



# 13장. 입출력

한림대학교 소프트웨어융합대학 양은샘.

	<p><b>이것이 자바다</b> 신용권의 Java 프로그래밍 정복</p> <p>저자 신용권 출판 한빛미디어   2015.1.5 페이지수 1,224   사이즈 183*235mm 판매가 서적 27,000원 구매이벤트 IT독자 설문이벤트 외 6건</p>		<p><b>혼자 공부하는 자바</b> JAVA 8 &amp; 11 지원/무료 동영상 강의 제공</p> <p>저자 신용권 출판 한빛미디어   2019.6.10. 페이지수 708   사이즈 188*257mm 판매가 서적 21,600원</p>
--	--	--	--

# 13장. 입출력

---

- ❖ 안녕하세요? 여러분!
- ❖ 오늘은 자바의 입출력 단원을 학습합니다.
- ❖ 프로그램은 데이터를 읽고 출력하는 작업을 빈번히 수행합니다.
  - 프로그램에서의 효과적인 입출력을 위해
  - 자바 표준 API에서는 스트림을 이용한 다양한 입출력 기능을 제공합니다.
  - 콘솔 또는 파일을 이용하여 데이터를 읽거나 출력하는 여러가지 방법 대해 알아보도록 하겠습니다.
- ❖ 지난 시간에 학습한 내용을 리뷰한 후 학습을 시작하도록 하겠습니다.

# 지난 시간 Review

---

1절. 제네릭(Generic)

2절. 컬렉션(Collection)

3절. List 컬렉션

4절. Set 컬렉션

5절. Map 컬렉션

6절. 검색 기능을 강화시킨 컬렉션

7절. LIFO와 FIFO 컬렉션

8절. 동기화된(synchronized) 컬렉션

9절. 병렬 처리를 위한 컬렉션

# 학습 목차

---

1절. java.io

2절. 콘솔(Console) 입출력

3절. File 클래스

4절. 파일(File) 입출력

5절. 바이트 출력 스트림

6절. 바이트 입력 스트림

7절. 바이트 단위 보조 스트림

8절. 문자 출력 스트림

9절. 문자 입력 스트림

10절. 문자 단위 보조 스트림

11절. 프린터 보조 스트림

12절. 네트워크 -> 컴퓨터 네트워크 교과목

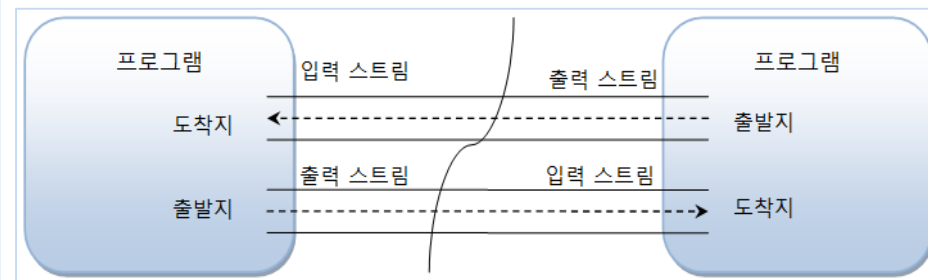
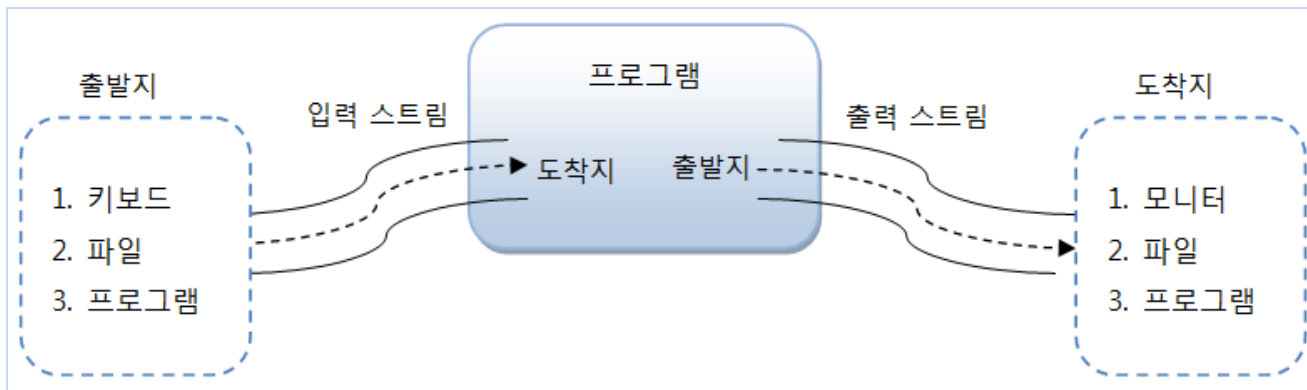
# 학습 목표

---

- ❖ 자바 입출력에 사용되는 스트림(Stream)의 개념을 안다.
- ❖ java.io의 서비스 범위와 사용법을 알고 적용할 수 있다.
- ❖ File을 생성하고 바이트 또는 문자 입출력 스트림을 사용해 요구사항에 적합한 입출력을 제시하고 구현할 수 있다.
- ❖ 보조스트림의 다양한 기능을 이용하여 효과적인 입출력을 구현할 수 있다.
- ❖ 문제 상황에서 입출력의 필요성 여부를 판단할 수 있으며, 요구사항에 가장 적합한 입출력 시스템을 설계 및 구현하여 해당 문제를 해결할 수 있다.

# 1절. java.io

## ❖ 스트림(Stream)



## ❖ 입출력 스트림의 종류

### ■ 바이트 기반 스트림

- 그림, 멀티미디어 등의 바이너리 데이터를 읽거나 출력할 때 사용

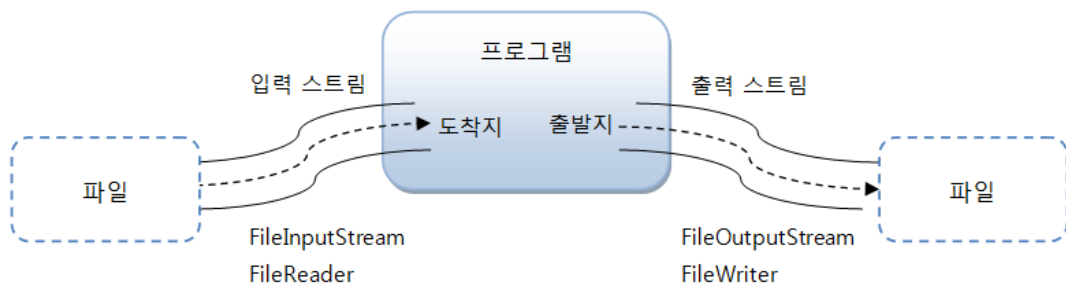
### ■ 문자 기반 스트림

- 문자 데이터를 읽거나 출력할 때 사용

# java.io 패키지

## ❖ 최상위 클래스로

### ■ 바이트 또는 문자 기반 스트림 판단



java.io 패키지의 주요 클래스	설명
File	파일 시스템의 파일의 정보를 얻기위한 클래스
Console	콘솔로부터 문자를 입출력하기 위한 클래스
InputStream / OutputStream	바이트 단위 입출력을 위한 최상위 입출력 스트림 클래스
FileInputStream / FileOutputStream DataInputStream / DataOutputStream ObjectInputStream / ObjectOutputStream PrintStream BufferedInputStream / BufferedOutputStream	바이트 단위 입출력을 위한 하위 스트림 클래스
Reader / Writer	문자 단위 입출력을 위한 최상위 입출력 스트림 클래스
FileReader / FileWriter InputStreamReader / OutputStreamWriter PrintWriter BufferedReader / BufferedWriter	문자 단위 입출력을 위한 하위 스트림 클래스

구분	바이트 기반 스트림		문자 기반 스트림	
	입력 스트림	출력 스트림	입력 스트림	출력 스트림
최상위 클래스	InputStream	OutputStream	Reader	Writer
하위 클래스 (예)	XXXputStream (FileInputStream)	XXXOutputStream (FileOutputStream)	XXXReader (FileReader)	XXXWriter (FileWriter)

## 2절. 콘솔(Console) 입출력

### ❖ 콘솔 (Console)

- 키보드로 입력 받고 모니터로 출력하는 소프트웨어
- 콘솔 입력 : System.in 사용
- 콘솔 출력 : System.out 사용
- 에러 출력 : System.err 사용
- 예)
  - Unix, Linux : 터미널
  - Windows 운영체제 : 명령 프롬프트
  - 이클립스 : Console 뷰



### ❖ Console 클래스

- 자바6부터 콘솔에서 입력된 문자열을 쉽게 읽을 수 있도록 제공
- 이클립스에서 System.console()은 null 리턴
- 명령 프롬프트에서 반드시 실행
- 읽기 메소드 ->

리턴타입	메소드	설명
String	readLine()	엔터키를 입력하기 전의 모든 문자열을 읽음
char[]	readPassword()	키보드 입력 문자를 콘솔에 보여주지 않고 문자열을 읽음



# System.in

## ❖ System.in

- 자바는 콘솔에서 키보드 데이터를 입력 받을 수 있도록 System 클래스의 in 정적 필드 제공.
- InputStream 타입의 입력 스트림 : InputStream 변수 대입 가능

```
InputStream is = System.in;
```

## ❖ System.out

- 자바는 콘솔에서 모니터로 데이터를 출력하기 위해 System 클래스의 out 정적 필드 제공.
- PrintStream 타입의 출력 스트림
  - PrintStream이 제공하는 print(), println(), printf() 등 메소드 사용
- OutputStream으로 타입 변환 가능

# Scanner 클래스

❖ Console 클래스는 문자열은 읽을 수 있지만, 기본 타입(정수, 실수) 값을 바로 읽을 수 없음.

❖ java.util.Scanner

- 콘솔로부터 기본 타입의 값을 바로 읽을 수 있음

❖ 메소드

리턴타입	메소드	설명
boolean	nextBoolean()	boolean(true/false) 값을 읽는다.
byte	nextByte()	byte 값을 읽는다.
short	nextShort()	short 값을 읽는다.
int	nextInt()	int 값을 읽는다.
long	nextLong()	long 값을 읽는다.
float	nextFloat()	float 값을 읽는다.
double	nextDouble()	double 값을 읽는다.
String	nextLine()	String 값을 읽는다.

생성된 Scanner를 변수에 저장

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

scanner 변수 선언    바이트 기반 입력 스트림으로부터 scanner 생성

읽은 문자열을 String 변수에 저장

```
String inputData = scanner.nextLine();
```

String 변수 선언    [Enter] 이전까지 입력된 행단위 문자열을 읽음

# 3절. File 클래스

## ❖ File 클래스 / java.io 패키지

### ■ 파일 및 폴더 관련 정보를 제공

- 파일 크기, 속성, 이름, 생성 정보, 삭제, 디렉토리 생성, 디렉토리에 존재하는 파일의 리스트 등의 정보 제공

## ❖ 파일 객체 생성

- 경로 구분자는 운영체제마다 조금씩 다름, 윈도우에서는 '₩' 또는 '/' 모두 사용 가능

```
File file = new File("C:₩₩Temp₩₩file.txt");  
File file = new File("C:/Temp/file.txt");
```

## ❖ 파일 및 디렉토리 생성/삭제 관련 메소드

리턴 타입	메소드	설명
boolean	createNewFile()	새로운 파일을 생성합니다.
boolean	mkdir()	새로운 폴더를 생성합니다.
boolean	mkdirs()	경로상에 없는 모든 폴더를 생성합니다.

# File 클래스의 메소드

## ❖ 파일 또는 디렉토리 존재 유무 확인 메소드

```
boolean isExist = file.exists();
```

## ❖ exists()의 리턴 값이 true이면

- 다음 메소드 사용 가능 ->

리턴 타입	메소드	설명
boolean	delete()	파일 또는 폴더를 삭제합니다.
boolean	canExecute()	실행할 수 있는 파일인지 여부를 확인합니다.
boolean	canRead()	읽을 수 있는 파일인지 여부를 확인합니다.
boolean	canWrite()	수정 및 저장할 수 있는 파일인지 여부를 확인합니다.
String	getName()	파일의 이름을 리턴합니다.
String	getParent()	부모 폴더를 리턴합니다.
File	getParentFile()	부모 폴더를 File 객체로 생성 후 리턴합니다.
String	getPath()	전체 경로를 리턴합니다.
boolean	isDirectory()	폴더인지 여부를 확인합니다.
boolean	isFile()	파일인지 여부를 확인합니다.
boolean	isHidden()	숨김 파일인지 여부를 확인합니다.
long	lastModified()	마지막 수정 날짜 및 시간을 리턴합니다.
long	length()	파일의 크기를 리턴합니다.
String[]	list()	폴더에 포함된 파일 및 서브 폴더 목록 전부를 String 배열로 리턴합니다.
String[]	list(FilenameFilter filter)	폴더에 포함된 파일 및 서브 폴더 목록 중에 FilenameFilter에 맞는 것만 String 배열로 리턴합니다.
File[]	listFiles()	폴더에 포함된 파일 및 서브 폴더 목록 전부를 File 배열로 리턴합니다.
File[]	listFiles(FilenameFilter filter)	폴더에 포함된 파일 및 서브 폴더 목록 중에 FilenameFilter에 맞는 것만 File 배열로 리턴합니다.



C:/Temp 폴더의 내용 목록을  
File 배열로 얻음

파일 또는 폴더 정보를 출력

55 실행결과

# 4절. 파일(File) 입출력

## ❖ 바이트 단위 파일 입출력

### ■ FileInputStream / FileOutputStream

- InputStream 하위 클래스, OutputStream 하위 클래스
- 바이트 단위로 파일에서 데이터를 읽어 들일 때 / 파일로 데이터를 저장할 때 사용.
- 그림, 오디오, 비디오, 텍스트 파일 등 모든 종류의 데이터를 파일 읽기 또는 저장 가능.

InputStream / OutputStream	바이트 단위 클래스
FileInputStream / FileOutputStream DataInputStream / DataOutputStream ObjectInputStream / ObjectOutputStream PrintStream BufferedInputStream / BufferedOutputStream	바이트 단위
Reader / Writer	문자 단위 클래스
FileReader / FileWriter InputStreamReader / OutputStreamWriter PrintWriter BufferedReader / BufferedWriter	문자 단위

## ❖ 문자 단위 파일 입출력

### ■ FileReader / FileWriter

- Reader 하위 클래스 / Writer 하위 클래스
- 텍스트 파일로부터 문자 데이터를 읽어 들일 때 / 텍스트 파일에 문자 데이터를 저장할 때 사용.
- 텍스트가 아닌 그림, 오디오, 비디오 등의 파일은 읽거나 쓸 수 없음.

# FileInputStream / FileOutputStream

## ❖ FileInputStream 객체 생성 방법

- FileInputStream(파일명)
- FileInputStream(File객체)

- 객체가 생성될 때 파일과 직접 연결 됨.
- 파일이 존재하지 않으면 FileNotFoundException 발생, try-catch문으로 예외 처리.

//첫 번째 방법

```
FileInputStream fis = new FileInputStream("C:/Temp/image.gif");
```

//두 번째 방법

```
File file = new File("C:/Temp/image.gif");  
FileInputStream fis = new FileInputStream(file);
```

## ❖ FileOutputStream 객체 생성 방법

- FileOutputStream(파일명, 옵션)
- FileOutputStream(File객체, 옵션)

```
FileOutputStream fis = new FileOutputStream("C:/Temp/data.txt", true);  
FileOutputStream fis = new FileOutputStream(file, true);
```

- 파일이 이미 존재할 경우, 데이터를 출력하게 되면 파일을 덮어씀.
- 기존 파일 내용 끝에 데이터를 추가할 경우에는 두 번째 매개변수에 true사용.

# FileReader / FileWriter

## ❖ FileReader 객체 생성 방법

- FileReader(파일명)
- FileReader(File객체)

- 객체가 생성될 때 파일과 직접 연결 됨.
- 만약 파일이 존재하지 않으면 FileNotFoundException 발생, try-catch문으로 예외 처리.

//방법 1

```
FileReader fr = new FileReader("C:/Temp/file.txt");
```

//방법 2

```
File file = new File("C:/Temp/file.txt");
```

```
FileReader fr = new FileReader(file);
```

## ❖ FileWriter 객체 생성 방법

- FileWriter(파일명, 옵션)
- FileWriter(File객체, 옵션)

```
FileWriter fw = new FileWriter("C:/Temp/file.txt", true);
```

```
FileWriter fw = new FileWriter(file, true);
```

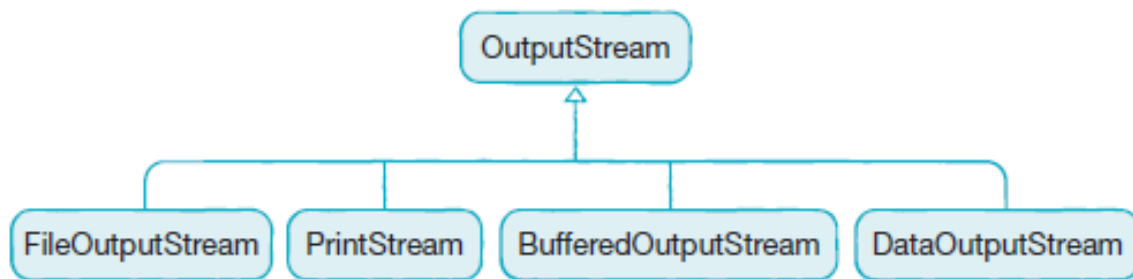
- 파일이 이미 존재할 경우, 데이터를 출력하게 되면 파일을 덮어쓰게 됨.
- 기존 파일 내용 끝에 데이터를 추가할 경우에는 두 번째 매개변수에 true사용.



# 5절. 바이트 출력 스트림

## ❖ OutputStream

- 바이트 기반 출력 스트림의 최상위 클래스
- 모든 바이트 기반 출력 스트림은 OutputStream 클래스를 상속받아 만들어짐.

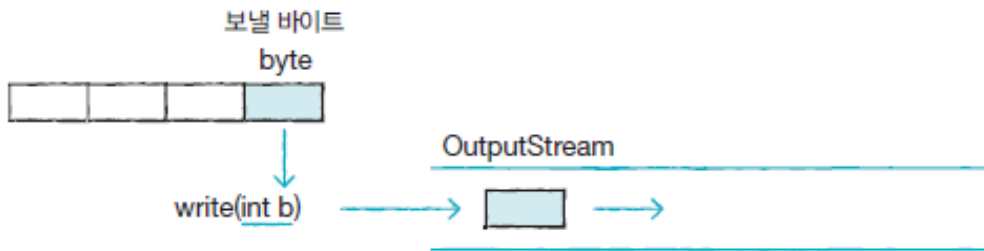


리턴 타입	메소드	설명
void	write(int b)	1byte를 출력합니다.
void	write(byte[] b)	매개값으로 주어진 배열 b의 모든 바이트를 출력합니다.
void	write(byte[] b, int off, int len)	매개값으로 주어진 배열 b[off]부터 len개까지의 바이트를 출력합니다.
void	flush()	출력 버퍼에 잔류하는 모든 바이트를 출력합니다.
void	close()	출력 스트림을 닫습니다.

# OutputStream : write(int b)

## ❖ write(int b)

- 매개 변수로 주어지는 int(4byte)에서 끝 1byte만 출력 스트림으로 보냄



- 예) 1byte씩 출력하기 ->

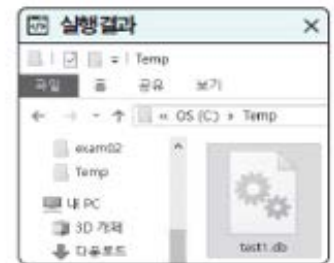
```
03 import java.io.FileOutputStream;
04 import java.io.OutputStream;
05
06 public class WriteExample {
07     public static void main(String[] args) throws Exception {
08         OutputStream os = new FileOutputStream("C:/Temp/test1.db");
09
10         byte a = 10;
11         byte b = 20;
12         byte c = 30;
13
14         os.write(a);
15         os.write(b);
16         os.write(c);
17
18         os.flush();
19         os.close();
20     }
21 }
```

데이터 도착지를 test1.db로 하는  
바이트 기반 파일 출력 스트림을 생성

1byte씩 출력

출력 버퍼에 잔류하는 모든 바이트를 출력

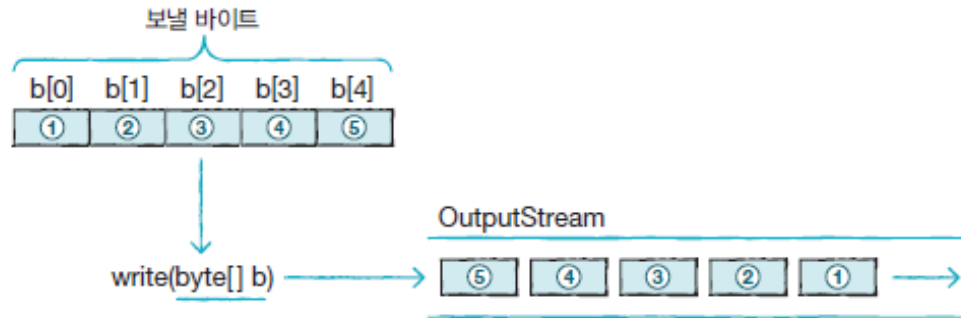
출력 스트림을 닫음



# OutputStream : write(byte[] b)

## ❖ write(byte[] b)

- 매개값으로 주어진 배열의 모든 바이트를 출력 스트림으로 보냄



- 예) 배열 전체를 출력하기 ->

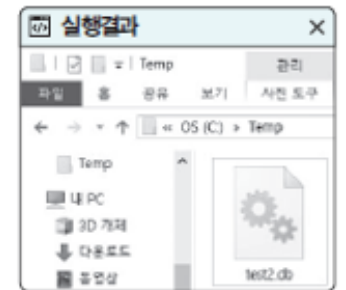
```
03 import java.io.FileOutputStream;
04 import java.io.OutputStream;
05
06 public class WriteExample {
07     public static void main(String[] args) throws Exception {
08         OutputStream os = new FileOutputStream("C:/Temp/test2.db");
09
10         byte[] array = { 10, 20, 30 };
11
12         os.write(array);
13
14         os.flush();
15         os.close();
16     }
17 }
```

데이터 도착지를 test2.db로 하는  
바이트 기반 파일 출력 스트림을 생성

배열의 모든 바이트를 출력

출력 버퍼에 잔류하는 모든 바이트를 출력

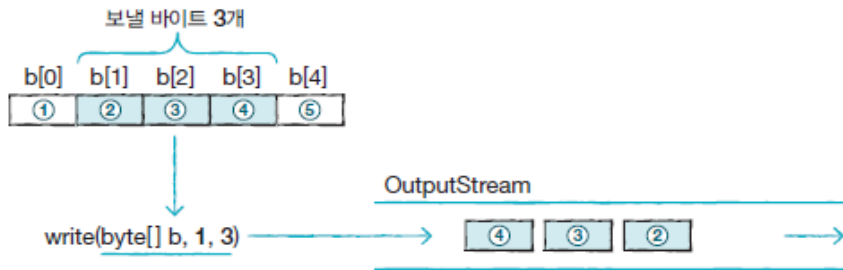
출력 스트림을 닫음



# OutputStream : write(byte[] b, int off, int len)

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

- 매개값으로 주어진 배열의 일부 바이트를 출력 스트림으로 보냄



```
03 import java.io.FileOutputStream;
04 import java.io.OutputStream;
05
06 public class WriteExample {
07     public static void main(String[] args) throws Exception {
08         OutputStream os = new FileOutputStream("C:/Temp/test3.db");
09
10         byte[] array = { 10, 20, 30, 40, 50 };
11
12         os.write(array, 1, 3);
13
14         os.flush();
15         os.close();
16     }
17 }
```

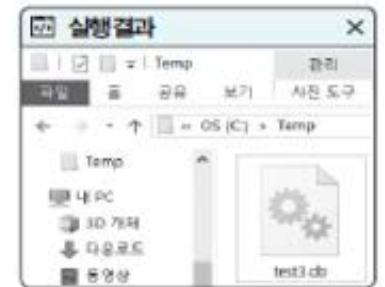
데이터 도착지를 test3.db로 하는  
바이트 기반 파일 출력 스트림을 생성

배열의 1번 인덱스부터  
3개를 출력

출력 버퍼에 잔류하는 모든 바이트를 출력

출력 스트림을 닫음

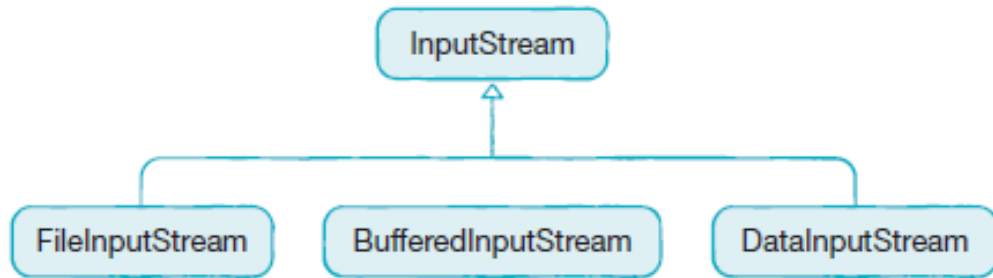
- 예) 배열 일부를 출력하기 ->



# 6절. 바이트 입력 스트림

## ❖ InputStream

- 바이트 기반 입력 스트림의 최상위 클래스
- 모든 바이트 기반 입력 스트림은 InputStream 클래스를 상속받아 만들어짐.

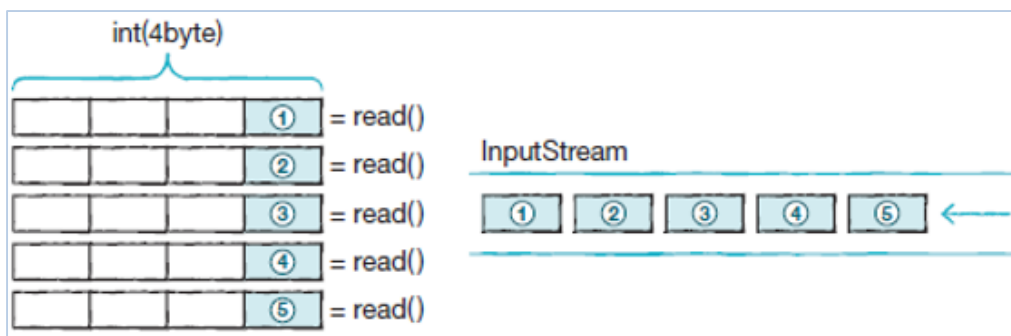


리턴 타입	메소드	설명
int	read()	1byte를 읽고 읽은 바이트를 리턴합니다.
int	read(byte[] b)	읽은 바이트를 매개값으로 주어진 배열에 저장하고 읽은 바이트 수를 리턴합니다.
int	read(byte[] b, int off, int len)	len개의 바이트를 읽고 매개값으로 주어진 배열에서 b[off]부터 len개 까지 저장합니다. 그리고 읽은 바이트 수를 리턴합니다.
void	close()	입력 스트림을 닫습니다.

# InputStream : read()

## ❖ read()

- 입력 스트림으로부터 1byte를 읽어, int(4byte) 타입으로 리턴.
- 리턴 된 4byte 중 끝 1byte에만 데이터가 들어 있음.
- 더 이상 바이트를 읽을 수 없게 되면 -1 리턴.



- 예) 1byte씩 읽기 ->

```
03 import java.io.FileInputStream;
04 import java.io.InputStream;
05
06 public class ReadExample {
07     public static void main(String[] args) throws Exception {
08         InputStream is = new FileInputStream("C:/Temp/test1.db");
09         while(true) {
10             int data = is.read();
11             if(data == -1) break;
12             System.out.println(data);
13         }
14         is.close();
15     }
16 }
```

데이터출발지를 test1.db로 하는  
바이트 기반 파일 입력 스트림을 생성

1byte씩 읽기

파일 끝에 도달했을 경우

입력 스트림을 닫음

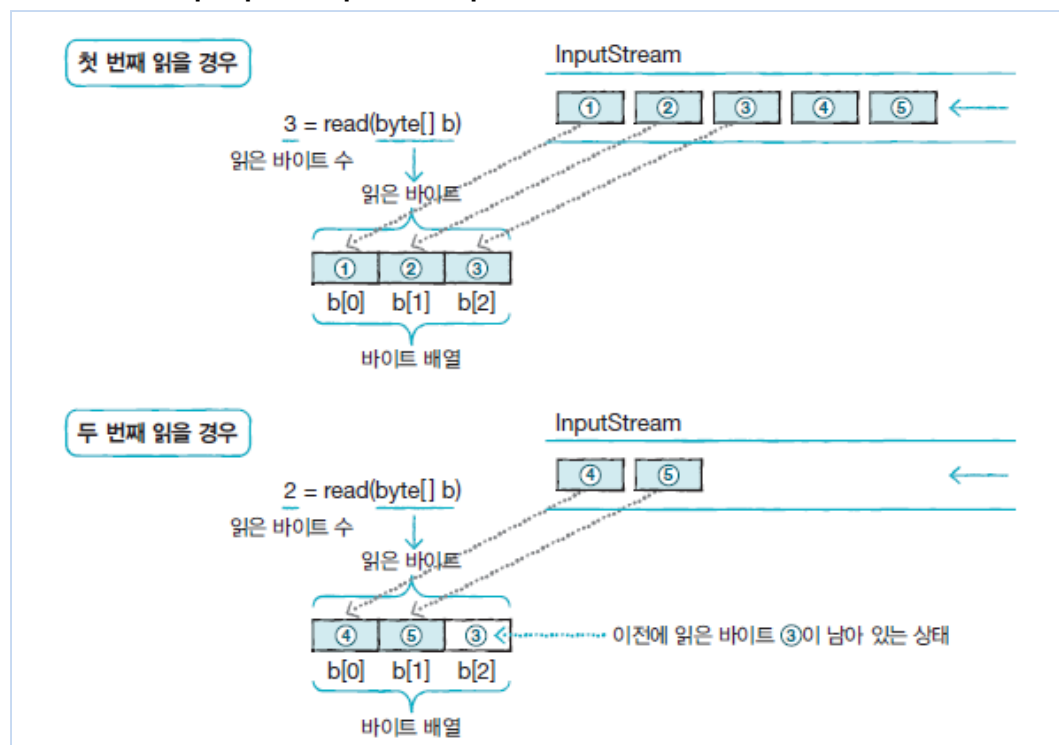
실행결과

10
20
30

# InputStream : read(byte[] b)

## ❖ read(byte[] b)

- 입력 스트림으로부터 매개값으로 주어진 배열의 길이만큼 바이트를 읽어, 해당 배열에 저장.
- 읽은 바이트 수를 리턴



```
03 import java.io.FileInputStream;
04 import java.io.InputStream;
05
06 public class ReadExample {
07     public static void main(String[] args) throws Exception {
08         InputStream is = new FileInputStream("C:/Temp/test2.db");
09
10         byte[] buffer = new byte[100];
11
12         while(true) {
13             int readByteNum = is.read(buffer);
14             if(readByteNum == -1) break;
15             for(int i=0; i<readByteNum; i++) {
16                 System.out.println(buffer[i]);
17             }
18         }
19
20         is.close();
21     }
22 }
```

데이터 출발지를 test2.db로 하는 바이트 기반 파일 입력 스트림을 생성

길이 100인 배열 생성

배열 길이만큼 읽기

파일 끝에 도달했을 경우

읽은 바이트 수만큼 반복하면서 배열에 저장된 바이트를 출력

입력 스트림을 닫음

실행결과

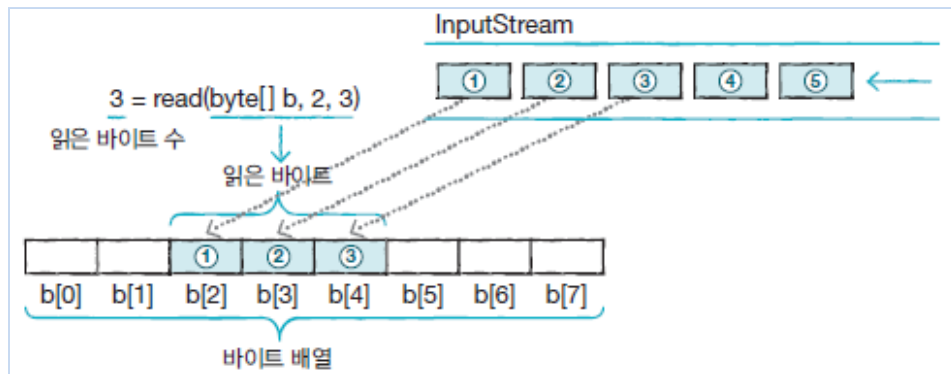
```
10
20
30
```

- 예) 배열의 길이 만큼 읽기 ->

# InputStream : read(byte[] b, int off, int len)

## ❖ read(byte[] b, int off, int len)

- 입력 스트림으로부터 len개의 바이트만큼 읽고,
- 매개값으로 주어진 바이트 배열 b[off]부터 len개 까지 저장.
- 읽은 바이트 수인 len개 리턴.



- 예) 지정한 길이 만큼 읽기 ->

```
03 import java.io.FileInputStream;
04 import java.io.InputStream;
05
06 public class ReadExample {
07     public static void main(String[] args) throws Exception {
08         InputStream is = new FileInputStream("C:/Temp/test3.db");
09
10         byte[] buffer = new byte[5];
11
12         int readByteNum = is.read(buffer, 2, 3);
13         if(readByteNum != -1) {
14             for(int i=0; i<buffer.length; i++) {
15                 System.out.println(buffer[i]);
16             }
17         }
18
19         is.close();
20     }
21 }
```

데이터 출발지를 test3.db로 하는 바이트 기반 파일 입력 스트림을 생성

입력 스트림으로부터 3byte를 읽고 buffer[2] buffer[3], buffer[4]에 각각 저장

읽은 바이트가 있다면

배열 전체를 읽고 출력

입력 스트림을 닫음

실행결과

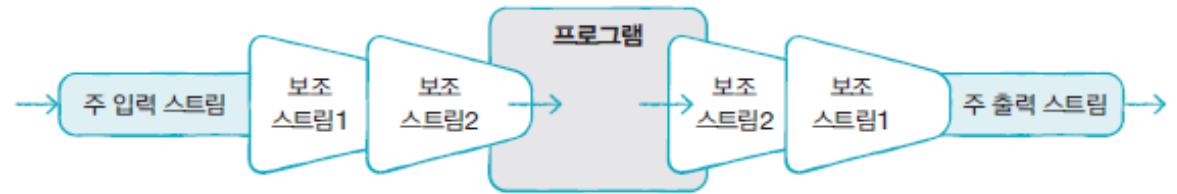
```
0
0
20
30
40
```



# 7절. 바이트 단위 보조 스트림

## ❖ 보조 스트림

- 입출력 스트림과 연결되어 여러가지 편리한 기능을 제공.
- 자체적으로는 입출력 수행할 수 없음.
- InputStream, OutputStream, Reader, Writer 등에 연결하여 입출력 수행.
- 보조 스트림을 연속적으로 연결할 수도 있음.



## ❖ 보조 스트림 연결 방법

- 보조 스트림 생성 시 자신이 연결될 스트림을 생성자의 매개 값으로 제공.

```
보조스트림 변수 = new 보조스트림(연결스트림)
```

```
BufferedInputStream bis = new BufferedInputStream(바이트 기반 입력 스트림);
```

# 바이트 단위 성능 향상 보조 스트림

## ❖ 입출력 성능에 영향을 미치는 요소

- 입출력 소스, 하드 디스크, 느린 네트워크

## ❖ BufferedInputStream / BufferedOutputStream

- 바이트 기반 스트림에서 사용
- 입출력 소스와 직접 작업하지 않고 버퍼(buffer)를 이용해 해결
  - 프로그램에서 전송한 데이터를 내부 버퍼에 쌓아두었다가 버퍼가 꽉 차면 한꺼번에 처리
  - 읽기 / 쓰기 속도가 향상 됨.

```
BufferedInputStream bis = new BufferedInputStream(바이트 기반 입력 스트림);
```



# BufferedInputStream/BufferedOutputStream 예

## ❖ 파일 복사 성능 비교

```
03 import java.io.*;
04
05 public class NonBufferVsBufferExample {
06     public static void main(String[] args) throws Exception {
07         String originalFilePath1 =
08             NonBufferVsBufferExample.class.getResource("originalFile1.jpg").getPath();
09         String targetFilePath1 = "C:/Temp/targetFile1.jpg";
10         FileInputStream fis = new FileInputStream(originalFilePath1);
11         FileOutputStream fos = new FileOutputStream(targetFilePath1);
12
13         String originalFilePath2 =
14             NonBufferVsBufferExample.class.getResource("originalFile2.jpg").getPath();
15         String targetFilePath2 = "C:/Temp/targetFile2.jpg";
16         FileInputStream fis2 = new FileInputStream(originalFilePath2);
17         FileOutputStream fos2 = new FileOutputStream(targetFilePath2);
18         BufferedInputStream bis = new BufferedInputStream(fis2);
19         BufferedOutputStream bos = new BufferedOutputStream(fos2);
20
21         long nonBufferTime = copy(fis, fos);
22         System.out.println("버퍼를 사용하지 않았을 때:\t" + nonBufferTime + "ns");
23
24         long bufferTime = copy(bis, bos);
25         System.out.println("버퍼를 사용했을 때:\t\t" + bufferTime + "ns");
26
27         fis.close();
28         fos.close();
29         bis.close();
30         bos.close();
31     }
32
33     static int data = -1;
34     public static long copy(InputStream is, OutputStream os) throws Exception {
35         long start = System.nanoTime();
36         while(true) {
37             data = is.read();
38             if(data == -1) break;
39             os.write(data);
40         }
41         os.flush();
42         long end = System.nanoTime();
43         return (end-start);
44     }
45 }
```

기본 스트림 생성

버퍼 보조 스트림 연결

FileInputStream, FileOutputStream을 이용한 복사 시간 측정

BufferedInputStream, BufferedOutputStream을 이용한 복사 시간 측정

시작 시간 저장

[파일 복사] 원본 파일에서 읽은 byte를 타겟 파일로 바로 출력

끝 시간 저장

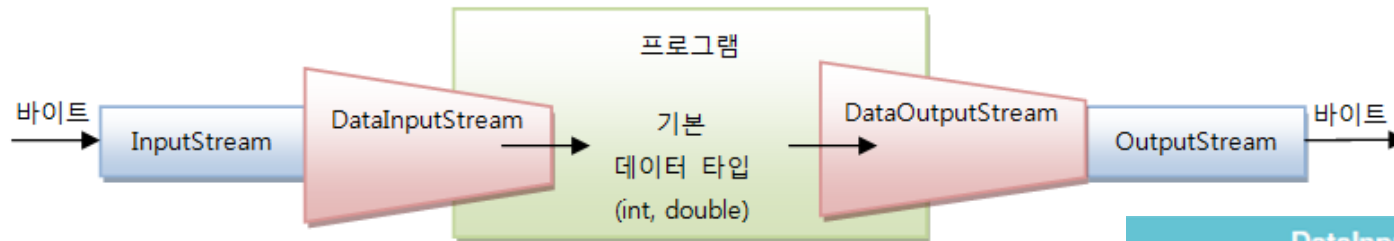
복사에 걸린 시간 리턴

실행결과	
버퍼를 사용하지 않았을 때:	5330067700ns
버퍼를 사용했을 때:	42501200ns

# 기본 타입 입출력 보조 스트림

## ❖ DataInputStream / DataOutputStream

- 기본 타입인 boolean, char, short, int, long, float, double 데이터를 입출력 할 수 있음.
- 데이터 타입의 크기가 모두 다르므로 DataOutputStream으로 출력한 데이터를 DataInputStream으로 읽어들 때는 출력한 순서와 동일한 순서로 읽어야 함.



```
DataInputStream dis = new DataInputStream(바이트 기반 입력 스트림);  
DataOutputStream dos = new DataOutputStream(바이트 기반 출력 스트림);
```

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

# DataInputStream / DataOutputStream 예

```
03 import java.io.*;
04
05 public class DataInputStreamOutputStreamExample {
06     public static void main(String[] args) throws Exception {
07         FileOutputStream fos = new FileOutputStream("C:/Temp/primitive.db");
08         DataOutputStream dos = new DataOutputStream(fos);
09
10         dos.writeUTF("홍길동");
11         dos.writeDouble(95.5);
12         dos.writeInt(1);
13
14         dos.writeUTF("감자바");
15         dos.writeDouble(90.3);
16         dos.writeInt(2);
```

← 기본 타입 값 출력

← 바이트 기반 출력 스트림을 생성하고 DataOutputStream 보조 스트림 연결

```
17
18     dos.flush(); dos.close(); ← 출력 스트림 닫기
19
20     FileInputStream fis = new FileInputStream("C:/Temp/primitive.db");
21     DataInputStream dis = new DataInputStream(fis);
22
23     for(int i=0; i<2; i++) {
24         String name = dis.readUTF();
25         double score = dis.readDouble(); ← 기본 타입 값 읽기
26         int order = dis.readInt();
27         System.out.println(name + " : " + score + " : " + order);
28     }
29
30     dis.close(); ← 입력 스트림 닫기
31 }
32 }
```

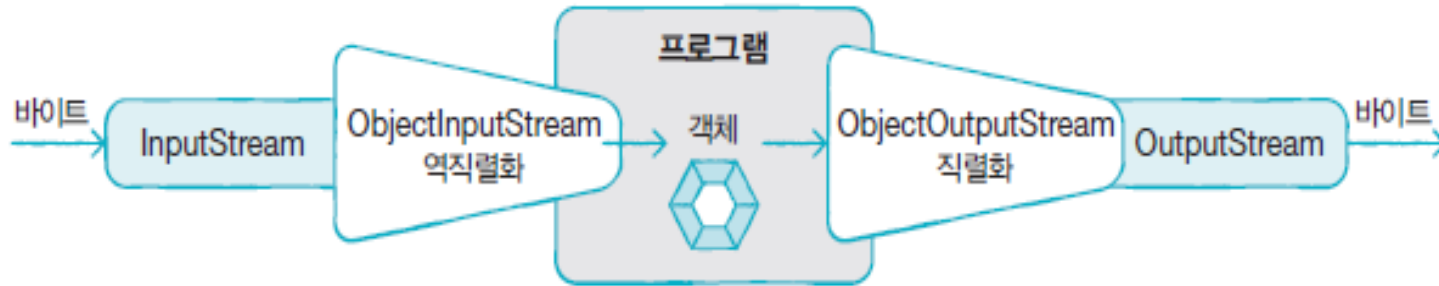
← 바이트 기반 입력 스트림을 생성하고 DataInputStream 보조 스트림 연결

실행결과	
홍길동	: 95.5 : 1
감자바	: 90.3 : 2

# 객체 입출력 보조 스트림

## ❖ ObjectInputStream / ObjectOutputStream

- 객체를 파일 또는 네트워크로 입출력 할 수 있는 기능 제공 : 객체 직렬화
- 객체는 문자가 아니므로 바이트 기반 스트림으로 데이터 변경 필요



```
ObjectInputStream ois = new ObjectInputStream(바이트 기반 입력 스트림);  
ObjectOutputStream oos = new ObjectOutputStream(바이트 기반 출력 스트림);
```

# java.io.Serializable

- ❖ 자바는 java.io.Serializable 인터페이스를 구현한 객체만 직렬화 할 수 있도록 제한
  - transient 필드는 제외
  - Serializable 인터페이스는 메소드 선언이 없으므로, 네트워크로 전송하려는 경우 클래스 선언 시 implements Serializable을 추가해야 함.

```
public class XXX implements Serializable { ... }
```

- 객체를 직렬화 할 때 private 필드를 포함한 모든 필드를 바이트로 변환 가능.
  - 직렬화된 객체를 역직렬화 할 때는 직렬화 했을 때와 같은 클래스 사용해야 함.
  - 클래스의 이름이 같더라도 클래스의 내용이 변경된 경우는 역직렬화 실패.
- 
- ❖ serialVersionUID 필드
    - 같은 클래스임을 알려주는 식별자 역할
    - Serializable 인터페이스를 구현하면 컴파일 시 자동적으로 serialVersionUID 정적 필드가 추가 됨.
      - 재컴파일하면 serialVersionUID의 값 변경
    - 불가피한 수정 있을 경우 명시적으로 serialVersionUID 선언 `static final long serialVersionUID = 정수값;`



# writeObject() / readObject()

## ❖ writeObject(ObjectOutputStream out)

- 직렬화 직전 자동 호출, 추가적으로 직렬화할 내용 작성 가능

```
oos.writeObject(객체);
```

## ❖ readObject(ObjectInputStream in)

- 역직렬화 직전 자동 호출, 추가적으로 역직렬화 내용 작성 가능
- 원래 타입으로 강제 변환 필요

```
객체타입 변수 = (객체타입) ois.readObject();
```

## ❖ 추가 직렬화 및 역직렬화가 필요한 경우

- 부모 클래스가 Serializable을 구현하지 않고, 자식 클래스가 Serializable을 구현한 경우
  - 부모 필드는 직렬화에서 제외
  - writeObject()에서 부모 필드 직렬화 필요 / readObject()에서 부모 필드 역직렬화 필요
- 부모 클래스가 Serializable 구현하도록 하는 것이 제일 쉬움



# 객체 입출력 보조 스트림 예 (1)

```
03 import java.io.*;
04 import java.text.SimpleDateFormat;
05 import java.util.*;
06
07 public class ObjectOutputStreamExample {
08     public static void main(String[] args) throws Exception {
09         writeList(); ← List를 파일에 저장
10         List<Board> list = readList(); ← 파일에 저장된 List 읽기
11
12         SimpleDateFormat sdf = new SimpleDateFormat("yyyy-MM-dd");
13         for(Board board : list) {
14             System.out.println(
15                 board.getBno() + "\t" + board.getTitle() + "\t" +
16                 board.getContent() + "\t" + board.getWriter() + "\t" +
17                 sdf.format(board.getDate()) ← List 내용을 모니터에 출력
18             );
19         }
20     }
21
22     public static void writeList() throws Exception {
23         List<Board> list = new ArrayList<>(); ← List 생성
```

```
24
25         list.add(new Board(1, "제목1", "내용1", "글쓴이1", new Date()));
26         list.add(new Board(2, "제목2", "내용2", "글쓴이2", new Date())); ← list에 Board 객체 저장
27         list.add(new Board(3, "제목3", "내용3", "글쓴이3", new Date()));
28
29         FileOutputStream fos = new FileOutputStream("C:/Temp/board.db");
30         ObjectOutputStream oos = new ObjectOutputStream(fos);
31         oos.writeObject(list);
32         oos.flush();
33         oos.close();
34     }
35
36     public static List<Board> readList() throws Exception {
37         FileInputStream fis = new FileInputStream("C:/Temp/board.db");
38         ObjectInputStream ois = new ObjectInputStream(fis);
39         List<Board> list = (List<Board>) ois.readObject(); ← 객체 입력 스트림을 이용해서 list 읽기
40         return list;
41     }
42 }
```

실행결과				
1	제목1	내용1	글쓴이1	2019-05-07
2	제목2	내용2	글쓴이2	2019-05-07
3	제목3	내용3	글쓴이3	2019-05-07

# 객체 입출력 보조 스트림 예 (2)

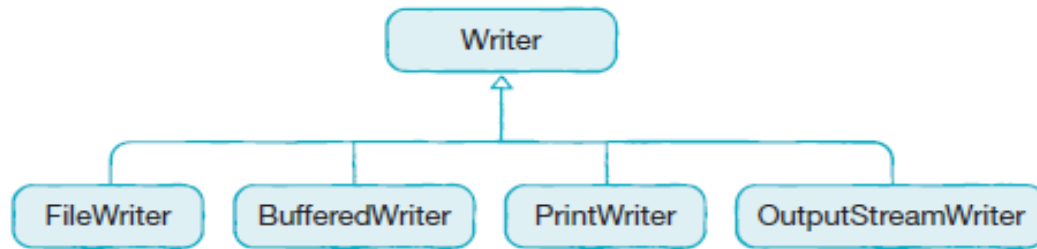
```
03 import java.io.Serializable;
04 import java.util.Date;
05
06 public class Board implements Serializable {
07     private int bno;
08     private String title;
09     private String content;
10     private String writer;
11     private Date date;
12
13     public Board(int bno, String title, String content, String writer, Date date) {
14         this.bno = bno;
15         this.title = title;
16         this.content = content;
17         this.writer = writer;
18         this.date = date;
19     }
20
21     public int getBno() { return bno; }
22     public void setBno(int bno) { this.bno = bno; }
23     public String getTitle() { return title; }
24     public void setTitle(String title) { this.title = title; }
25     public String getContent() { return content; }
26     public void setContent(String content) { this.content = content; }
27     public String getWriter() { return writer; }
28     public void setWriter(String writer) { this.writer = writer; }
29     public Date getDate() { return date; }
30     public void setDate(Date date) { this.date = date; }
31 }
```

Serializable 인터페이스 구현

# 8절. 문자 출력 스트림

## ❖ Writer

- 문자 기반 출력 스트림의 최상위 클래스.
- 모든 문자 기반 출력 스트림은 Writer 클래스를 상속받아 만들어짐.

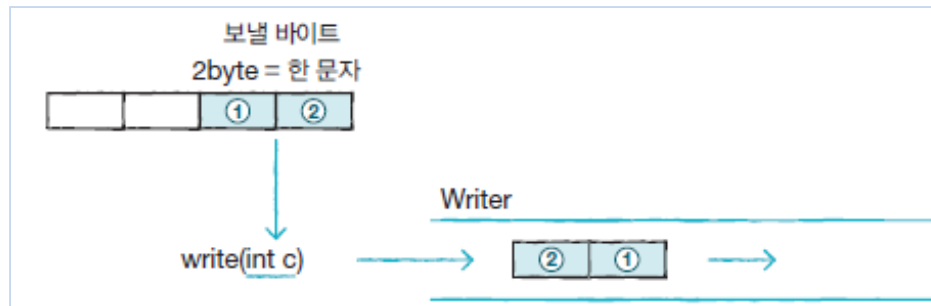


리턴 타입	메소드	설명
void	write(int c)	매개값으로 주어진 한 문자를 보냅니다.
void	write(char[] cbuf)	매개값으로 주어진 배열의 모든 문자를 보냅니다.
void	write(char[] cbuf, int off, int len)	매개값으로 주어진 배열에서 cbuf[off]부터 len개까지의 문자를 보냅니다.
void	write(String str)	매개값으로 주어진 문자열을 보냅니다.
void	write(String str, int off, int len)	매개값으로 주어진 문자열에서 off 순번부터 len개까지의 문자를 보냅니다.
void	flush()	버퍼에 잔류하는 모든 문자를 출력합니다.
void	close()	출력 스트림을 닫습니다.

# Writer : write(int c)

## ❖ writer(int c)

- 매개 변수로 주어지는 int(4byte)에서 끝 2byte(1개 문자)만 출력 스트림으로 보냄.



- 예) 한 문자씩 출력하기 ->

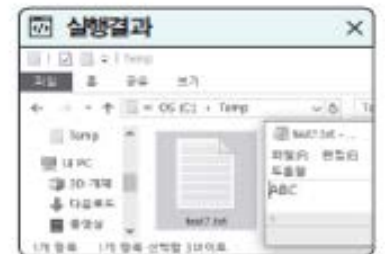
```
03 import java.io.FileWriter;
04 import java.io.Writer;
05
06 public class WriteExample {
07     public static void main(String[] args) throws Exception {
08         Writer writer = new FileWriter("C:/Temp/test7.txt");
09
10         char a = 'A';
11         char b = 'B';
12         char c = 'C';
13
14         writer.write(a);
15         writer.write(b);
16         writer.write(c);
17
18         writer.flush();
19         writer.close();
20     }
21 }
```

데이터 도착지를 test7.txt로 하는  
문자 기반 파일 출력 스트림을 생성

한 문자씩 출력

출력 버퍼에 잔류하는 모든 문자를 출력

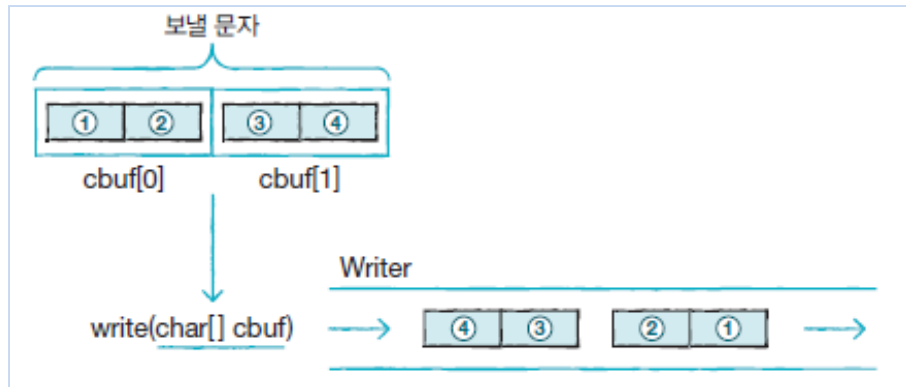
출력 스트림을 닫음



# Writer : write(char[] cbuf)

## ❖ write(char[] cbuf)

- 매개값으로 주어진 char[] 배열의 모든 문자를 출력 스트림으로 보냄.



- 예) 배열 전체를 출력하기 ->

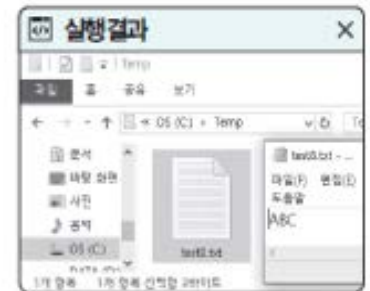
```
03 import java.io.FileWriter;
04 import java.io.Writer;
05
06 public class WriteExample {
07     public static void main(String[] args) throws Exception {
08         Writer writer = new FileWriter("C:/Temp/test8.txt");
09
10         char[] array = { 'A', 'B', 'C' };
11
12         writer.write(array);
13
14         writer.flush();
15         writer.close();
16     }
17 }
```

데이터 도착지를 test8.txt로 하는  
문자 기반 파일 출력 스트림을 생성

배열의 모든 문자를 출력

출력 버퍼에 잔류하는 모든 문자를 출력

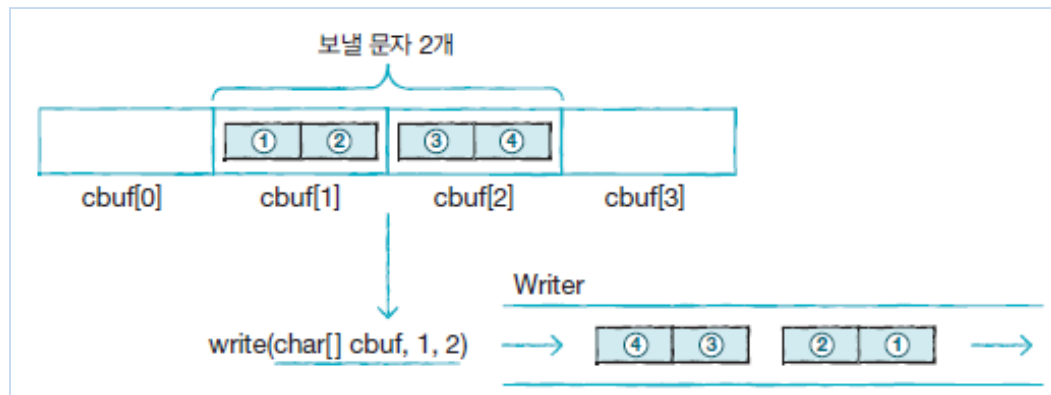
출력 스트림을 닫음



# Writer : write(char[] cbuf, int off, int len)

## ❖ write(char[] cbuf, int off, int len)

- c[off]부터 len개의 문자를 출력 스트림으로 보냄.



- 예) 배열의 일부를 출력하기 ->

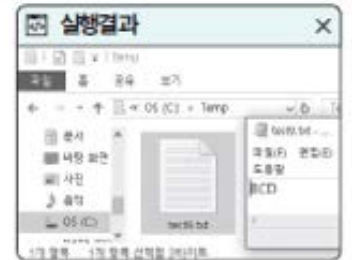
```
03 import java.io.FileWriter;
04 import java.io.Writer;
05
06 public class WriteExample {
07     public static void main(String[] args) throws Exception {
08         Writer writer = new FileWriter("C:/Temp/test9.txt");
09
10         char[] array = { 'A', 'B', 'C', 'D', 'E' };
11
12         writer.write(array, 1, 3);
13
14         writer.flush();
15         writer.close();
16     }
17 }
```

데이터 도착지를 test9.txt로 하는  
문자 기반 파일 출력 스트림을 생성

배열의 1번 인덱스부터  
3개를 출력

출력 버퍼에 잔류하는 모든 문자를 출력

출력 스트림을 닫음





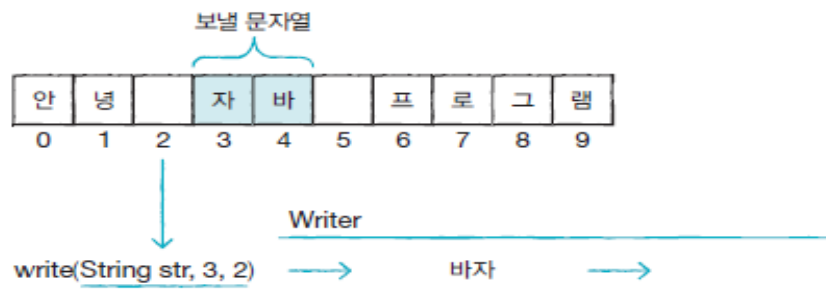
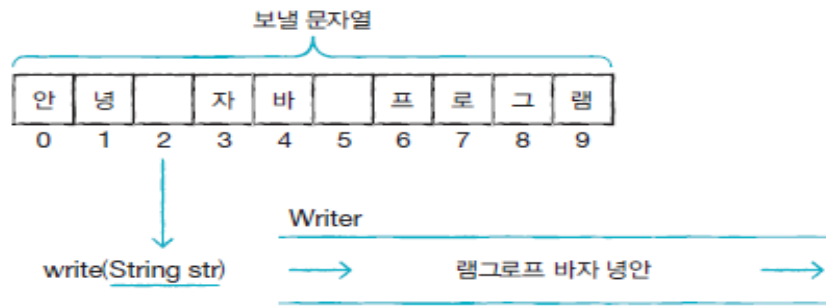
# Writer 문자열 출력

## ❖ write(String str)

- 문자열 전체를 출력 스트림으로 보냄

## ❖ write(String str, int off, int len)

- 주어진 문자열 off순번부터 len개 까지의 문자를 출력 스트림으로 보냄



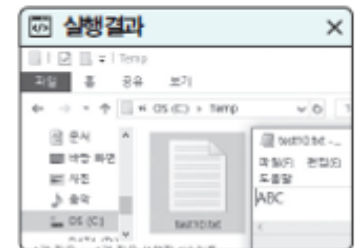
```
03 import java.io.FileWriter;
04 import java.io.Writer;
05
06 public class WriteExample {
07     public static void main(String[] args) throws Exception {
08         Writer writer = new FileWriter("C:/Temp/test10.txt");
09
10         String str = "ABC";
11
12         writer.write(str);
13
14         writer.flush();
15         writer.close();
16     }
17 }
```

데이터 도착지를 test10.txt로 하는  
문자 기반 파일 출력 스트림을 생성

문자열 전체를 출력

출력 버퍼에 잔류하는 모든 문자열을 출력

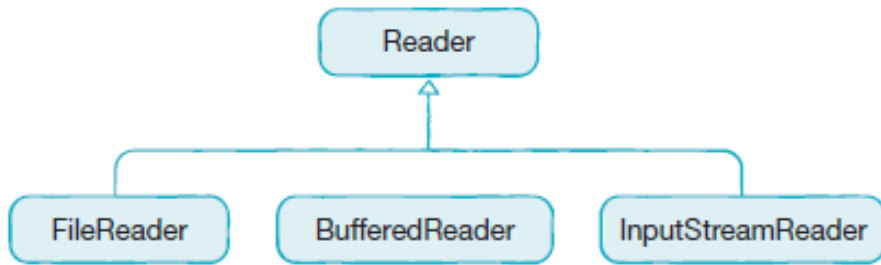
출력 스트림을 닫음



# 9절. 문자 입력 스트림

## ❖ Reader

- 문자 기반 입력 스트림의 최상위 클래스
- 모든 문자 기반 입력 스트림은 Reader 클래스를 상속받아 만들어짐.



리턴 타입	메소드	설명
int	<code>read()</code>	1개의 문자를 읽고 리턴합니다.
int	<code>read(char[] cbuf)</code>	읽은 문자들을 매개값으로 주어진 문자 배열에 저장하고 읽은 문자 수를 리턴합니다.
int	<code>read(char[] cbuf, int off, int len)</code>	len개의 문자를 읽고 매개값으로 주어진 문자 배열에서 <code>cbuf[off]</code> 부터 len개까지 저장합니다. 그리고 읽은 문자 수를 리턴합니다.
void	<code>close()</code>	입력 스트림을 닫습니다.

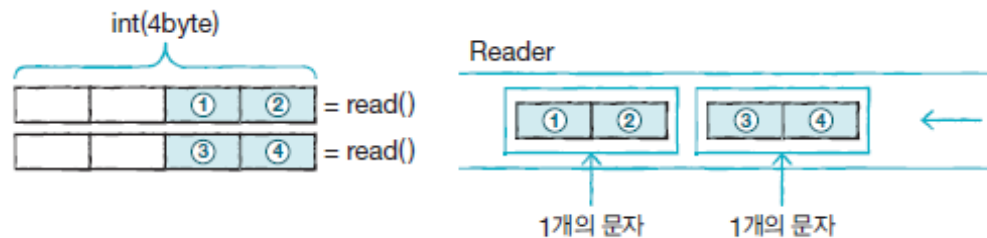


# Reader : read()

## ❖ read()

- 입력 스트림으로부터 1개의 문자(2byte) 읽어 int(4byte) 타입으로 리턴.
- 리턴된 4byte 중 끝 2byte에 문자 데이터가 들어 있음.
- 리턴된 int 값을 char 타입으로 변환하면 읽은 문자를 얻을 수 있음.

```
char charData = (char) read();
```



- 예) 한 문자씩 읽기 ->

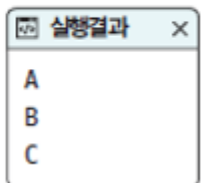
```
03 import java.io.FileReader;
04 import java.io.Reader;
05
06 public class ReadExample {
07     public static void main(String[] args) throws Exception {
08         Reader reader = new FileReader("C:/Temp/test7.txt");
09         while(true) {
10             int data = reader.read();
11             if(data == -1) break;
12             System.out.println((char)data);
13         }
14         reader.close();
15     }
16 }
```

데이터출발지를 test7.txt로 하는  
문자 기반 파일 입력 스트림을 생성

한 문자씩 읽기

파일 끝에 도달했을 경우

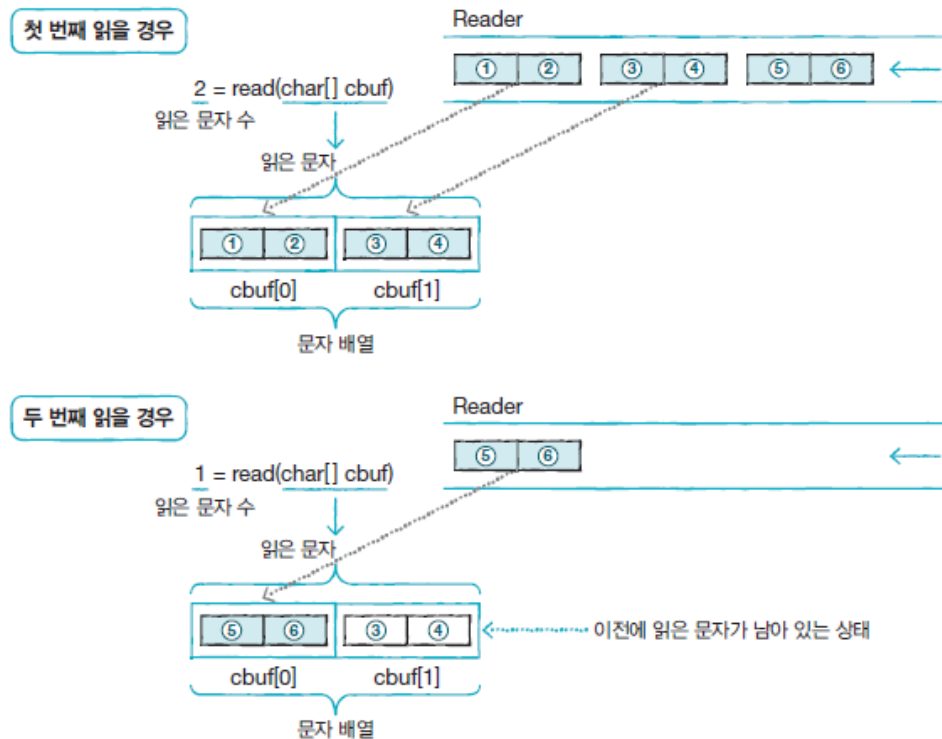
입력 스트림을 닫음



# Reader : read(char[] cbuf)

## ❖ read(char[] cbuf)

- 입력 스트림으로부터 매개값으로 주어진 문자 배열의 길이만큼 문자를 읽고 배열에 저장.
- 읽은 문자 수를 리턴.



```
03 import java.io.FileReader;
04 import java.io.Reader;
05
06 public class ReadExample {
07     public static void main(String[] args) throws Exception {
08         Reader reader = new FileReader("C:/Temp/test8.txt");
09
10         char[] buffer = new char[100];
11
12         while(true) {
13             int readCharNum = reader.read(buffer);
14             if(readCharNum == -1) break;
15             for(int i=0; i<readCharNum; i++) {
16                 System.out.println(buffer[i]);
17             }
18         }
19
20         reader.close();
21     }
22 }
```

데이터 출력을 test8.txt로 하는 문자 기반 파일 입력 스트림을 생성

길이 100인 배열 생성

배열 길이만큼 읽기

파일 끝에 도달했을 경우

읽은 문자 수만큼 반복하면서 배열에 저장된 문자를 출력

입력 스트림을 닫음

**실행결과**

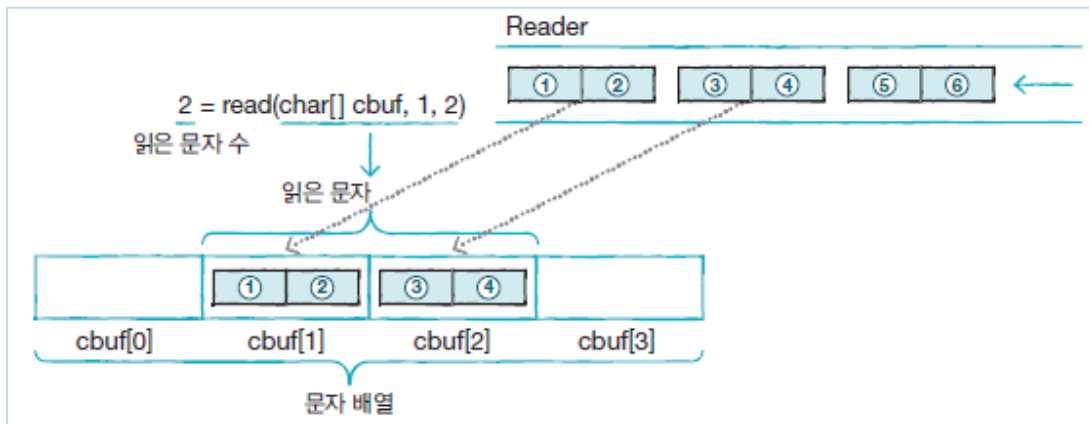
```
A
B
C
```

- 예) 배열 길이 만큼 읽기 ->

# Reader : read(char[] cbuf, int off, int len)

## ❖ read(char[] cbuf, int off, int len) 메소드

- 입력 스트림으로부터 len개의 문자만큼 읽어, 매개값으로 주어진 배열에 cbuf[off]부터 len개까지 저장.
- 읽은 문자 수를 리턴.



- 예) 지정한 길이만큼 읽기 ->

```
03 import java.io.FileReader;
04 import java.io.Reader;
05
06 public class ReadExample {
07     public static void main(String[] args) throws Exception {
08         Reader reader = new FileReader("C:/Temp/test9.txt");
09
10         char[] buffer = new char[5];
11
12         int readCharNum = reader.read(buffer, 2, 3);
13         if(readCharNum != -1) {
14             for(int i=0; i<buffer.length; i++) {
15                 System.out.println(buffer[i]);
16             }
17         }
18
19         reader.close();
20     }
21 }
```

데이터 출발지를 test9.txt로 하는 문자 기반 파일 입력 스트림을 생성

입력 스트림으로부터 3개의 문자를 읽고 buffer[2], buffer[3], buffer[4]에 각각 저장

읽은 문자가 있다면

배열 전체를 읽고 출력

입력 스트림을 닫음

실행결과

```
B
C
D
```

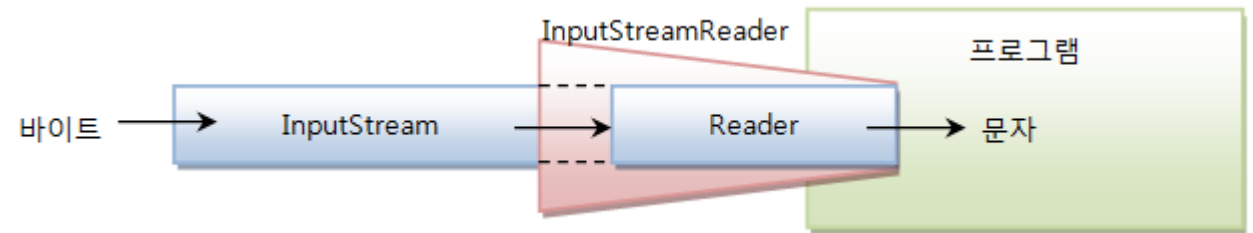
# 10절. 문자 단위 보조 스트림

## ❖ 문자 변환 보조 스트림 InputStreamReader / OutputStreamWriter

- 소스 스트림이 바이트 기반 스트림이지만 데이터가 문자일 경우 사용
- Reader와 Writer는 문자 단위로 입출력 - 바이트 기반 스트림보다 편리
- 문자 셋의 종류를 지정할 수 있기 때문에 다양한 문자 입출력 가능

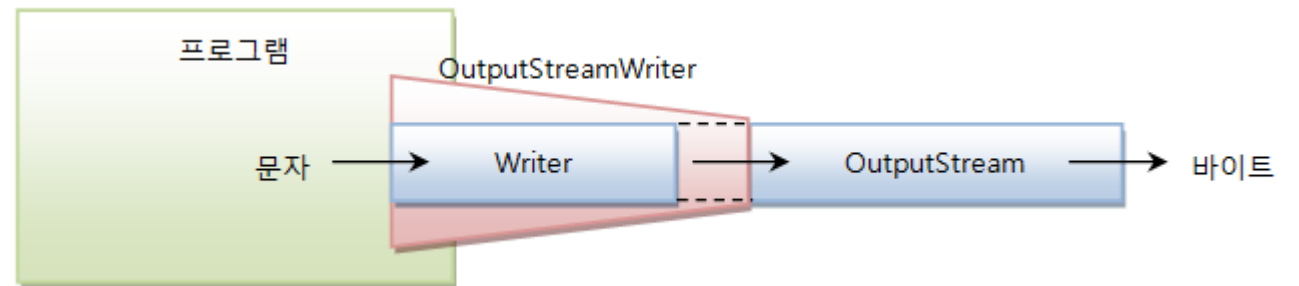
## ❖ InputStreamReader

```
InputStream is = ...;  
InputStreamReader reader = new InputStreamReader(is);
```



## ❖ OutputStreamWriter

```
FileOutputStream fos = new FileOutputStream("C:/Temp/test1.txt");  
Writer writer = new OutputStreamWriter(fos);
```



# 문자 변환 보조 스트림 예

```
03 import java.io.FileInputStream;
04 import java.io.FileOutputStream;
05 import java.io.InputStreamReader;
06 import java.io.OutputStreamWriter;
07 import java.io.Reader;
08 import java.io.Writer;
09
10 public class CharacterConvertStreamExample {
11     public static void main(String[] args) throws Exception {
12         write("문자 변환 스트림을 사용합니다.");
13         String data = read();
14         System.out.println(data);
15     }
16
17     public static void write(String str) throws Exception {
18         FileOutputStream fos = new FileOutputStream("C:/Temp/test1.txt");
19         Writer writer = new OutputStreamWriter(fos);
```

FileOutputStream에  
OutputStreamWriter  
보조 스트림을 연결

```
20     writer.write(str);
21     writer.flush();
22     writer.close();
23 }
24
25 public static String read() throws Exception {
26     FileInputStream fis = new FileInputStream("C:/Temp/test1.txt");
27     Reader reader = new InputStreamReader(fis);
28     char[] buffer = new char[100];
29     int readCharNum = reader.read(buffer);
30     reader.close();
31     String data = new String(buffer, 0, readCharNum);
32     return data;
33 }
34 }
```

OutputStreamWriter 보조 스트림을  
이용해서 문자 출력

FileInputStream에  
InputStreamReader  
보조 스트림을 연결

InputStreamReader 보조 스트림을  
이용해서 문자 입력

char 배열에서 읽은  
수만큼 문자열로 변환

실행결과 X  
문자 변환 스트림을 사용합니다.

# 문자 단위 성능 향상 보조 스트림

## ❖ BufferedReader / BufferedWriter

- BufferedReader 보조 스트림을 생성하고 readLine()으로 반복해서 읽으면,
- 키보드에서 입력한 [Enter]키 이전의 모든 문자열 및 파일의 내용을 라인 단위로 읽을 수 있음.

```
03 import java.io.*;
04
05 public class ReadLineExample {
06     public static void main(String[] args) throws Exception {
07         Reader reader = new FileReader(
08             ReadLineExample.class.getResource("language.txt").getPath()
09         );
10         BufferedReader br = new BufferedReader(reader);
11
12         while(true) {
13             String data = br.readLine();
14             if(data == null) break;
15             System.out.println(data);
16         }
17
18         br.close();
19     }
20 }
```

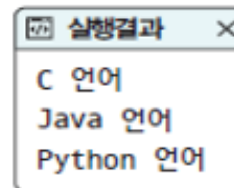
문자 기반  
입력 스트림 얻기

BufferedReader 보조 스트림 연결

라인 단위 문자열을 읽고 리턴

파일 끝에 도달했을 경우

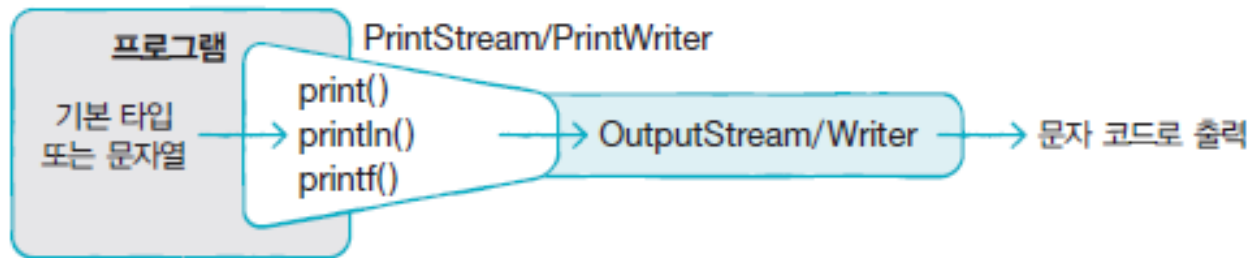
입력 스트림 닫기



# 11절. 프린터 보조 스트림

## ❖ PrintStream / PrintWriter

- 바이트 기반 출력 스트림 / 문자 기반 출력 스트림과 연결
- 프린터와 유사하게 출력하는 print(), println() 메소드 가진 보조 스트림



```
PrintStream ps = new PrintStream(바이트 기반 출력 스트림);  
PrintWriter pw = new PrintWriter(문자 기반 출력 스트림);
```



# PrintStream 예

## ❖ 라인 단위로 출력하기

```
03 import java.io.FileOutputStream;
```

```
04 import java.io.PrintStream;
```

```
05
```

```
06 public class PrintStreamExample {
```

```
07     public static void main(String[] args) throws Exception {
```

```
08         FileOutputStream fos = new FileOutputStream("C:/Temp/printstream.txt");
```

```
09         PrintStream ps = new PrintStream(fos);
```

```
10
```

```
11         ps.println("[프린터 보조 스트림]");
```

```
12         ps.print("마지 ");
```

```
13         ps.println("프린터가 출력하는 것처럼 ");
```

```
14         ps.println("데이터를 출력합니다.");
```

```
15
```

```
16         ps.flush();
```

```
17         ps.close();
```

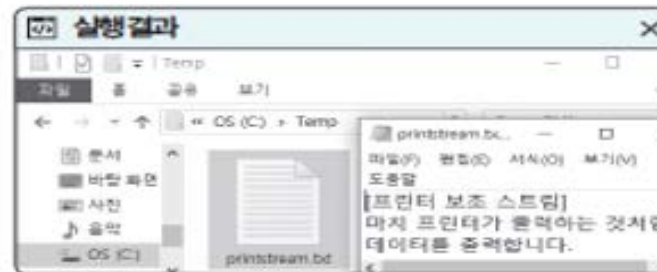
```
18     }
```

```
19 }
```

바이트 기반 출력 스트림을 생성하고  
PrintStream 보조 스트림 연결

라인 단위로 문자열 출력

버퍼에 잔류하는  
문자열을 모두 보내고,  
출력 스트림 닫음





# 12절. 네트워크 -> 컴퓨터 네트워크 교과목

## ❖ 네트워크

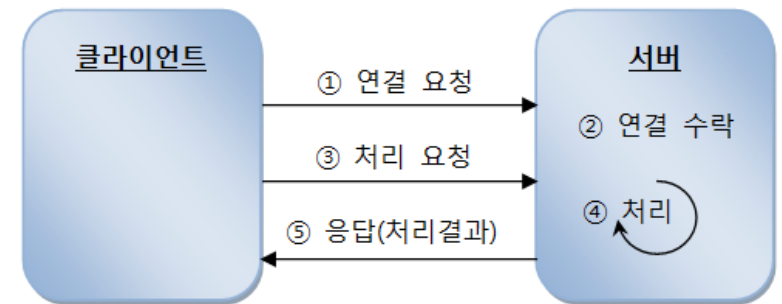
- 여러 대의 컴퓨터를 통신 회선으로 연결한 것
- 홈 네트워크, 지역 네트워크, 인터넷

## ❖ 서버

- 서비스를 제공하는 프로그램
- 클라이언트의 연결을 수락하고, 요청 내용을 처리한 후 응답 보내는 역할
- 웹 서버, FTP서버, DBMS, 메신저 서버

## ❖ 클라이언트

- 서비스를 받는 프로그램
- 웹 브라우저, FTP 클라이언트, 메신저
- 네트워크 데이터를 필요로 하는 모든 애플리케이션(모바일 앱 포함)



# 학습 정리 1

---

## ❖ 입출력 스트림

- 자바에서 데이터는 스트림을 통해 입출력 됨. 프로그램이 기준이 되어 입출력 스트림의 종류를 결정.

## ❖ InputStream

- 바이트 기반 입력 스트림의 최상위 클래스, 추상 클래스. 모든 바이트 기반 입력 스트림 클래스는 InputStream 클래스를 상속받아 만들어짐.

## ❖ OutputStream

- 바이트 기반 출력 스트림의 최상위 클래스, 추상 클래스. 모든 바이트 기반 출력 스트림 클래스는 OutputStream 클래스 상속받아 만들어짐.

## ❖ Reader

- 문자 기반 입력 스트림의 최상위 클래스, 추상 클래스. 모든 문자 기반 입력 스트림 클래스는 Reader 클래스 상속받아 만들어짐.

## ❖ Writer

- 문자 기반 출력 스트림의 최상위 클래스, 추상 클래스. 모든 문자 기반 출력 스트림 클래스는 Writer 클래스 상속받아 만들어짐.

# 학습 정리 2

## ❖ System.in

- 자바는 콘솔에서 키보드의 데이터를 입력 받을 수 있도록 System 클래스의 in 정적 필드 제공함. 주로 InputStreamReader 보조스트림과 BufferedReader 보조스트림을 연결해서 사용하거나 Scanner를 이용해서 입력된 문자열을 읽음.

## ❖ System.out

- 콘솔에서 모니터로 데이터를 출력하기 위해서는 System 클래스의 out 정적 필드 사용. PrintStream이 제공하는 print(), println(), printf()와 같은 메소드를 이용하여 모니터로 출력 가능.

## ❖ Scanner

- Scanner 클래스는 입출력 스트림도, 보조 스트림도 아님. Scanner는 문자 파일이나 바이트 기반 입출력 스트림에서 라인 단위 문자열을 쉽게 읽도록 하기 위해 java.util 패키지에서 제공하는 클래스.

## ❖ File

- java.io 패키지에서 제공하는 File 클래스는 파일 및 폴더 정보를 제공하는 역할.

## ❖ 보조 스트림

- 다른 스트림과 연결되어 여러 편리한 기능 제공. 자체적으로 입출력을 수행할 수 없기 때문에 입출력 소스와 바로 연결되는 InputStream, OutputStream, Reader, Writer 등에 연결해서 입출력 수행. 문자 변환, 입출력 성능 향상, 기본 타입 입출력 등의 기능을 제공.

# 학습 정리 3

---

## ❖ 문자 변환

- 소스 스트림이 바이트 기반 스트림이면서 입출력 데이터가 문자라면 Reader와 Writer로 변환해서 사용하는 것이 편리함. OutputStreamWriter는 Writer로 변환하는 보조 스트림, InputStreamReader는 Reader로 변환하는 보조 스트림.

## ❖ 성능 향상

- 메모리 버퍼를 추가로 제공하여 프로그램의 실행 성능을 향상시킴. 바이트 기반 스트림에서는 BufferedInputStream, BufferedOutputStream, 문자 기반 스트림에서는 BufferedReader, BufferedWriter 사용.

## ❖ 기본 타입 입출력

- 보조 스트림 DataInputStream과 DataOutputStream을 연결하면 기본 타입 boolean, char, short, int, long, float, double 입출력 가능.

## ❖ 개행 출력

- PrintStream/PrintWriter의 println() 메소드는 출력할 데이터 끝에 개행 문자인 '\n'을 추가하여 출력 시 콘솔이나 파일에서 줄 바꿈 일어남.

# 학습 정리 4

---

- ❖ 입출력 스트림에 대한 설명으로 맞는 것에 O, 틀린 것에 X 하세요.
  - 하나의 스트림으로 입력과 출력이 동시에 가능하다. (      )
  - 프로그램을 기준으로 데이터가 들어오면 입력 스트림이다. (      )
  - 프로그램을 기준으로 데이터가 나가면 출력 스트림이다. (      )
  - 파일에 데이터를 저장하려면 출력 스트림을 사용해야 한다. (      )
  
- ❖ InputStream과 Reader에 대한 설명으로 맞는 것에 O, 틀린 것에 X 하세요.
  - 이미지 데이터는 InputStream 또는 Reader로 모두 읽을 수 있다. (      )
  - Reader의 read() 메소드는 1문자를 읽고 리턴 한다. (      )
  - InputStream의 read() 메소드는 1byte를 읽고 리턴 한다. (      )
  - InputStream의 read(byte[] b) 메소드는 읽은 바이트 수를 리턴 한다. (      )

# 학습 정리 5

- ❖ 보조 스트림에 대한 설명 중 맞는 것에 O, 틀린 것에 X 하세요.
  - InputStreamReader는 InputStream을 Reader로 변환하는 보조 스트림이다. (     )
  - BufferedInputStream은 데이터 읽기 성능을 향상시키는 보조 스트림이다. (     )
  - DataInputStream은 객체를 입출력하는 보조 스트림이다. (     )
  - PrintStream은 라인 단위로 출력할 수 있는 보조 스트림이다. (     )
  
- ❖ 출력 스트림에서 데이터를 출력한 후 flush() 메소드를 호출하는 이유가 무엇입니까?
  1. 출력 스트림의 버퍼에 있는 데이터를 모두 출력하고 버퍼를 비운다.
  2. 출력 스트림을 메모리에서 제거한다.
  3. 출력 스트림의 버퍼에 있는 데이터를 모두 삭제한다.
  4. 출력 스트림을 닫는 역할을 한다.

# 적용 확인 학습 & 응용 프로그래밍

---

- ❖ 다음 파일에 있는 문제들의 해답을 스스로 작성 해 보신 후 개념 & 적용 확인 학습 영상을 학습 하시기 바랍니다.
  - java\_13장\_입출력\_ex.pdf
- ❖ 퀴즈와 과제가 출제되었습니다.
  - 영상 수업을 학습하신 후 과제와 퀴즈를 수행 하시기 바랍니다.

# Q & A

- ❖ “입출력”에 대한 학습이 모두 끝났습니다.
- ❖ 모든 내용을 이해 하셨나요?
- ❖ 아직 이해가 안되는 내용이 있다면 다시 한번 복습하시기 바랍니다.
- ❖ 질문은 한림 SmartLEAD 쪽지 또는 e-mail 또는 전화상담을 이용하시기 바랍니다.



- ❖ 퀴즈와 과제가 출제되었습니다. 마감시간에 늦지 않도록 주의해 주세요.
- ❖ 한 학기 열심히 공부해 준 여러분을 칭찬합니다^^.
- ❖ 수고하셨습니다.^^