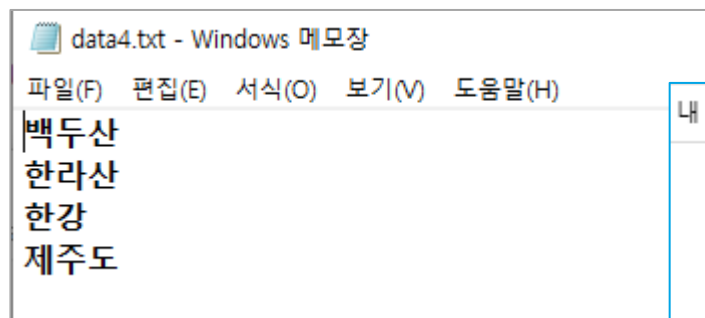
03 파일 입출력

■ 파일을 이용한 출력

■ [실습10-14] 파일을 이용한 출력2



- 12행 `!(str = sc.nextLine()).equals(""))`의 부분은 `sc.nextLine()`으로 키보드에서 한 행을 읽어들이고 그 결과를 `str`에 저장함. 그런데 여기서 [Enter]를 그냥 누르면 "" 만 반환되므로 `str`이 "" 와 같은지 비교해서 "" 가 아닐(!) 경우 14행을 반복하게 됨.



Self study12-3

1. 파일 data2-1.txt 를 만들어 소수점이 들어간 실수 6줄을 쓰고 저장한다. 이 실수 6줄의 합계가 나오도록 [실습10-12]를 수정해보자.

Hint : System.out.printf()에서 실수는 “%7.2f”와 같은 서식을 사용한다.

2. [실습10-14]의 `fw.write(str + "\r\n");` 에서 “\r\n”을 붙이지 않아도 행이 넘어가지 않도록 코드를 수정해보자.

Hint : FileWriter 대신 Printwriter 를 사용하고 Write() 대신 println()을 사용한다.

질문은 이메일을 이용해주세요.
ds.june2@gmail.com

감사합니다