제25장
IO기반의 입출력-Part_1

1. IO 패키지 소개

❖ java.io 패키지

- 자바의 기본적인 데이터 입출력(IO: Input/Output) API 제공

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 InutStreamReader / OutputStreamWriter PrintWriter BufferedReader / BufferedWriter	문자 단위 입출력을 위한 하위 스트림 클래스

파일의 정보 : 이름, 경로, 파일의 크기, 속성(실행할수 있느냐? 읽을수 있느냐..)

바이트 단위 : 그림, 멀티미디어, 문자 등 다 읽고 쓰는 클래스

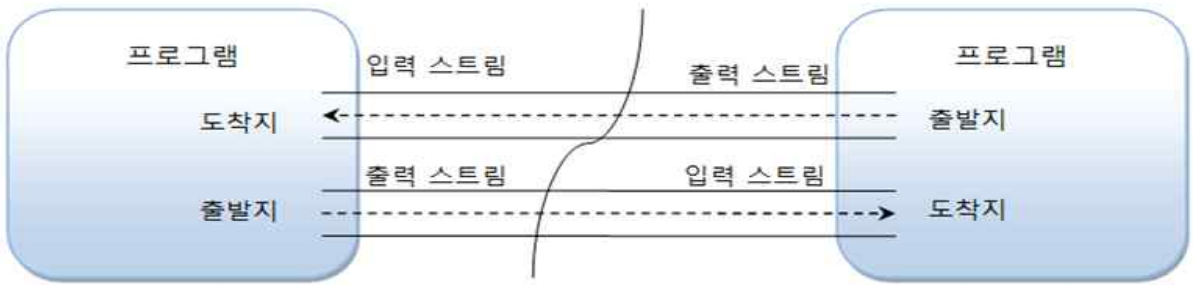
문자 단위 : 문자만 읽고 쓰는 특화된 클래스

2. 입력 스트림과 출력 스트림

❖ 입력 스트림과 출력 스트림의 개념

스트림 : 데이터의 흐름
입력 스트림 : 프로그램을 기준으로 하여 데이터가 들어오는 부분
출력 스트림 : 프로그램을 기준으로 하여 데이터가 나가는 부분

** 기준이 되는 것이 프로그램이 된다는 것을 주목하자.



위 그림은 네트워크 통신 프로그램을 나타낸 것이다.

2. 입력 스트림과 출력 스트림

❖ 바이트 기반 스트림과 문자 기반 스트림

- **바이트 기반 스트림(주고 받는 데이터가 바이트이다.)**
  - 그림, 멀티미디어, 문자 등 모든 종류의 데이터를 받고 보낼 수 있다.

- **문자 기반 스트림(주고 받는 데이터가 문자이다)**
  - 문자만 받고 보낼 수 있도록 특화되어 있다.

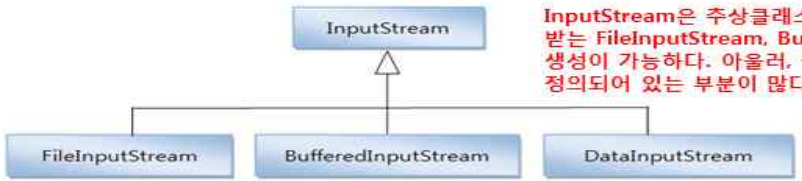
구분	바이트 기반 스트림		문자 기반 스트림	
	입력 스트림	출력 스트림	입력 스트림	출력 스트림
최상위 클래스	InputStream	OutputStream	Reader	Writer
하위 클래스 (예)	XXXputStream (FileInputStream)	XXXOutputStream (FileOutputStream)	XXXReader (FileReader)	XXXWriter (FileWriter)

접미사가 InputStream, OutputStream, Reader, Writer에 따라 어떤 기반의 입출력 스트림인지를 구분 지을 수가 있다.
예) 파일에서 바이트를 읽고 쓰고 싶다면 FileInputStream, FileOutPutStream을 사용할 수 가 있고, 파일로부터 문자를 읽고 싶다면, FileReader, FileWriter을 사용하면 될 것이다.

2. 입력 스트림과 출력 스트림

❖ InputStream

- **바이트 기반 입력 스트림의 최상위 클래스로 추상 클래스**



InputStream은 추상클래스로 객체 생성이 불가능하지만, 상속을 받는 FileInputStream, BufferedInputStream...등 으로 객체 생성이 가능하다. 아울러, 상속관계에 있기 때문에 리턴타입으로도 정의되어 있는 부분이 많다.

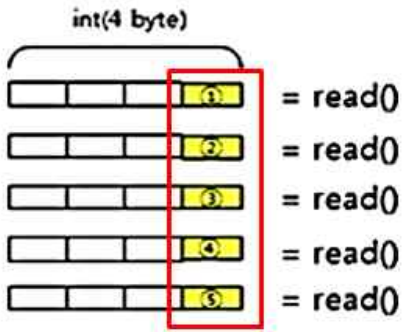
- **InputStream 클래스의 주요 메서드**

(모든 InputStream의 공통 메서드, 매우 중요하다.)

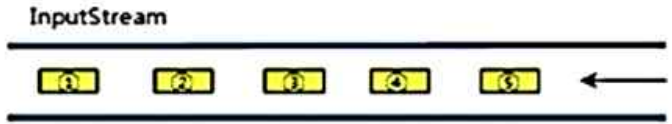
리턴타입	메소드	설명
int	read() 1바이트를 읽고 읽은 바이트를 Int(4바이트)끝 1바이트에 리턴함	입력 스트림으로부터 1 바이트를 읽고 읽은 바이트를 리턴한다.
int	read(byte[] b) 위에 read()와 완전 다르다. 읽은 내용은 b에 저장, 읽은 바이트 수를 리턴	입력 스트림으로부터 읽은 바이트들을 매개값으로 주어진 바이트 배열 b 에 저장하고 실제로 읽은 바이트 수를 리턴한다.
int	read(byte[] b, int off, int len) offset : 2 , length : 2 	입력 스트림으로부터 len 개의 바이트 만큼 읽고 매개값으로 주어진 바이트 배열 b[off] 부터 len 개까지 저장한다. 그리고 실제로 읽은 바이트 수인 len 개를 리턴한다. 만약 len 개를 모두 읽지 못하면 실제로 읽은 바이트 수를 리턴한다.
void	close()	사용한 시스템 자원을 반납하고 입력 스트림을 닫는다.

2. 입력 스트림과 출력 스트림

- **read() 메서드**



read()메서드는 한바이트만 읽고 그 바이트를 리턴하므로, 아래처럼 5개의 바이트가 들어온다면 read()는 5번 실행해야 하는 꼴이 된다.

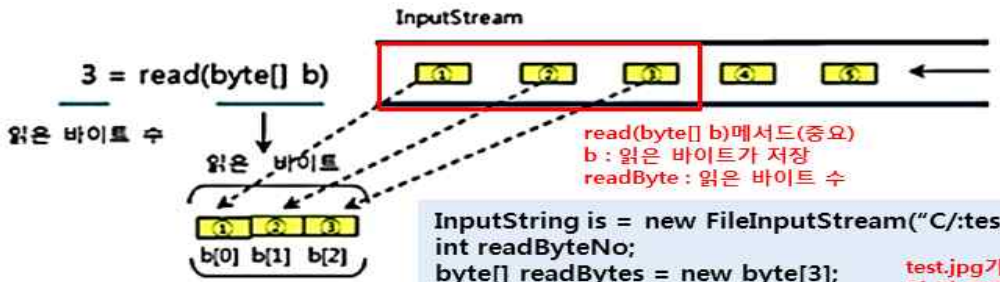



```
InputStream is = new FileInputStream( "C\:test.jpg" );
int readByte;

while( (readByte = is.read() ) != -1) {
            .........
}
```

** read()메서드는 더 이상 읽을 내용이 없다면 -1을 리턴한다(매우 중요하다)

2. 입력 스트림과 출력 스트림

- **read(byte[] b) 메서드**
  - 첫 번째 읽을 경우




```
InputString is = new FileInputStream("C/:test.jpg");
int readByteNo;
byte[] readBytes = new byte[3];

while( (readByteNo = is.read() ) != -1)
{
                .....
}
```

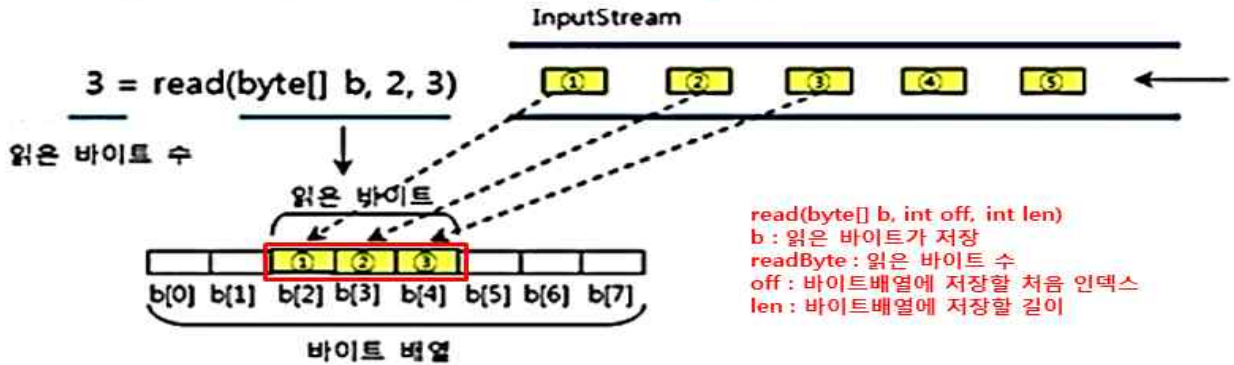
test.jpg가 5바이트라면?
몇 번 루프를 도는가?

더 이상 읽을게 없다면
-1을 리턴하는 것!

  - 두 번째 읽을 경우

2. 입력 스트림과 출력 스트림

- **read(byte[] b, int off, int len) 메서드**




```
InputStream is = ...;
byte[] readBytes = new byte[100];
Int readByteNo = is.read(readByteds);
```

```
InputStream is = ...;
byte[] readBytes = new byte[100];
Int readByteNo = is.read(readByteds,0,100);
```

위의 2개의 소스는 같은 동작을 한다.

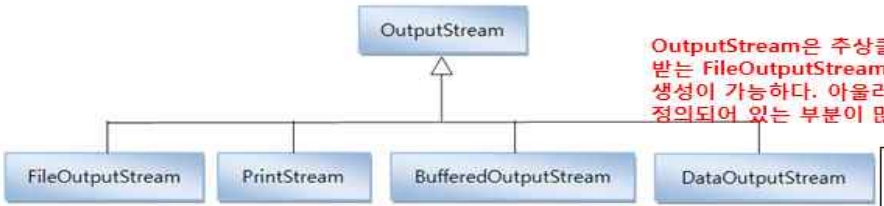
- **close()메서드**
  - InputStream을 더 이상 사용하지 않을 경우 호출해 준다.
  - InputStream에서 사용했던 시스템 자원을 풀어준다.(자원의 효율적 사용)

```
is.close();
```

2. 입력 스트림과 출력 스트림

❖ OutputStream

- **바이트 기반 출력 스트림의 최상위 클래스로 추상 클래스**



OutputStream은 추상클래스로 객체 생성이 불가능하지만, 상속을 받는 FileOutputStream, BufferedOutputStream...등 으로 객체 생성이 가능하다. 아울러, 상속관계에 있기 때문에 리턴타입으로도 정의되어 있는 부분이 많다.

write()메서드를 사용하게 되면, 바로바로 출력이 되는 것이 아니라, 메모리 버퍼에 쌓였다가 다 쌓이면 비로소 출력하게 된다. 만약에 버퍼에 데이터가 다 쌓이지 않게되면 출력이 안된다. 이때 강제로 버퍼를 비우게 만들고 출력하게 만들어주는 역할을 하는 메서드가 바로 flush()이다. 매우 중요하다 아울러 write()를 하면 항상 flush()를 하는것이 습관이 되어야 한다.

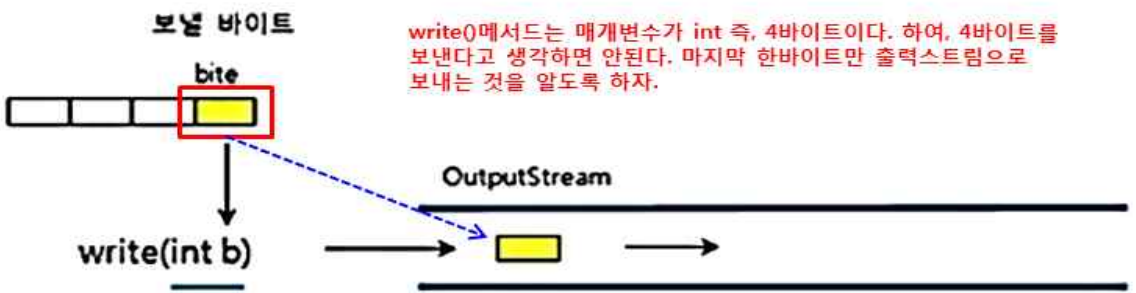
- **OutputStream의 주요 메서드**

(모든 OutputStream의 공통 메서드, 매우 중요하다.)

리턴타입	메소드	설명
void	wrtie(int b)	출력 스트림으로 1 바이트를 보낸다.
void	write(byte[] b)	출력 스트림에 매개값으로 주어진 바이트 배열 b 의 모든 바이트를 보낸다.
void	write(byte[] b, int off, int len)	출력 스트림에 매개값으로 주어진 바이트 배열 b[off] 부터 len 개까지의 바이트를 보낸다.
void	flush()	버퍼에 잔류하는 모든 바이트를 출력한다.
void	close()	사용한 시스템 자원을 반납하고 출력 스트림을 닫는다.

2. 입력 스트림과 출력 스트림

- write(int b) 메서드



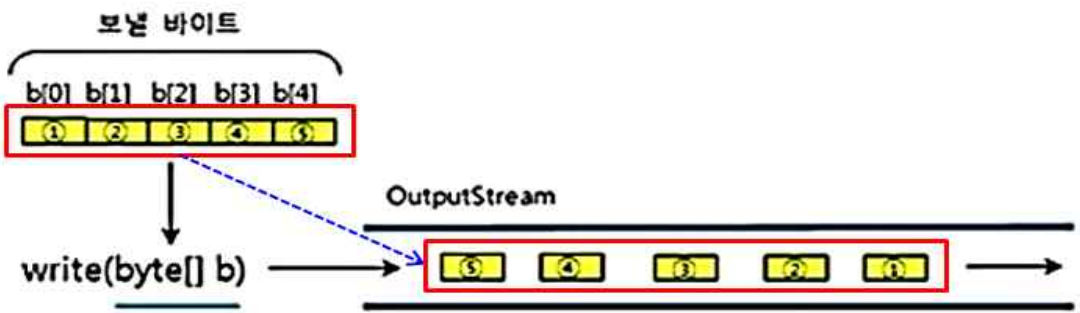
write()메서드는 매개변수가 int 즉, 4바이트이다. 하여, 4바이트를 보낸다고 생각하면 안된다. 마지막 한바이트만 출력스트림으로 보내는 것을 알도록 하자.

```
OutputStream os = new FileOutputStream("C/:test.txt");
byte[] data = "ABC".getBytes();

for(int i=0; i<data.length; i++) {
        os.write(data[i]);   // "A","B","C"를 하나씩 출력
}
```

위 소스는 test.txt파일에 ABC를 한자씩 출력한다.
곧, 루핑을 3번 한다.

2. 입력 스트림과 출력 스트림

- write(byte[] b) 메서드



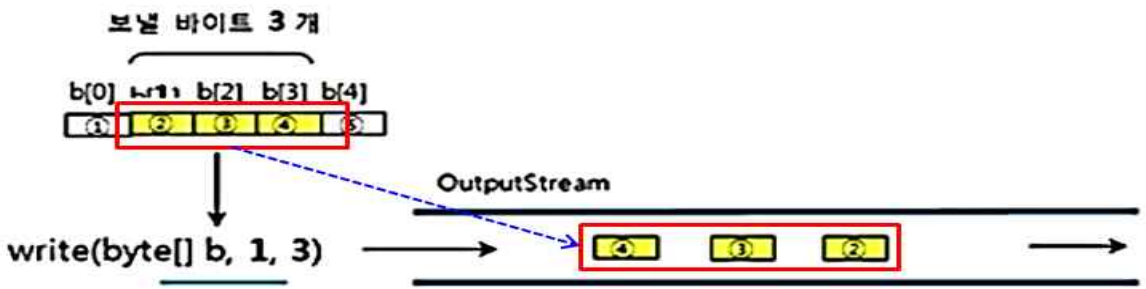

```
OutputStream os = new FileOutputStream("C/:test.txt");
byte[] data = "ABC".getBytes();

os.write(data);   // "ABC"를 모두 한번에 출력
```

위 소스는 test.txt파일에 ABC를 한번에 출력한다.
곧, 1번만에 출력한다.(루핑이 필요없다)

앞서 본 write(int b)보다 훨씬 효율적이다.

2. 입력 스트림과 출력 스트림

- write(byte[] b, int off, int len) 메서드




```
OutputStream os = new FileOutputStream("C/:test.txt");
byte[] data = "ABC".getBytes();

os.write(data, 1, 2);   // "BC"만 출력
```

offset : 1 , length : 2

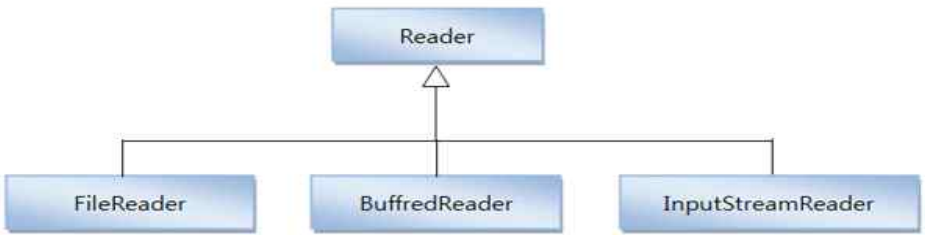
- flush()와 close()메서드

```
OutputStream os = new FileOutputStream("C/:test.txt");
byte[] data = "ABC".getBytes();
os.write(data);
os.flush();     // write()를 하고 나면 무조건 flush()를 호출하도록 하자.
os.close();
```

2. 입력 스트림과 출력 스트림

❖ Reader

- **문자 기반 입력 스트림의 최상위 클래스로 추상 클래스 (문자만 특화되어 읽을수 있다.)**



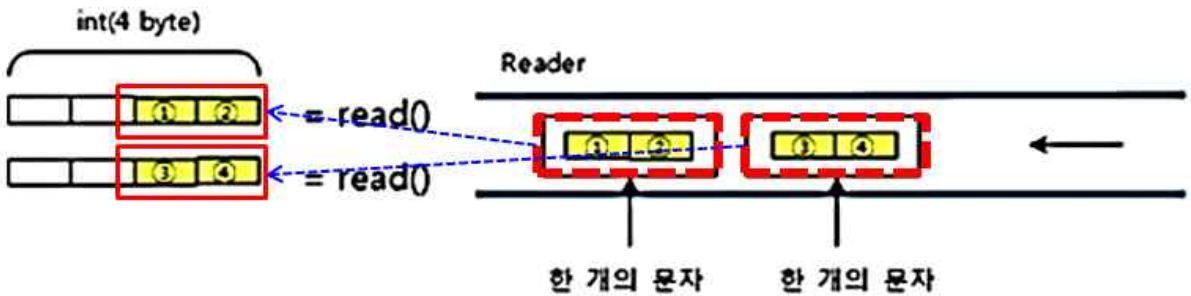
- **Reader의 주요 메서드**

한 문자는 2바이트이기 때문에, int 타입의 끝 2바이트만 읽고 리턴.

메소드		설명
int	read()	입력 스트림으로부터 한개의 문자를 읽고 리턴한다.
int	read(char[] cbuf) InputSteam과의 차이는 byte이냐 char이냐의 차이다.	입력 스트림으로부터 읽은 문자들을 매개값으로 주어진 문자 배열 cbuf 에 저장하고 실제로 읽은 문자 수를 리턴한다.
int	read(char[] cbuf, int off, int len)	입력 스트림으로부터 len 개의 문자를 읽고 매개값으로 주어진 문자 배열 cbuf[off] 부터 len 개까지 저장한다. 그리고 실제로 읽은 문자 수인 len 개를 리턴한다.
void	close()	사용한 시스템 자원을 반납하고 입력 스트림을 닫는다.

2. 입력 스트림과 출력 스트림

- read() 메서드

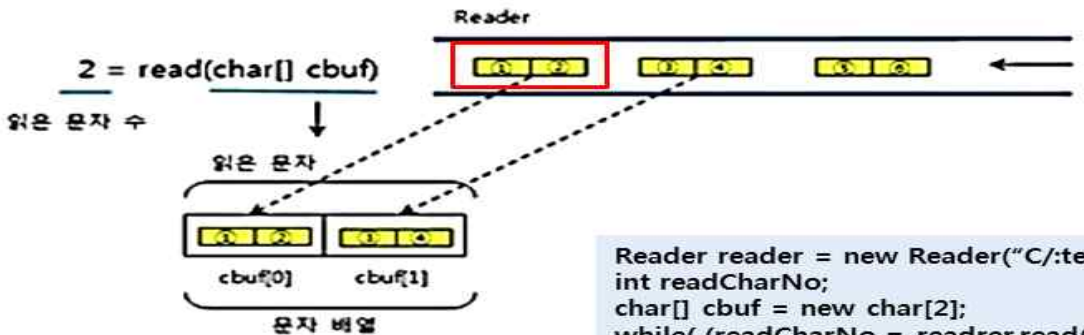


한 문자는 2바이트이기 때문에, int타입의 끝 2바이트만 읽고 리턴.

```
Reader reader = new FileReader("C/:test.txt");
int readData;

while( (readData = readrer.read() ) != -1)  //더 이상 읽을 문자가 없다면 -1을 리턴
{
        char charData = (char)readData;  읽은 문자를 char타입으로 강제타입 변환하여 출력해보면 된다.
}
```

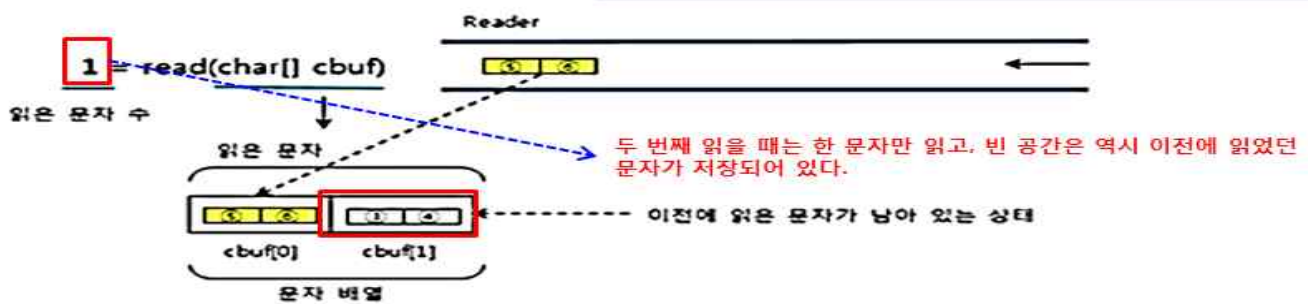
2. 입력 스트림과 출력 스트림

- read(char[] cbuf) 메서드
  - 첫 번째 읽을 경우



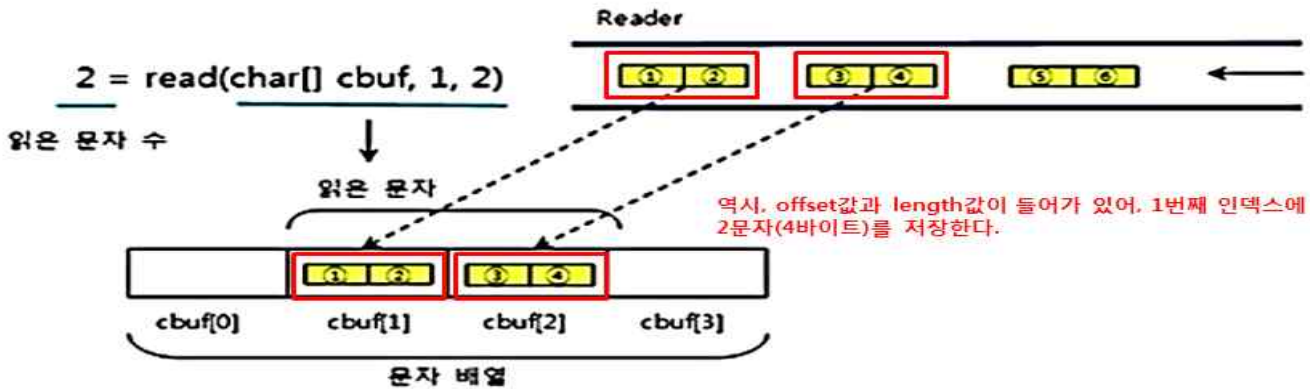

```
Reader reader = new Reader("C/:test.txt");
int readCharNo;
char[] cbuf = new char[2];
while( (readCharNo = readrer.read(cbuf) ) != -1) {
                    .....
}
```

  - 두 번째 읽을 경우



2. 입력 스트림과 출력 스트림

- read(char[] cbuf, int off, int len) 메서드




```
Reader reader = ...;
char[] cbuf = new cbuf[100];
Int readCharNo = reader.read(cbuf);
```

```
Reader reader = ...;
char[] cbuf = new cbuf[100];
Int readCharNo = reader.read(cbuf, 0, 100);
```

위의 2코드는 똑같은 코드이다.

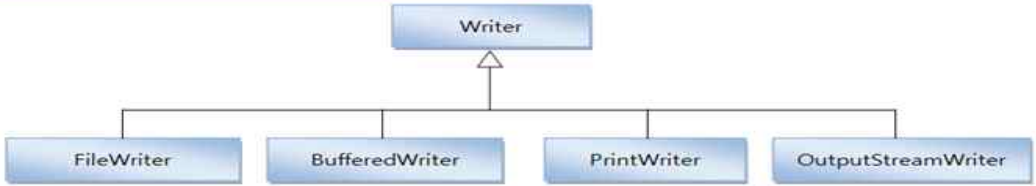
- close()메서드
  - Reader를 더 이상 사용하지 않을 경우 호출해 준다.
  - Reader에서 사용했던 시스템 자원을 풀어준다.

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

2. 입력 스트림과 출력 스트림

❖ Writer

- **문자 기반 출력 스트림의 최상위 클래스로 추상 클래스 (문자만 특화되어 출력할 수 있다.)**



- **Writer의 주요 메소드** 한 문자는 2바이트이기 때문에, int 타입의 끝 2바이트만 스트림으로 출력한다.

리턴타입	메소드	설명
void	wrtie(int c)	출력 스트림으로 매개값으로 주어진 한 문자를 보낸다.
void	write(char[] cbuf)	출력 스트림에 매개값으로 주어진 문자 배열 cbuf 의 모든 문자를 보낸다.
void	write(char[] cbuf, int off, int len)	출력 스트림에 매개값으로 주어진 문자 배열 cbuf[off] 부터 len 개까지의 문자를 보낸다.
void	write(String str)	출력 스트림에 매개값으로 주어진 문자열을 전부 보낸다.
void	write(String str, int off, int len)	출력 스트림에 매개값으로 주어진 문자열 off 순번부터 len 개까지의 문자를 보낸다.
void	flush()	버퍼에 잔류하는 모든 문자열을 출력한다.
void	close()	사용한 시스템 자원을 반납하고 출력 스트림을 닫는다.

2. 입력 스트림과 출력 스트림

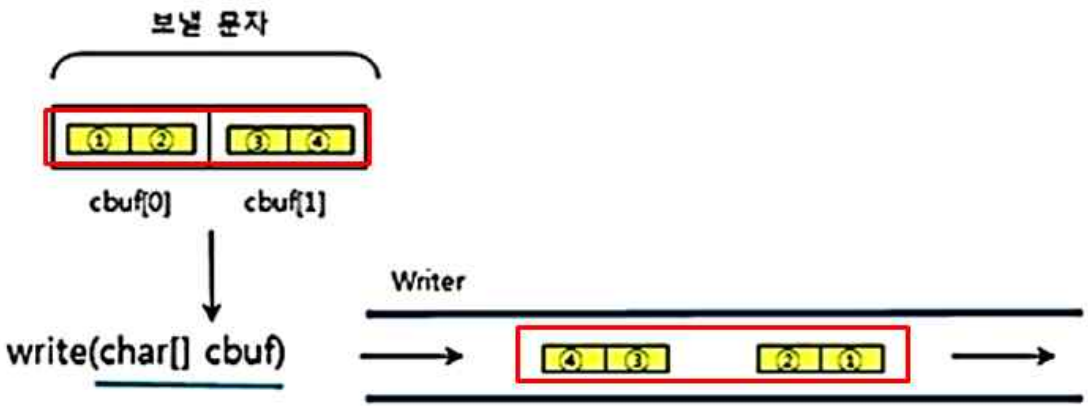
- write(int c) 메서드


```
Writer wrier = new FileWriter("C/:test.txt");  //다형성
char[] data = "홍길동".toCharArray();

for(int i=0; i<data.length; i++) {
        writer.write(data[i]);   // "홍","길","동"를 하나씩 출력
}
```

한 문자는 2바이트이기 때문에, int 타입의 끝 2바이트만 스트림으로 출력하므로.
이 루핑은 곧 3번이 이루어진다.
아울러, String에서 char[]을 얻기 위해선 .toCharArray()를 이용하는 것을 잊지
않도록 한다.

2. 입력 스트림과 출력 스트림

- write(char[] cbuf) 메서드

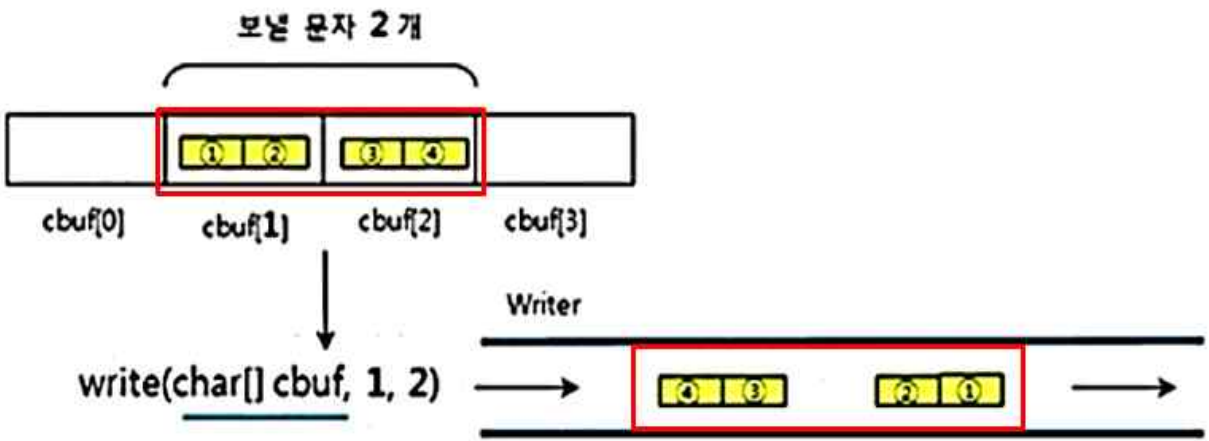



```
Writer wrier = new FileWriter("C₩:test.txt");
char[] data = "홍길동".toCharArray();
Writer.write(data);  //"홍길동"을 모두 출력
```

이전, 코드보다 문자배열자체를 매개값으로 보내기 때문에, 훨씬 효율적이다.

2. 입력 스트림과 출력 스트림

- write(char[] cbuf, int off, int len) 메서드

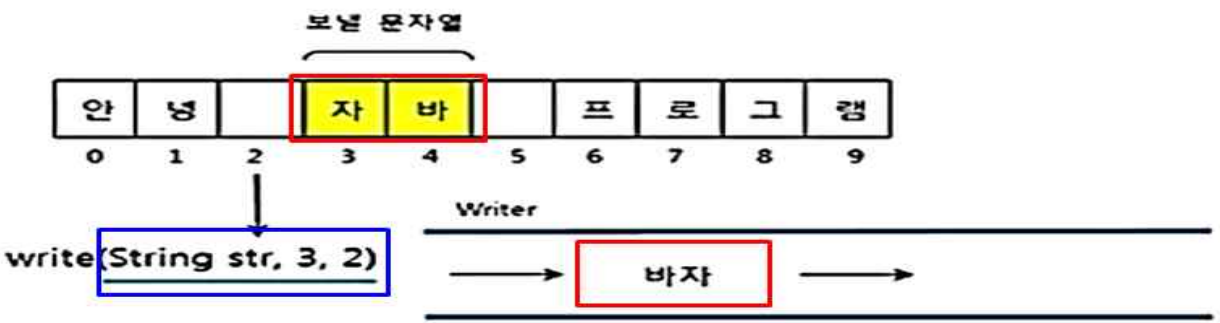



```
Writer wrier = new FileWriter("C/:test.txt");
char[] data = "홍길동".toCharArray();
Writer.write(data, 1, 2);  //"길동"만 출력
```

2. 입력 스트림과 출력 스트림

- write(String str)와 wirte(String str, int off, int len)메서드

write()의 매개타입이 문자열이다.



2. 입력 스트림과 출력 스트림

- flush()와 close()메서드

```
Writer writer = new FileWriter("C₩:test.txt");

String data = "안녕 자바 프로그램";

writer.write(data);

writer.flush()  //다시 한번 얘기하지만 write()를 쓰면 꼭 flush()를 호출하는 습관을 들이자.

writer.close()  //아울러 자원해제도 꼭 해주도록 하자.
```

감사합니다.