제23장
IO기반의 입출력-Part_2

3. 파일 입출력

❖ File 클래스

- 파일 시스템의 파일을 표현하는 클래스
  - 파일 크기, 파일 속성, 파일 이름 등의 정보 제공, 파일 생성 및 삭제 기능 제공한다.
  - 디렉토리 생성, 디렉토리에 존재하는 파일 리스트 얻어내는 기능 제공, 또한 디렉토리 삭제기능도 포함하고 있다.
- 파일 객체 생성

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

생성자의 매개값으로 파일의 경로, 혹은 디렉토리의 경로를 주고 파일객체를 얻으면 된다.(경로 구분자는 ₩₩ 혹은 /를 사용하면 된다.)
자바에서는 ₩는 escape문자를 나타내므로 ₩₩ 두번 적어줘야 된다.

- 파일 또는 디렉토리 존재 유무 확인 메서드

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

파일이나 디렉토리가 실제로 존재하는지 확인한다. 그 이유는 File클래스는 생성자를 통해서 만들면 파일 혹은 디렉토리가 있건 없건 같에 만들어지기 때문이다.

- 파일 및 디렉토리 생성 및 삭제 메서드

리턴타입	메소드	설명
boolean	createNewFile()	새로운 파일을 생성
booelan	mkdir()	새로운 디렉토리를 생성
boolean	mkdirs()	경로상에 없는 모든 디렉토리를 생성
boolean	delete()	파일 또는 디렉토리 삭제

설정한 경로에 디렉토리가 없다면 생성자의 매개값으로 준 디렉토리를 다 만들어주는 것이다.
예) C:/make/mymedia 를 다 만들어 준다.

3. 파일 입출력

- **파일 및 디렉토리의 정보를 리턴하는 메소드**

C:₩₩Temp₩₩test.txt
부모디렉토리 / 파일 이름

리턴타입	메소드	설명
boolean	canExecute()	실행할 수 있는 파일인지 여부
boolean	canRead()	읽을 수 있는 파일인지 여부
boolean	canWrite()	수정 및 저장할 수 있는 파일인지 여부
String	getName()	파일의 이름을 리턴
String	getParent()	부모 디렉토리를 리턴
File	getParentFile()	부모 디렉토리를 File 객체로 생성후 리턴
String	getPath()	전체 경로를 리턴
boolean	isDirectory()	디렉토리인지 여부
boolean	isFile()	파일인지 여부
boolean	isHidden()	숨김 파일인지 여부
long	lastModified()	마지막 수정 날짜 및 시간을 리턴
long	length()	파일의 크기 리턴
String[]	list()	디렉토리에 포함된 파일 및 서브디렉토리 목록 전부를 String 배열로 리턴
String[]	list(FilenameFilter filter) 필터링 해준다.	디렉토리에 포함된 파일 및 서브디렉토리 목록 중에 FilenameFilter 에 맞는 것만 String 배열로 리턴
File[]	listFiles()	디렉토리에 포함된 파일 및 서브 디렉토리 목록 전부를 File 배열로 리턴
File[]	listFiles(FilenameFilter filter)	디렉토리에 포함된 파일 및 서브디렉토리 목록 중에 FilenameFilter 에 맞는 것만 File 배열로 리턴

설정한 경로가 있다면 해당 파일의 이름만 얻는다.
전체경로를 얻는게 아니다.

단위가 바이트이다.

리턴타입만 다를 뿐 기능은 똑같다.

3. 파일 입출력

❖ FileInputStream

- 파일로부터 바이트 단위로 읽어 들일 때 사용
  - 그림, 오디오, 비디오, 텍스트 파일 등 모든 종류의 파일을 읽을 수 있다.

- 객체 생성 방법

```
//첫 번째 방법
FileInputStream fis = new FileInputStream("C:/image.gif");

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

  - FileInputStream 객체가 생성될 때 파일과 직접 연결
  - 만약 파일이 존재하지 않으면 FileNotFoundException 발생
  - try-catch문으로 예외 처리를 해야 한다.

- InputStream 하위 클래스이기 때문에, 사용 방법이 InputStream과 동일

3. 파일 입출력

❖ FileOutputStream

- 파일에 **바이트 단위**로 데이터를 저장할 때 사용
  - 그림, 오디오, 비디오, 텍스트 등 모든 종류의 데이터를 파일로 저장

- 객체 생성 방법

```
//첫 번째 방법
FileOutputStream fos = new FileOutputStream("C:/image.gif");

//두 번째 방법
File file = new File("C:/image.gif");
FileOutputStream fos = new FileOutputStream(file);
```

  - 파일이 이미 존재할 경우, 데이터를 출력하게 되면 파일을 덮어쓰는 단점이 존재한다.
  - 기존 파일 내용 끝에 데이터를 추가할 경우에는 생성자에 true매개값을 주면 된다.(중요)

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

- OutputStream 하위 클래스이기 때문에 사용 방법이 OutputStream과 동일하다.

3. 파일 입출력

❖ FileReader

- 텍스트 파일로부터 데이터를 읽어 들일 때 사용
  - 문자 단위로 읽기 때문에, 텍스트가 아닌 그림, 오디오, 비디오 등의 파일은 읽을 수 없다.
- 객체 생성 방법

```
//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);
```

  - FileReader 객체가 생성될 때 파일과 직접 연결
  - 만약 파일이 존재하지 않으면 FileNotFoundException 발생
  - try-catch문으로 예외 처리를 해야 한다.
- Reader 하위 클래스이므로 사용 방법 Reader와 동일하다.

3. 파일 입출력

❖ FileWriter

- 텍스트 파일에 문자 데이터를 저장할 때 사용
  - 텍스트가 아닌 그림, 오디오, 비디오 등의 데이터를 파일로 저장할 수 없다.

- 객체 생성 방법

```
//첫 번째 방법
FileWriter fw = new FileWriter("C:/file.txt");

//두 번째 방법
File file = new File("C:/file.txt");
FileWriter fw = new FileWriter(file);
```

  - 파일이 이미 존재할 경우, 데이터를 출력하게 되면 파일을 덮어쓰는 단점이 존재한다.
  - 기존 파일 내용 끝에 데이터를 추가할 경우에는 생성자에 true매개값을 주면 된다.(중요)

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

- Writer 하위 클래스이므로 사용 방법이 Writer와 동일하다.

4. 보조 스트림

❖ 보조 스트림

- 다른 스트림과 연결이 되어 여러가지 편리한 기능을 제공해주는 스트림
  - 문자 변환, 입출력 성능 향상, 기본 데이터 타입 입출력, 객체 입출력 등의 기능을 제공

예) 입력스트림으로 문자가 들어온다면 InputStream으로 받는것 보다 보조스트림 즉 InputStreamReader를 연결해서 사용하는 것이 훨씬 효율적일 것이다.

- 보조 스트림 생성

```
보조스트림 변수 = new 보조스트림(연결스트림)
```

```
InputStream is = System.in;  //InputStream이 주입력스트림이 된다.(키보드로입력)
InputStreamReader reader = new InputStreamReader(is);  //보조스트림(문자)을 연결함
```

- 보조 스트림 체인 – 다른 보조 스트림과 연결되어 역할 수행

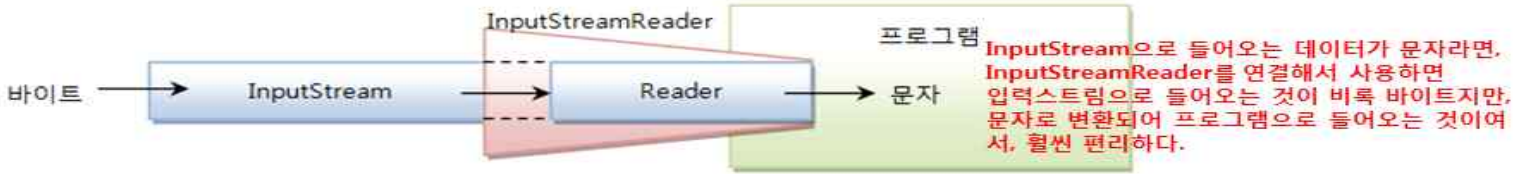
예) 주입력스트림으로 들어오는 것이 문자라면 문자스트림을 보조스트림으로 연결하고, 속도 및 성능 향상을 위해서 스트림을 연결할 수 있다.(대용량데이터)

4. 보조 스트림

❖ 문자 변환 보조 스트림

- 소스 스트림이 바이트 기반 스트림이지만 데이터가 문자일 경우 사용함.
  - Reader와 Writer는 문자 단위로 입,출력하기 때문에 바이트 기반 스트림 보다 편리하다.
  - 문자셋의 종류를 지정할 수 있기 때문에 다양한 문자를 입출력 가능하다.

- InputStreamReader(InputStream을 Reader로 변환한다)




```
InputStream is = System.in;  //콘솔로부터 얻는 InputStream이라면, 문자만 입력받는다.
                             (강제캐스팅이나 문자배열로 바꿔야 하는 작업이 생략될 수 있다.)
Reader reader = new InputStreamReader(is);  // InputStreamReader를 연결함.(보조스트림)
```

```
FileInputStream fis = new FileInputStream("C:/file.txt");  //위와 동일하다. 차이는 파일인 것이다.
Reader reader = new InputStreamReader(fis);
```

4. 보조 스트림

- **OutputStreamWriter**

프로그램에서 문자를 출력스트림으로 보낼때, OutputStream으로 보낼려면, 바이트 기반으로 보내야하므로 바이트배열로 변환하는 등의 작업이 필요하지만, OutputStreamWriter를 보조스트림으로 연결해주면 그러한 작업이 생략 가능하다.

```
//문자파일에 저장하는 FileOutputStream을 생성했지만 바이트기반이다. 하여 아래와 같이
//InputStreamWriter을 보조스트림으로 연결해주면 된다.
FileOutputStream fos = new FileOutputStream("C:/file.txt");
Writer writer = new OutputStreamWriter(fos);
```

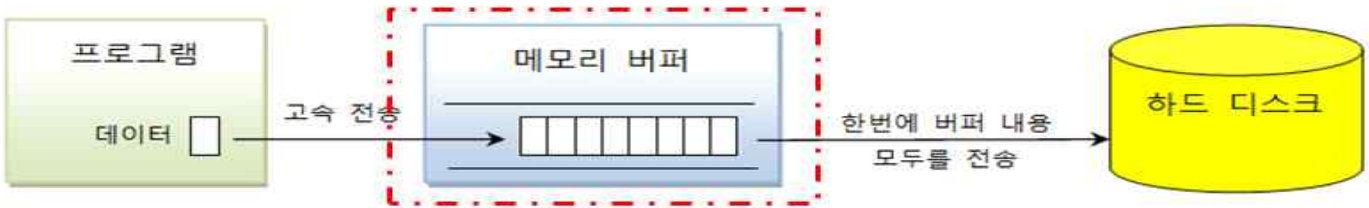
4. 보조 스트림

❖ **성능 향상 보조 스트림**

- 입,출력 성능에 영향을 미치는 입출력 소스
  - 하드 디스크 : 속도가 무지하게 느리다.(ex. 낮은 r(revolutions)pm수를 가진 HDD)
  - 느린 네트워크(ex. 10Mbps)

* 프로그램이 아무리 빠른 성능을 지니고 있다고 해서, 환경이 안좋으면 안좋은 쪽으로 맞춰지기 때문에 최대한 성능이 빠른 버퍼와 작업하는 것이 좋다.

- 버퍼를 이용한 해결
  - 입출력 소스와 직접 작업하지 않고 버퍼(buffer)와 작업하므로, 실행 성능 향상



  - 프로그램은 쓰기 속도 향상
  - 버퍼 차게 되면 데이터를 한꺼번에 하드 디스크로 보내 출력 횟수를 줄여줌

4. 보조 스트림

- **BufferedInputStream, BufferedReader**


```
BufferedInputStream bis = new BufferedInputStream (바이트입력스트림);
BufferedReader br = new BufferedReader(문자입력스트림);
```

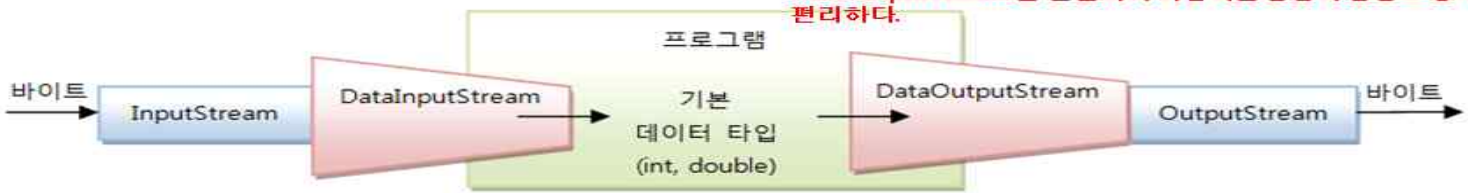
- **BufferedOutputStream과 BufferedWriter**


```
BufferedOutputStream bos = new BufferedOutputStream (바이트출력스트림);
BufferedWriter bw = new BufferedWriter(문자출력스트림);
```

4. 보조 스트림

❖ 기본 타입 입출력 보조 스트림

* 기본 타입을 입출력할때, 1바이트만 가지고 입출력을 하면 시간도 오래 걸리고 성능면에서 좋지가 않다. 하여, DataInputStream, DataOutputStream을 연결해서 사용하면 상당히 성능도 좋아지고 편리하다.



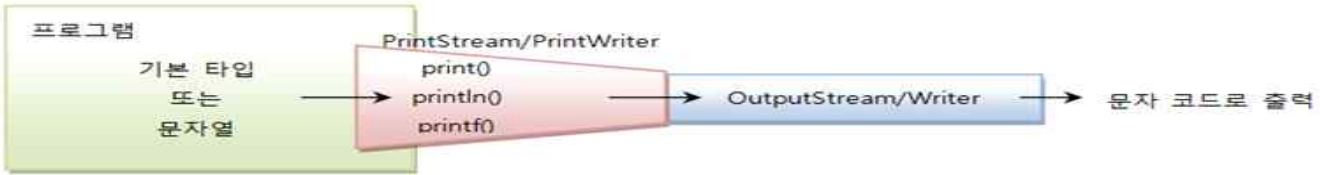

```
DataInputStream dis = new DataInputStream (바이트입력스트림);
DataOutputStream dos = new DataOutputStream(바이트출력스트림);
```

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)

4. 보조 스트림

❖ 프린터 보조 스트림

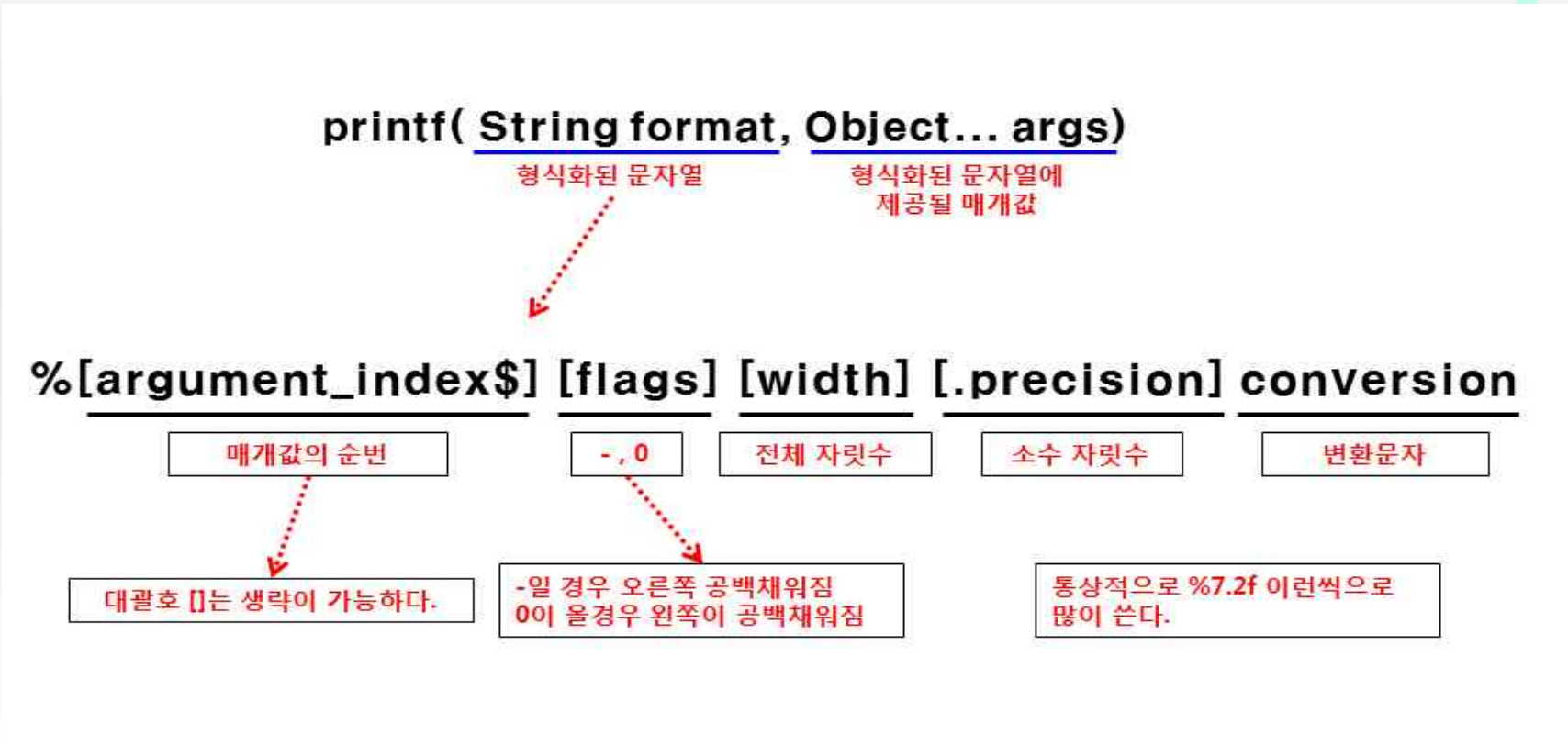
* 기본타입이나 문자열을 출력할때, 개행을 해야 할 경우가 빈번하다면 PrintStream을 출력스트림에 연결해서 사용하는 것이 편리하다.




```
PrintStream ps = new PrintStream (바이트출력스트림);
PrintWriter pw = new PrintWriter (문자출력스트림);
```

PrintStream / PrintWriter	* 사용 메서드는 동일하다.		
void	print(boolean b)	void	println(boolean b)
void	print(char c)	void	println(char c)
void	print(double d)	void	println(double d)
void	print(float f)	void	println(float f)
void	print(int i)	void	println(int i)
void	print(long l)	void	println(long l)
void	print(Object obj)	void	println(Object obj)
void	print(String s)	void	println(String s)
		void	println()

4. 보조 스트림


printf(String format, Object... args)
형식화된 문자열
형식화된 문자열에
제공될 매개값
%[argument_index$] [flags] [width] [.precision] conversion
매개값의 순번
- , 0
전체 자릿수
소수 자릿수
변환문자
대괄호 []는 생략이 가능하다.
-일 경우 오른쪽 공백채워짐
0이 올경우 왼쪽이 공백채워짐
통상적으로 %7.2f 이런씩으로
많이 쓴다.

5. 콘솔 입출력

❖ 콘솔(Console)

- 시스템을 사용하기 위해 키보드로 입력을 받고 화면으로 출력하는 소프트웨어를 칭함.

- Unix, Linux: 터미널
- Windows 운영체제: 명령 프롬프트
- 이클립스: Console 뷰

1. 입력 스트림 : 명령프롬프트를 사용해 입력을 받기 위해 입력스트림 즉, System.in을 제공함
2. 출력 스트림 : 프로그램에서 데이터를 콘솔에 출력하기 위한 출력스트림 즉, System.out를 제공함
   예) System.out.println()..
3. System.err스트림은 주로 프로그램에서 err를 출력할 때, 사용하는 스트림.

5. 콘솔 입출력

- System.in 필드
  - InputStream 타입의 입력 스트림이며, InputStream 변수를 대입할 수 있다.(바이트 기반)

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

  - 읽은 byte는 키보드의 아스키 코드(ascii code)이다.

```
int asciiCode = is.read();
```

  - 아스키 코드로부터 문자 읽기

```
char inputChar = (char) is.read();  //읽은 아스키코드를 문자로 보기 위해 강제타입변환
```

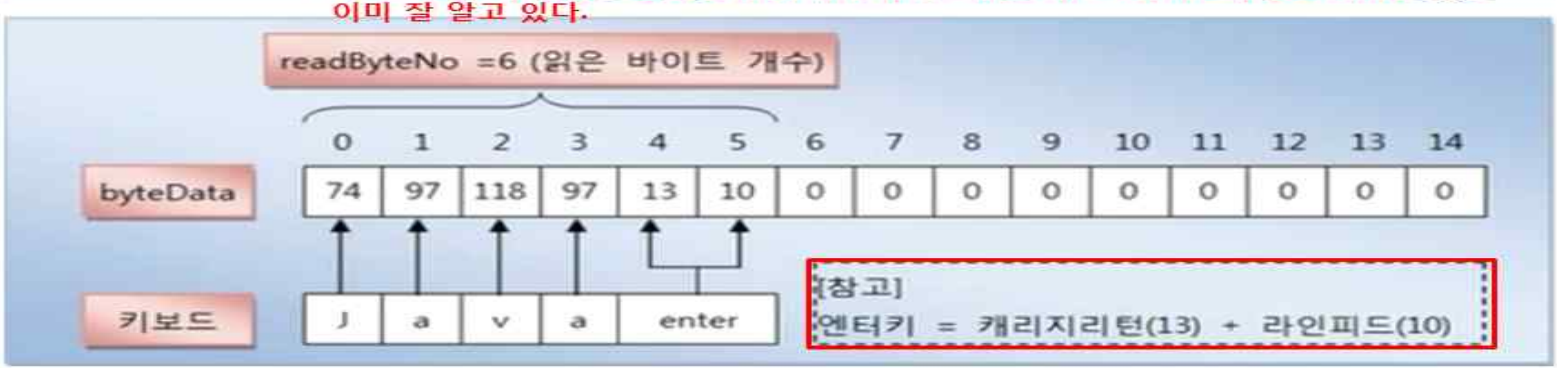
5. 콘솔 입출력

- 키보드로부터 입력된 한글 읽기

```
byte[] byteData = new byte[15];
int readByteNo = System.in.read(byteData);
```

읽은 바이트수 — 실제로 읽은 바이트가 저장됨

* 키보드로부터 한글을 입력받기 위해서 바이트배열 최소 2바이트가 필요하다는 것은 이미 잘 알고 있다.




```
String strData = new String( byteData, 0, readByteNo-2 );
```

바이트 배열 — 읽은 바이트 - 2

시작 인덱스

위의 예제는 0~3인덱스까지만 문자열로 생성하는 것이다.

5. 콘솔 입출력

❖ System.out 필드

- PrintStream 타입의 출력 스트림이므로, OutputStream으로 타입 변환할 수 있다.

* PrintStream은 OutputStream의 자손클래스임을 알고 있다.

```
OutputStream os = System.out;
```

- 아스키 코드를 출력하면 콘솔에는 문자가 출력된다.

```
byte b = 97;
os.write(b);
os.flush()
```

- 문자열을 출력하려면 바이트 배열을 얻어야 한다.

```
String name = "홍길동";
byte[] nameBytes = name.getBytes();
os.write(nameBytes);
os.flush();
```

```
PrintStream ps = System.out;
ps.println(...);
```

줄여서 ----------> `System.out.println(...);`

* System.out이 PrintStream이였고, 그의 멤버메서드인 println을 우측과 같이 줄여서 썼던 것이다.

5. 콘솔 입출력

❖ Console 클래스

- 자바6부터 콘솔에서 입력된 **문자열을 쉽게 읽을 수 있도록 제공**

```
Console console = System.console();
```

- 이클립스에서 System.console()은 null 리턴하기 때문에 명령 프롬프트에서 반드시 실행해야 한다.(중요)→ NullPointerException을 발생함.

- **Console 클래스의 읽기 메소드**

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

* readPassword()메서드는 패스워드이므로 콘솔에 보여주지않는 것이 보안상 좋다.(입력시 보이지 않는다)

5. 콘솔 입출력

- **Scanner 클래스**
  - **Console 클래스의 단점**
    - **문자열은 읽을 수 있지만 기본 타입(정수, 실수) 값을 바로 읽을 수 없다.**
  - **java.util.Scanner**
    - **콘솔로부터 기본 타입의 값을 바로 읽을 수 있다.**

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

    - **제공하는 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.

감사합니다.