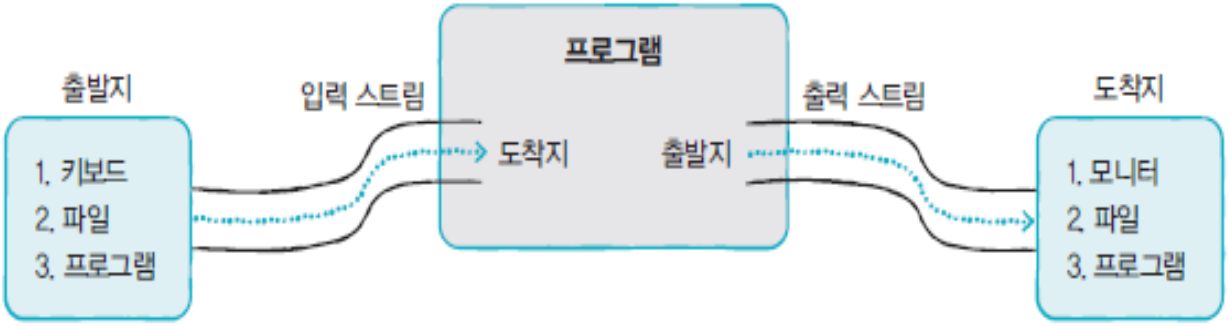
■ 입출력 범위, 입출력 모델

일반적인 입출력의 대상

- 키보드와 모니터
- 하드디스크에 저장되어 있는 파일
- USB와 같인 외부 메모리 장치
- 네트워크로 연결되어 있는 컴퓨터
- 사운드카드, 오디오카드와 같은 멀티미디어 장치
- 프린터, 팩시밀리와 같은 출력장치

입출력 대상이 달라지면 프로그램상에서의 입출력 방식도 달라지는 것이 보통이다. 그런데 자바에서는 입출력 대상에 상관없이 입출력의 진행 방식이 동일하도록 별도의 'I/O 모델'을 정의하고 있다.

I/O 모델의 정의로 인해서 입출력 대상의 차이에 따른 입출력 방식의 차이는 크지 않다. 기본적인 입출력의 형태는 동일하다. 그리고 이것이 JAVA의 I/O 스트림이 갖는 장점이다.



■ 입출력 스트림의 종류

● Byte Stream

- 그림, 멀티미디어 등의 바이너리 데이터 읽고 쓰기

● Character Stream

- 문자(텍스트) 데이터 읽고 쓰기

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

■ Byte Stream - 파일 읽기

ch16.ByteStreamExam

```
InputStream in = new FileInputStream("byte-stream-data.txt");

int letter = in.read();
System.out.printf("ASCII 값 : %s\n", letter);
System.out.printf("문자 형태로 변환된 값 %s\n", (char) letter);

letter = in.read();
System.out.printf("ASCII 값 : %s\n", letter);
System.out.printf("문자 형태로 변환된 값 %s\n", (char) letter);

in.close();
```

byte-stream-data.txt

```
python
java
c++
go
kotlin
fortran
```

```
ASCII 값 : 112
문자 형태로 변환된 값 p
ASCII 값 : 121
문자 형태로 변환된 값 y
```

■ Character Stream - 파일 읽기

ch16.CharacterStreamExam

```
Reader reader = new FileReader("byte-stream-data.txt");

int letter = reader.read();
System.out.printf("ASCII 값 : %s\n", letter);
System.out.printf("문자 형태로 변환된 값 %s\n", (char) letter);

letter = reader.read();
System.out.printf("ASCII 값 : %s\n", letter);
System.out.printf("문자 형태로 변환된 값 %s\n", (char) letter);

reader.close();
```

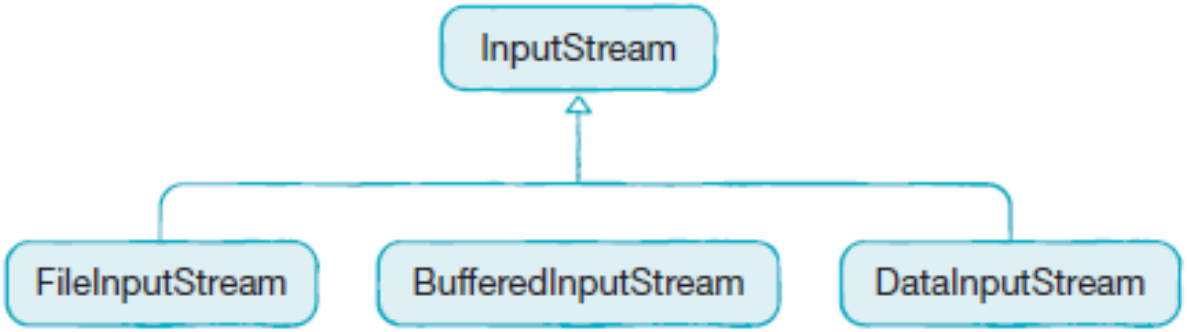
byte-stream-data.txt

```
python
java
c++
go
kotlin
fortran
```

```
ASCII 값 : 112
문자 형태로 변환된 값 p
ASCII 값 : 121
문자 형태로 변환된 값 y
```

■ Byte Stream - InputStream

● 바이트 기반 입력 스트림의 최상위 클래스

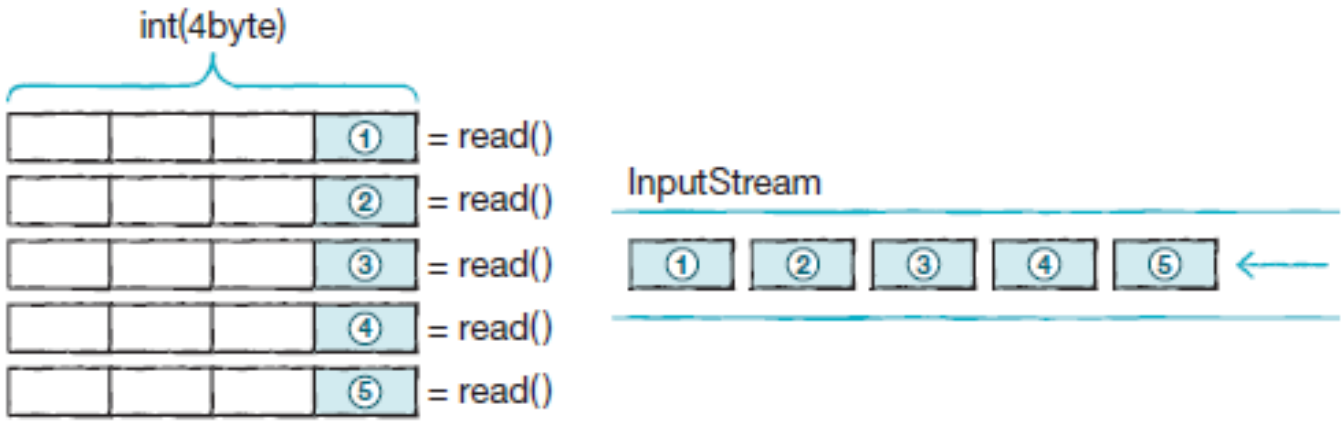


리턴 타입	메소드	설명
int	read()	1byte를 읽고 읽은 바이트를 리턴합니다.
int	read(byte[ ] b)	읽은 바이트를 매개값으로 주어진 배열에 저장하고 읽은 바이트 수를 리턴합니다.
int	read(byte[ ] b, int off, int len)	len개의 바이트를 읽고 매개값으로 주어진 배열에서 b[off]부터 len개까지 저장합니다. 그리고 읽은 바이트 수를 리턴합니다.
void	close()	입력 스트림을 닫습니다.

■ Byte Stream - InputStream

● read() 메소드

- 더 이상 읽을 수 없는 상태가 되면 -1 반환 (파일 내용의 끝)



■ Byte Stream - InputStream

● read() 메소드

ch16.ByteStream1

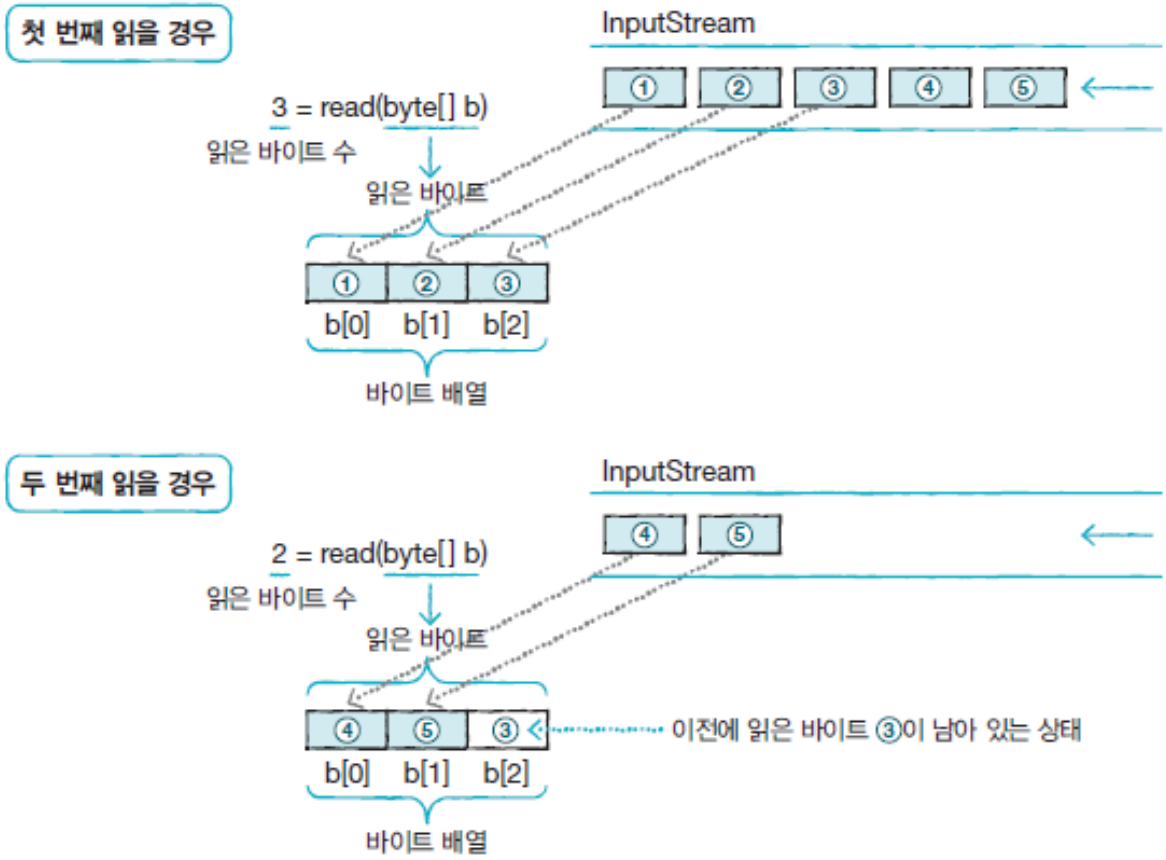
```
InputStream in = new FileInputStream("byte-stream-data.txt");

while(true) {
  int data = in.read();

  if(data == -1) break;

  System.out.printf("%s", (char) data);
}

in.close();
```

```
python
java
c++
go
kotlin
fortran
```

■ Byte Stream - InputStream

● read(byte[] b) 메소드

- 더 이상 읽을 수 없는 상태가 되면 -1 반환 (파일 내용의 끝)



■ Byte Stream - InputStream

● read(byte[] b) 메소드

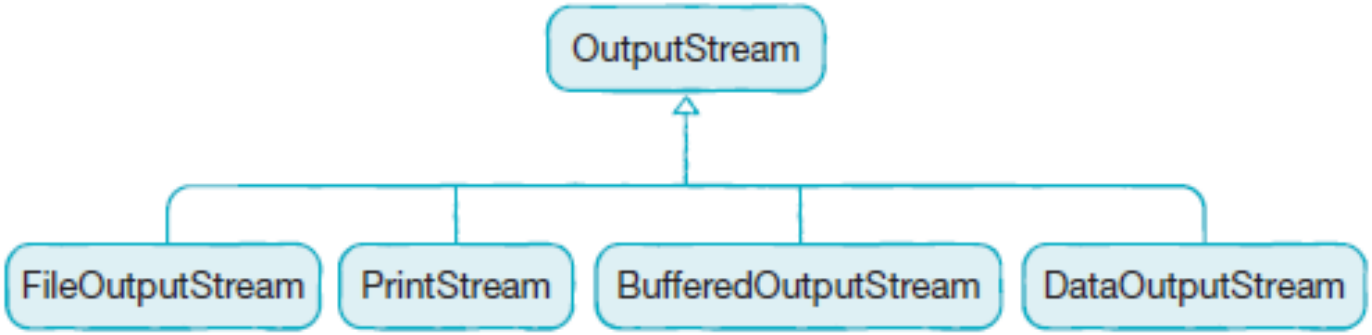
ch16.ByteStream2

```
InputStream in = new FileInputStream("byte-stream-data.txt");

byte[] buffer = new byte[8];

while(true) {
  int readByteNum = in.read(buffer);

  if(readByteNum == -1) break;

  for(int i = 0; i < readByteNum; i++) {
    System.out.printf("%s", (char) buffer[i]);
  }
}

in.close();
```

```
python
java
c++
go
kotlin
fortran
```

■ Byte Stream - OutputStream

● 바이트 기반 출력 스트림의 최상위 클래스

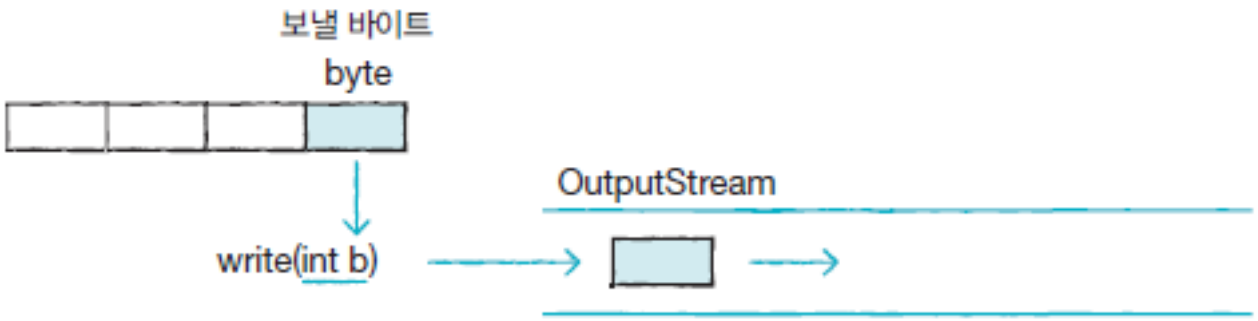


리턴 타입	메소드	설명
void	write(int b)	1byte를 출력합니다.
void	write(byte[ ] b)	매개값으로 주어진 배열 b의 모든 바이트를 출력합니다.
void	write(byte[ ] b, int off, int len)	매개값으로 주어진 배열 b[off]부터 len개까지의 바이트를 출력합니다.
void	flush( )	출력 버퍼에 잔류하는 모든 바이트를 출력합니다.
void	close( )	출력 스트림을 닫습니다.

■ Byte Stream - OutputStream

● write(int b) 메소드

- 1Byte 데이터 쓰기

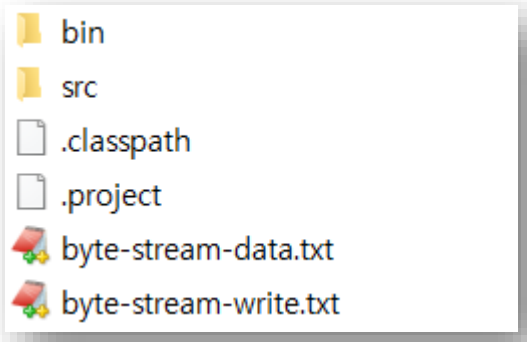


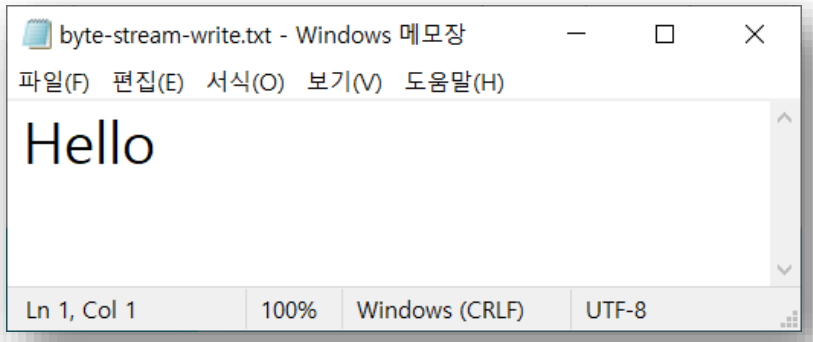
■ Byte Stream - OutputStream

● write(int b) 메소드

ch16.ByteStream3

```
OutputStream out = new FileOutputStream("byte-stream-write.txt");

out.write((int)'H');
out.write((int)'e');
out.write((int)'l');
out.write((int)'l');
out.write((int)'o');

out.flush();
out.close();
```

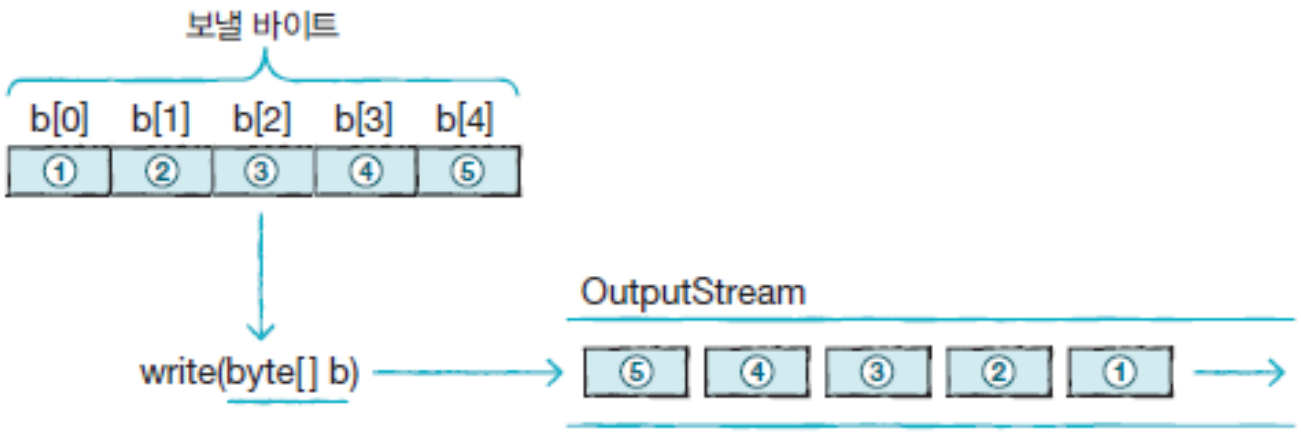




■ Byte Stream - OutputStream

● write(byte[] b) 메소드

- byte 배열의 크기만큼 데이터 쓰기

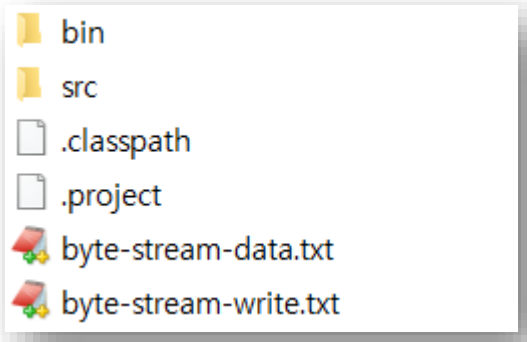


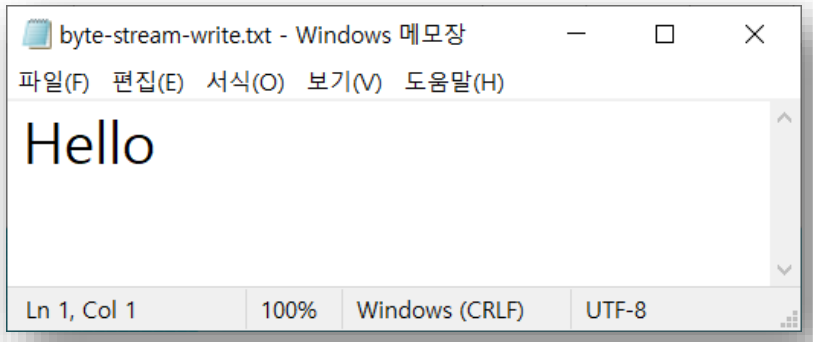
■ Byte Stream - OutputStream

● write(byte[] b) 메소드

ch16.ByteStream4

```
OutputStream out = new FileOutputStream("byte-stream-write.txt");

byte[] array = {
  (byte)'H', (byte)'e', (byte)'l', (byte)'l', (byte)'o'
};

out.write(array);

out.flush();
out.close();
```





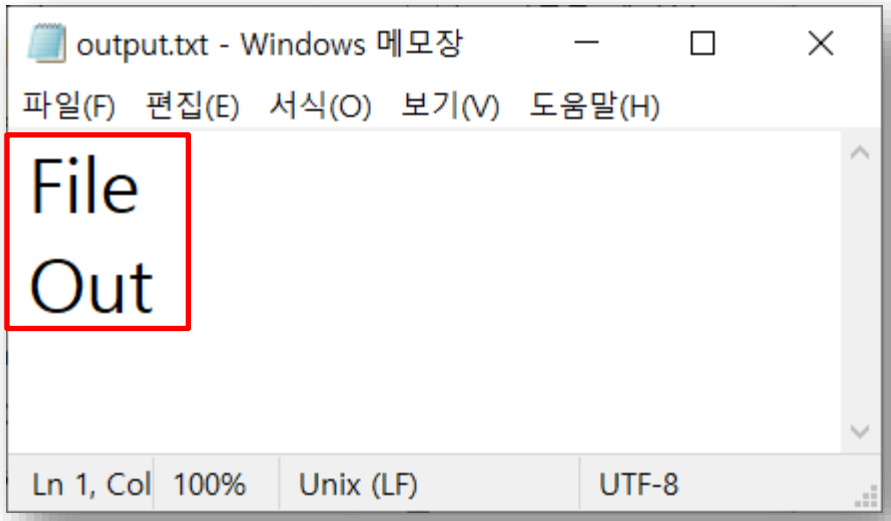
■ 연습문제 (ch16.연습문제01)

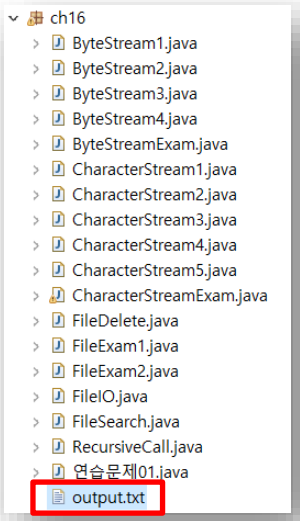
● FileOutputStream을 사용하여 문자열을 파일로 출력하기

- 파일 위치 ch16, 파일명 output.txt
- 줄바꿈 문자 '₩n'

```
OutputStream out = (   ①   );

(   ②   );

out.flush();
out.close();
```

결과





■ 연습문제 (ch16.연습문제02)

● FileInputStream을 사용하여 파일 내용 읽어오기

- 파일 위치 data, 파일명 harry_potter.txt

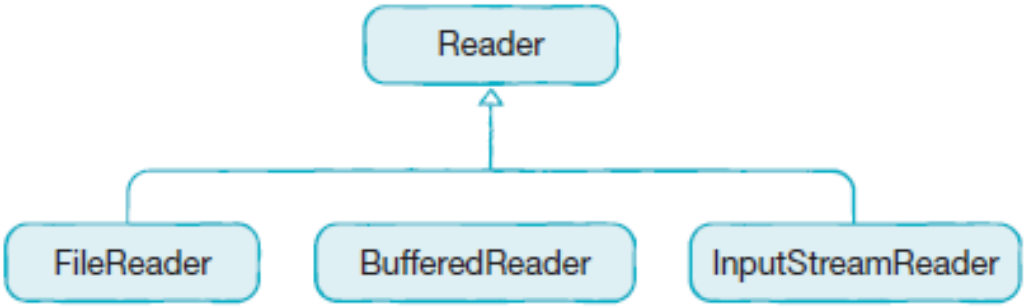
```
InputStream in = (   ①   );

(   ②   );

in.close();
```

결과

```
Harry Potter and the Sorcerer's Stone
CHAPTER ONE
THE BOY WHO LIVED
Mr. and Mrs. Dursley, of number four, Privet Drive, were proud to say
that they were perfectly normal, thank you very much. They were the last
people you'd expect to be involved in anything strange or mysterious,
because they just didn't hold with such nonsense.
Mr. Dursley was the director of a firm called Grunnings, which made
drills. He was a big, beefy man with hardly any neck, although he did
have a very large mustache. Mrs. Dursley was thin and blonde and had
nearly twice the usual amount of neck, which came in very useful as she
spent so much of her time craning over garden fences, spying on the
```

■ Character Stream - Reader

● 문자 기반 입력 스트림의 최상위 클래스

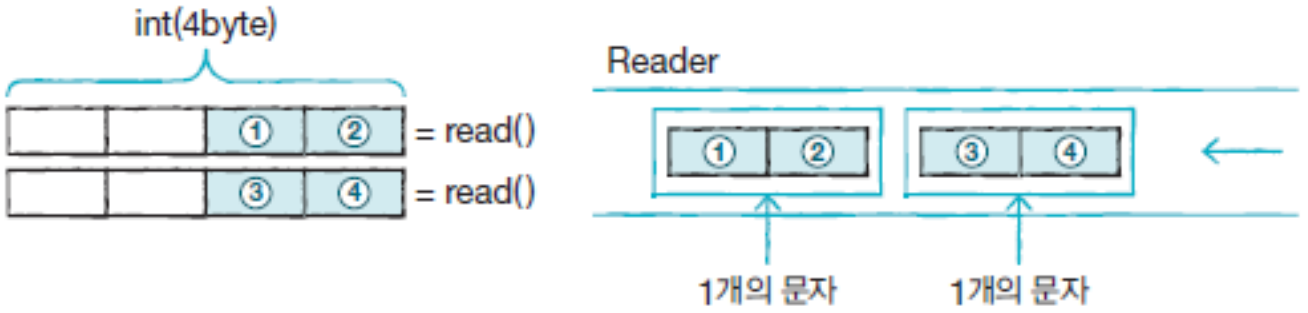


리턴 타입	메소드	설명
int	read()	1개의 문자를 읽고 리턴합니다.
int	read(char[] cbuf)	읽은 문자들을 매개값으로 주어진 문자 배열에 저장하고 읽은 문자 수를 리턴합니다.
int	read(char[] cbuf, int off, int len)	len개의 문자를 읽고 매개값으로 주어진 문자 배열에서 cbuf[off] 부터 len개까지 저장합니다. 그리고 읽은 문자 수를 리턴합니다.
void	close()	입력 스트림을 닫습니다.

■ Character Stream - Reader

● read() 메소드

- 더 이상 읽을 수 없는 상태가 되면 -1 반환 (파일 내용의 끝)



■ Character Stream - Reader

● read() 메소드

ch16.CharacterStream1

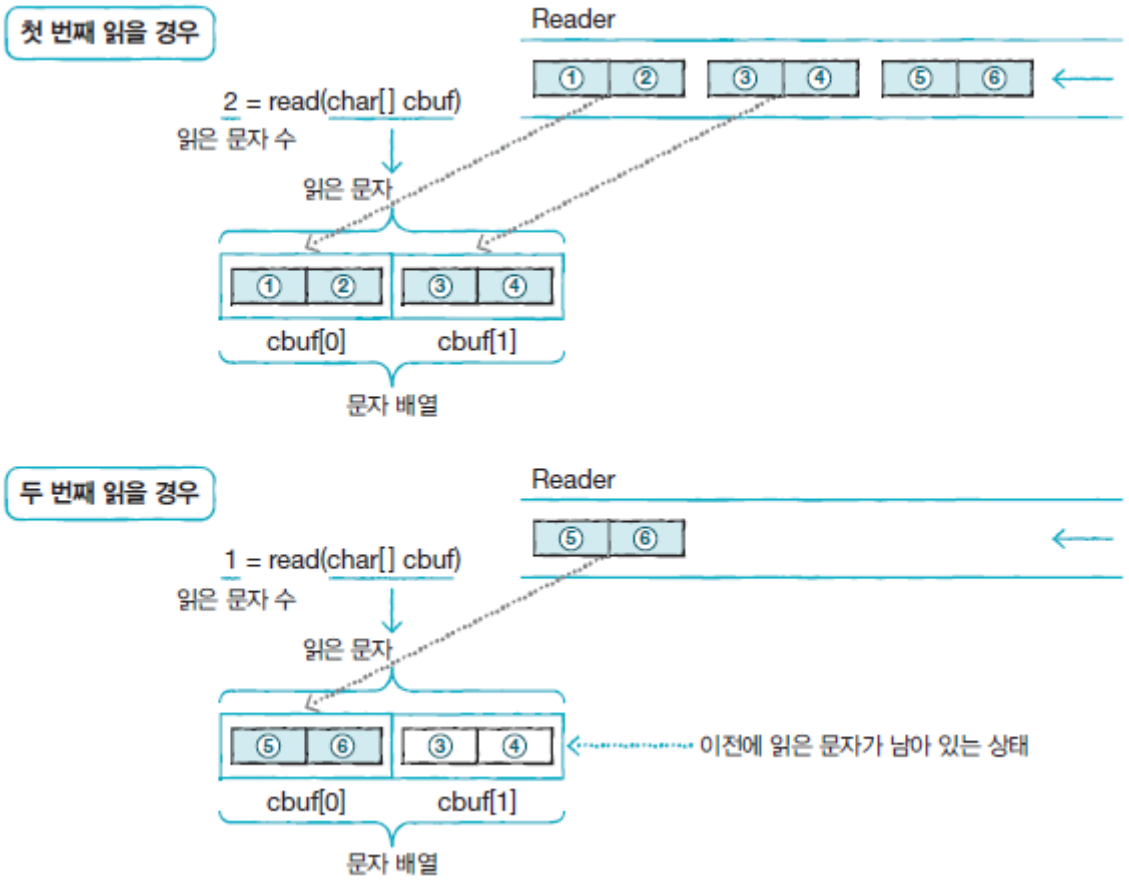
```
Reader reader = new FileReader("character-stream-data.txt");

while(true) {
  int data = reader.read();

  if(data == -1) break;

  System.out.printf("%s", (char) data);
}

reader.close();
```

파이썬
자바
씨++
고
코틀린
포트란

■ Character Stream - Reader

● read(char[] cbuf) 메소드

- 더 이상 읽을 수 없는 상태가 되면 -1 반환 (파일 내용의 끝)



■ Character Stream - Reader

● read(char[] cbuf) 메소드

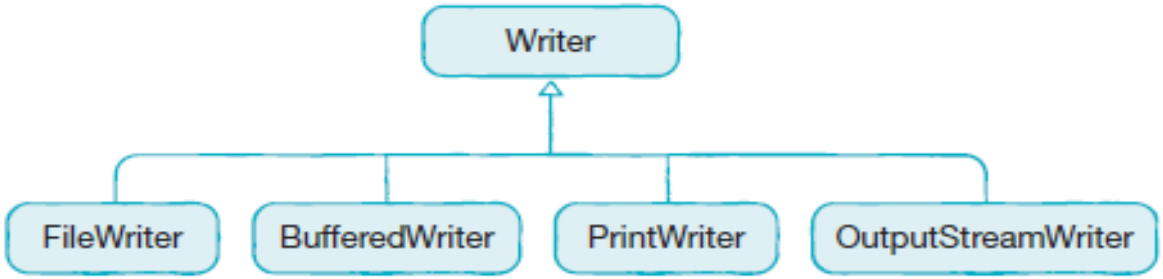
ch16.CharacterStream2

```
Reader reader = new FileReader("character-stream-data.txt");

char[] buffer = new char[100];

while(true) {
  int readCharNum = reader.read(buffer);

  if(readCharNum == -1) break;

  for(int i = 0; i < readCharNum; i++) {
    System.out.printf("%s", (char) buffer[i]);
  }
}
```

```
파이썬
자바
씨++
고
코틀린
포트란
```

■ Character Stream - Writer

● 문자 기반 출력 스트림의 최상위 클래스

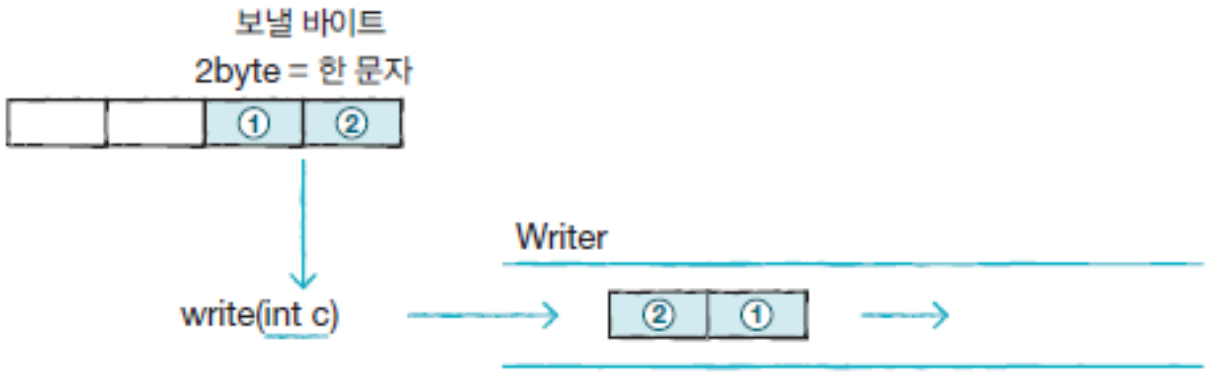


리턴 타입	메소드	설명
void	write(int c)	매개값으로 주어진 한 문자를 보냅니다.
void	write(char[ ] cbuf)	매개값으로 주어진 배열의 모든 문자를 보냅니다.
void	write(char[ ] cbuf, int off, int len)	매개값으로 주어진 배열에서 cbuf[off]부터 len개까지의 문자를 보냅니다.
void	write(String str)	매개값으로 주어진 문자열을 보냅니다.
void	write(String str, int off, int len)	매개값으로 주어진 문자열에서 off 순번부터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

■ Character Stream - Writer

● write(int c) 메소드

- 1Byte 데이터 쓰기

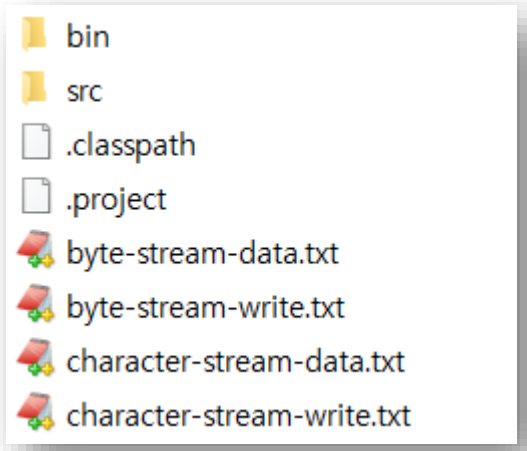


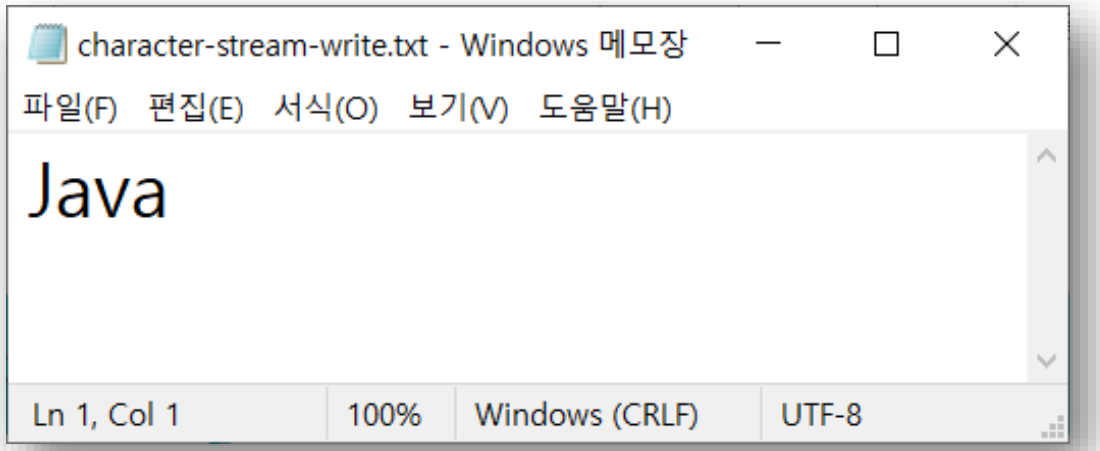
■ Character Stream - Writer

● write(int c) 메소드

ch16.CharacterStream3

```
Writer writer = new FileWriter("character-stream-write.txt");

writer.write('J');
writer.write('a');
writer.write('v');
writer.write('a');

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

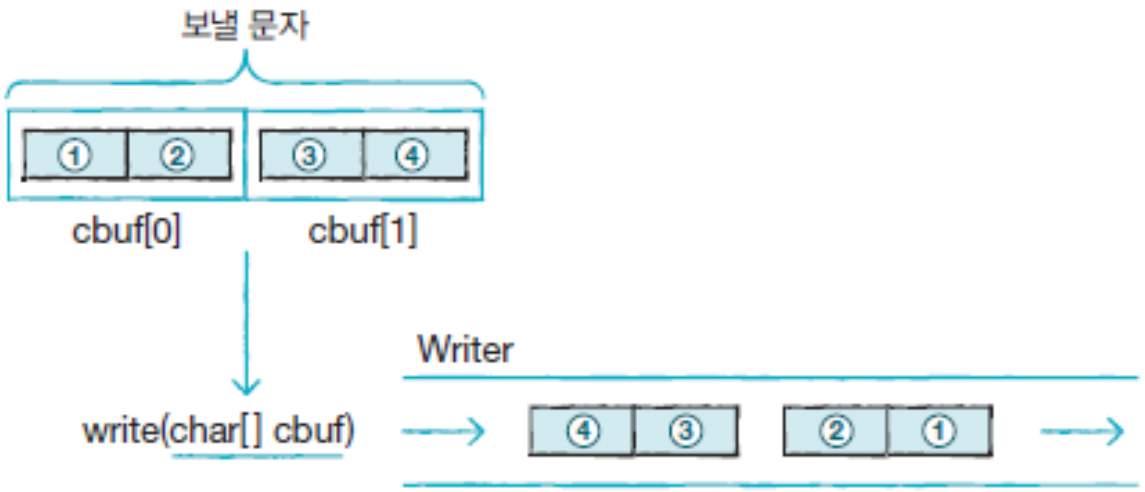




■ Character Stream - Writer

● write(char[] cbuf) 메소드

- char 배열의 크기만큼 데이터 쓰기

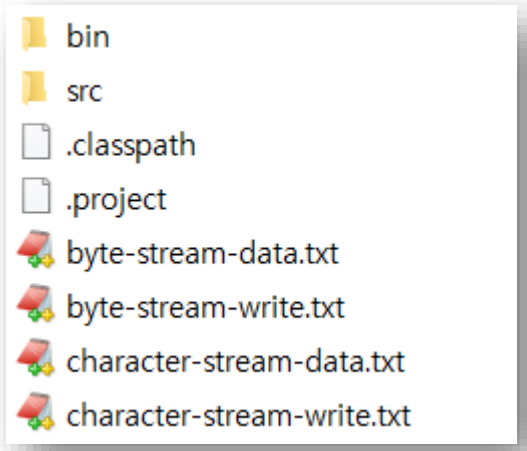


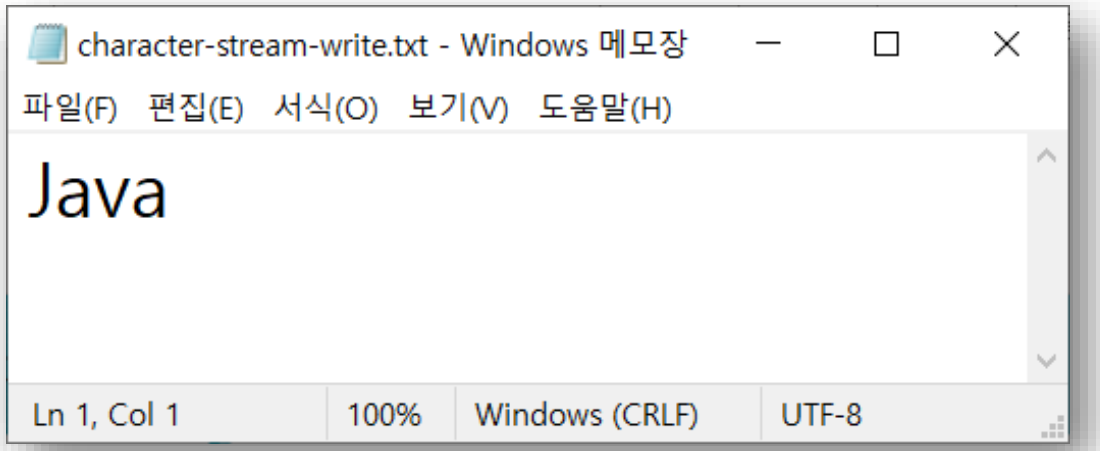
■ Character Stream - Writer

● write(char[] cbuf) 메소드

ch16.CharacterStream4

```
Writer writer = new FileWriter("chracter-stream-write.txt");

char[] array = {
  (int)'J', (int)'a', (int)'v', (int)'a'
};

writer.write(array);

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

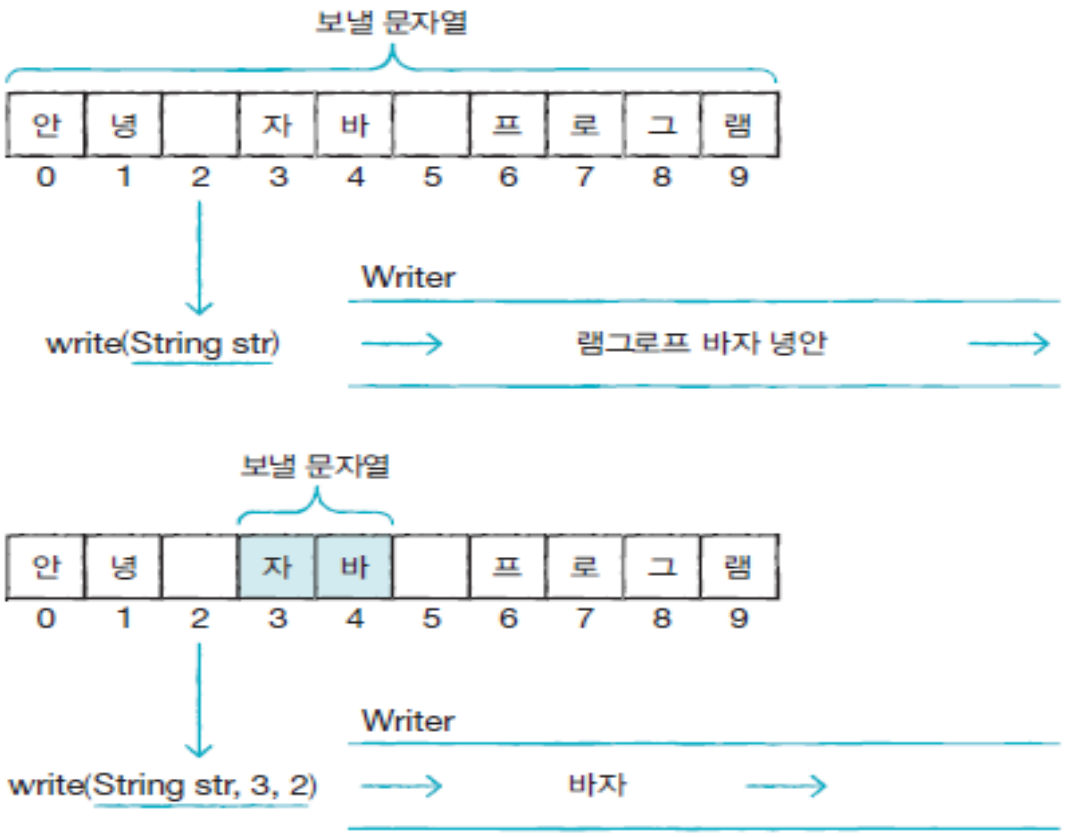




■ Character Stream - Writer

● write(String str) 메소드

- 문자열 데이터 쓰기

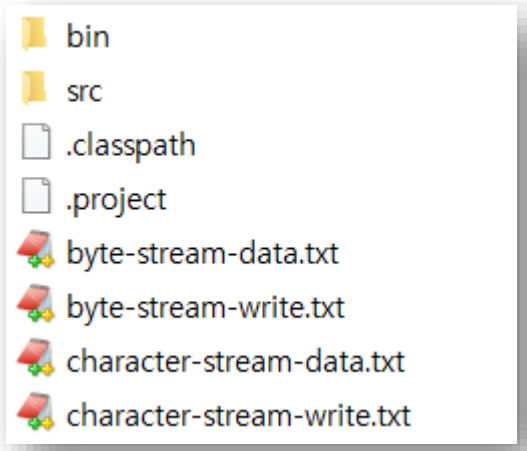


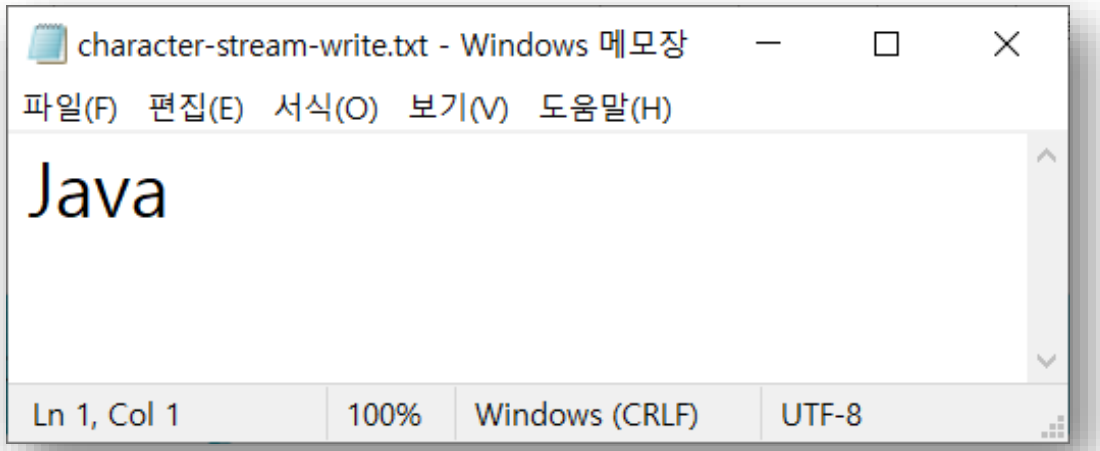
■ Character Stream - Writer

● write(String str) 메소드

ch16.CharacterStream5

```
Writer writer = new FileWriter("chracter-stream-write.txt");

String str = "Java";

writer.write(str);

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





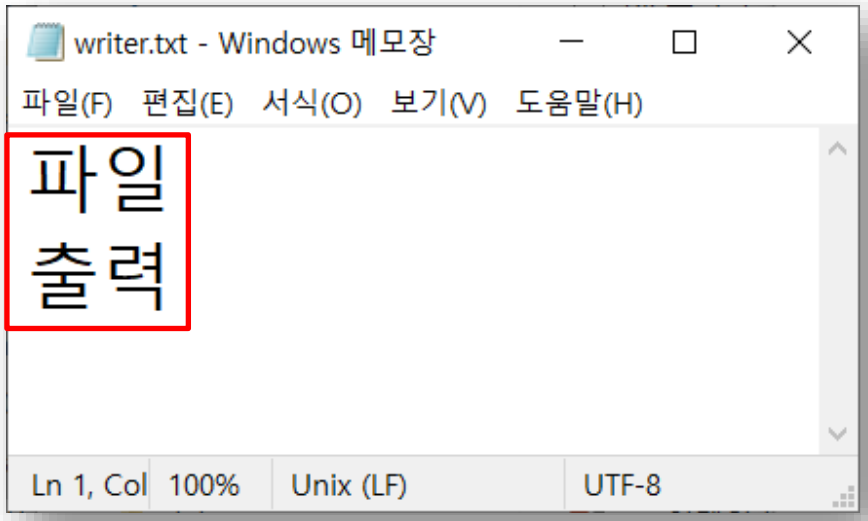
■ 연습문제 (ch16.연습문제03)

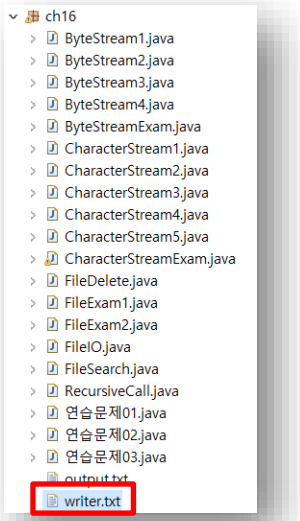
● FileWriter를 사용하여 문자열을 파일로 출력하기

- 파일 위치 ch16, 파일명 write.txt
- 줄바꿈 문자 '₩n'

```
Writer writer = (   ①   );

(   ②   );

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

결과





■ 연습문제 (ch16.연습문제04)

● FileReader를 사용하여 파일 내용 읽어오기

- 파일 위치 data, 파일명 운수좋은날.txt

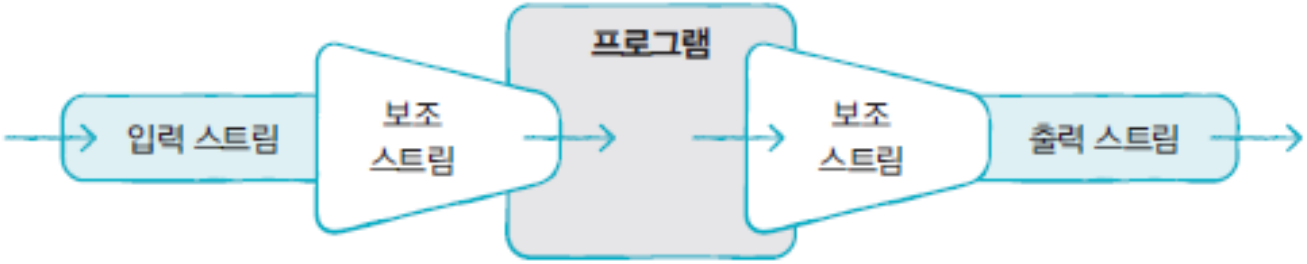
```
Reader reader = (   ①   );

(   ②   );

reader.close();
```

결과

```
새침하게 흐린 품이 눈이 올 듯하더니 눈은 아니 오고 얼다가 만 비가 추적추적 내리었다.

이날이야말로 동소문 안에서 인력거꾼 노릇을 하는 김 첨지에게는 오래간만에도 닥친 운수 좋은 날이

첫번에 삼십 전, 둘째 번에 오십 전 - 아침 댓바람에 그리 흔치 않은 일이었다. 그야말로 재수가

그의 아내가 기침으로 쿨럭거리기는 벌써 달포가 넘었다. 조밥도 굶기를 먹다시피 하는 형편이니 물

그때도 김 첨지가 오래간만에 돈을 얻어서 좁쌀 한 되와 십 전짜리 나무 한 단을 사다 주었더니 김

"에이, 오라질 년, 조롱복은 할 수가 없어, 못 먹어 병, 먹어서 병, 어쩌란 말이야! 왜 눈을 바
```

■ 보조 스트림

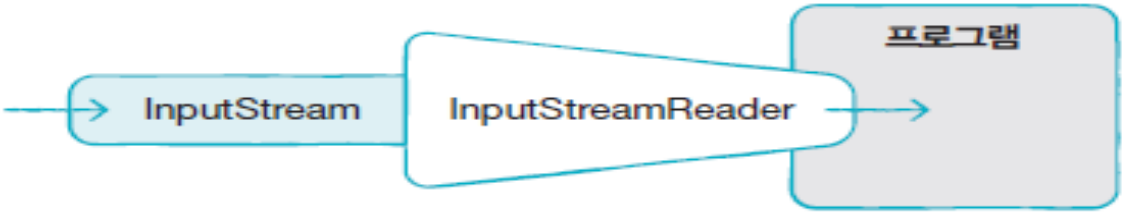
- 다른 스트림과 연결되어 여러가지 편리한 기능을 제공하는 스트림
- 자체적으로 입출력을 수행할 수 없음
- Input/OutputStream, Reader/Writer와 연결하여 입출력 수행

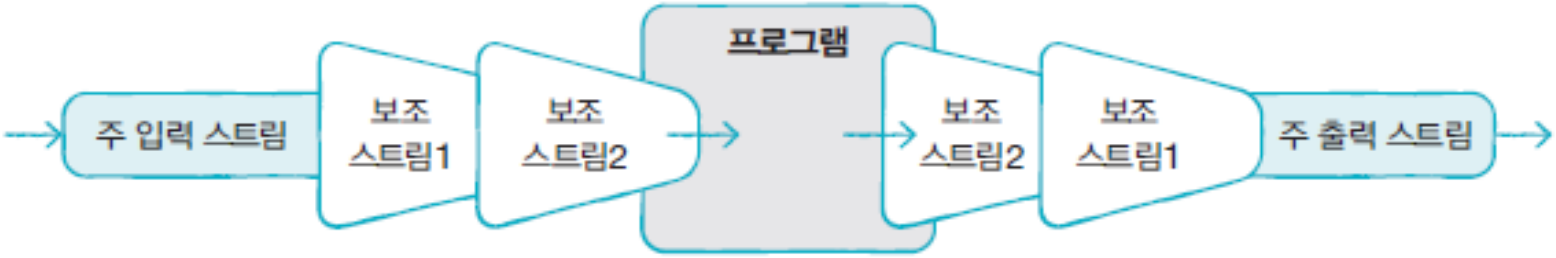


■ 보조 스트림

● 보조 스트림 연결

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

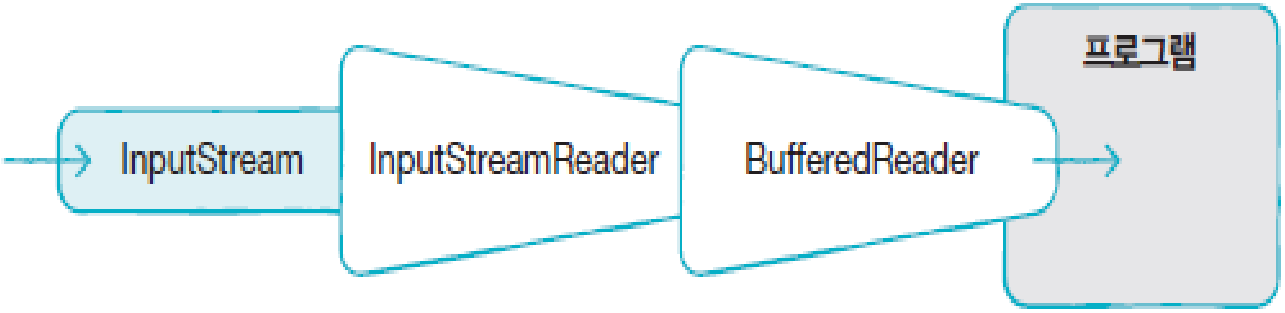




■ 보조 스트림

● 보조 스트림 연결

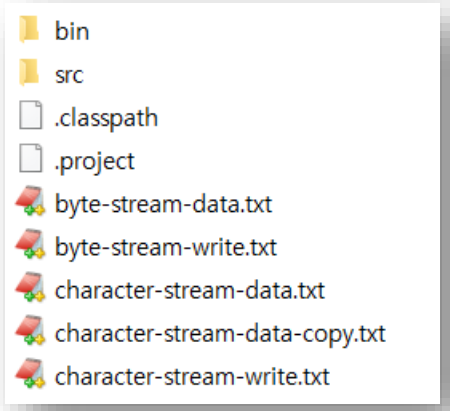
```
InputStream is = System.in;
InputStreamReader reader = new InputStreamReader(is);
BufferedReader br = new BufferedReader(reader);
```

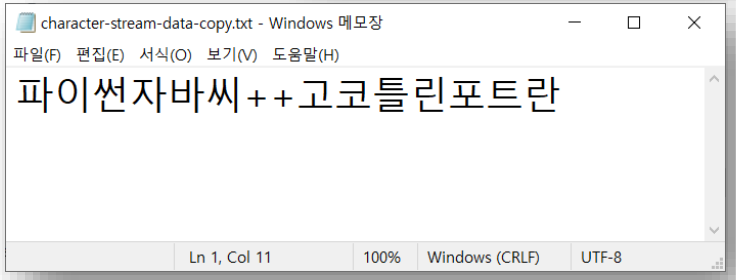


■ 일반적으로 많이 사용되는 파일 읽기/쓰기 코드

ch16.FileIO

```
InputStream in = new FileInputStream("character-stream-data.txt");
InputStreamReader isr = new InputStreamReader(in);
BufferedReader reader = new BufferedReader(isr);

OutputStream out = new FileOutputStream("character-stream-data-copy.txt");
OutputStreamWriter osw = new OutputStreamWriter(out);
BufferedWriter writer = new BufferedWriter(osw);

String data = "";
while (true) {
  data = reader.readLine();
  if(data == null) break;
  System.out.println(data);
  writer.write(data);
}

reader.close();
isr.close();
in.close();

writer.close();
osw.close();
out.close();
```





■ File 클래스

- java.io 패키지에서 제공
- 파일 및 폴더 정보 제공

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

- 파일이나 폴더가 존재하는지 확인하려면 exists() 메소드 호출

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

- exists() 메소드의 반환값이 false인 경우 다음 메소드로 파일 또는 폴더 생성

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

■ File 클래스

● exists() 메소드의 반환값이 true인 경우 다음 메소드로 파일 또는 폴더 제어

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.

■ File 클래스 사용

ch16.FileExam1

```
File dir = new File("text");

if(!dir.exists()) {
  dir.mkdirs();
}

File file1 = new File("text/file1.txt");
if(!file1.exists()) file1.createNewFile();

File file2 = new File("text/file2.txt");
if(!file2.exists()) file2.createNewFile();

File file3 = new File("text/file3.txt");
if(!file3.exists()) file3.createNewFile();

File text = new File("text");
File[] files = text.listFiles();
for(File f : files) {
  System.out.println(f.getName());
  System.out.println(f.getPath());
}
```

```
file1.txt
text\file1.txt
file2.txt
text\file2.txt
file3.txt
text\file3.txt
```

■ File 클래스 사용

ch16.FileExam2

```
File text = new File("text");
File[] files = text.listFiles();

System.out.println("시간\t\t\t형태\t\t크기\t이름");
System.out.println(
  "-------------------------------------------------------");

SimpleDateFormat format = new SimpleDateFormat("yyyy-MM-dd a HH:mm");

for(File f : files) {
  System.out.print(format.format(new Date(f.lastModified())));
  if(f.isDirectory()) {
    System.out.print("\t<DIR>\t\t\t" + f.getName());
  } else {
    System.out.print("\t\t\t" + f.length() + "\t" + f.getName());
  }
  System.out.println();
}
```

```
시간                          형태          크기        이름
-------------------------------------------------------
2022-06-09 오후 14:42                       0         file1.txt
2022-06-09 오후 14:42                       0         file2.txt
2022-06-09 오후 14:42                       0         file3.txt
```

■ 하위 경로 파일 탐색 - 1 (재귀호출)

ch16.RecursiveCall

```
public static void main(String[] args) {
  System.out.println("재귀호출 시작");
  int number = 5;
  long result = factorial(number);
  System.out.println("재귀호출 결과 : " + result);
  System.out.println("재귀호출 끝");
}

public static long factorial(int n) {
  long result = 0;
  if (n == 1) {
    result = 1;
  } else {
    result = n * factorial(n - 1);
  }
  return result;
}
```

```
재귀호출 시작
재귀호출 결과 : 120
재귀호출 끝
```

■ 하위 경로 파일 탐색 - 2 (재귀호출)

ch16.FileSearch

```
public static void main(String[] args) {
  FileSearch fs = new FileSearch();
  File dir = new File("");
  fs.subDirList(dir.getAbsolutePath());
}

public void subDirList(String source) {
  File dir = new File(source);
  File[] fileList = dir.listFiles();
  for (int i = 0; i < fileList.length; i++) {
    File file = fileList[i];
    if (file.isFile()) {
      System.out.println("\t 파일 이름 = " + file.getName());
    } else if (file.isDirectory()) {
      System.out.println("디렉토리 이름 = " + file.getName());
      subDirList(file.getAbsolutePath());
    }
  }
}
```

```
디렉토리 이름 = src
디렉토리 이름 = ch01
        파일 이름 = Comment.java
        파일 이름 = Print.java
디렉토리 이름 = ch02
        파일 이름 = Casting.java
        파일 이름 = Overflow.java
        파일 이름 = Variable1.java
        파일 이름 = Variable2.java
디렉토리 이름 = ch03
```

■ 하위 경로 및 모든 파일 삭제

ch16.FileDelete

```
public static void main(String[] args) {
  FileDelete dd = new FileDelete();

  File dir = new File("text");
  dd.deleteDirectory(dir.getAbsolutePath());
}

public boolean deleteDirectory(String path) {
  File file = new File(path);
  if (!file.exists()) {
    return false;
  }

  File[] files = file.listFiles();
  for (File f : files) {
    if (f.isDirectory()) {
      deleteDirectory(f.getAbsolutePath());
    } else {
      f.delete();
    }
  }
  return file.delete();
}
```