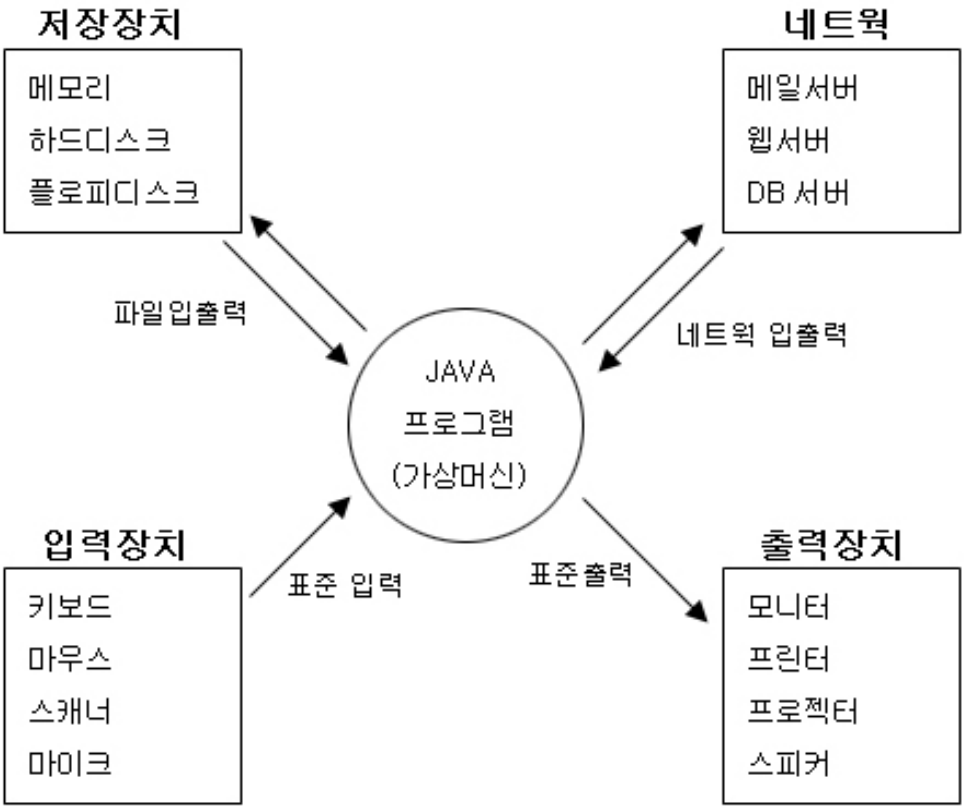
데이터의 입출력


저장장치
메모리
하드디스크
플로피디스크
네트웍
메일서버
웹서버
DB 서버
파일입출력
네트웍 입출력
JAVA
프로그램
(가상머신)
입력장치
키보드
마우스
스캐너
마이크
표준 입력
표준출력
출력장치
모니터
프린터
프로젝터
스피커

Java 의 I/O 프로그래밍

입출력 프로그래밍이란?

- 프로그램에서 필요한 데이터를 입력 장치로부터 입력 받는 것
- 프로그램 수행 결과(데이터)를 출력 장치로 출력하는 것

입출력 프로그램은

컴퓨터와 연결된 다양한 형태의 입출력 디바이스들과 연관

→

시스템 의존적인 작업

디바이스에 따라 입출력 기능을 구현하는 방법이 다름

스트림(stream)이란? (단방향)

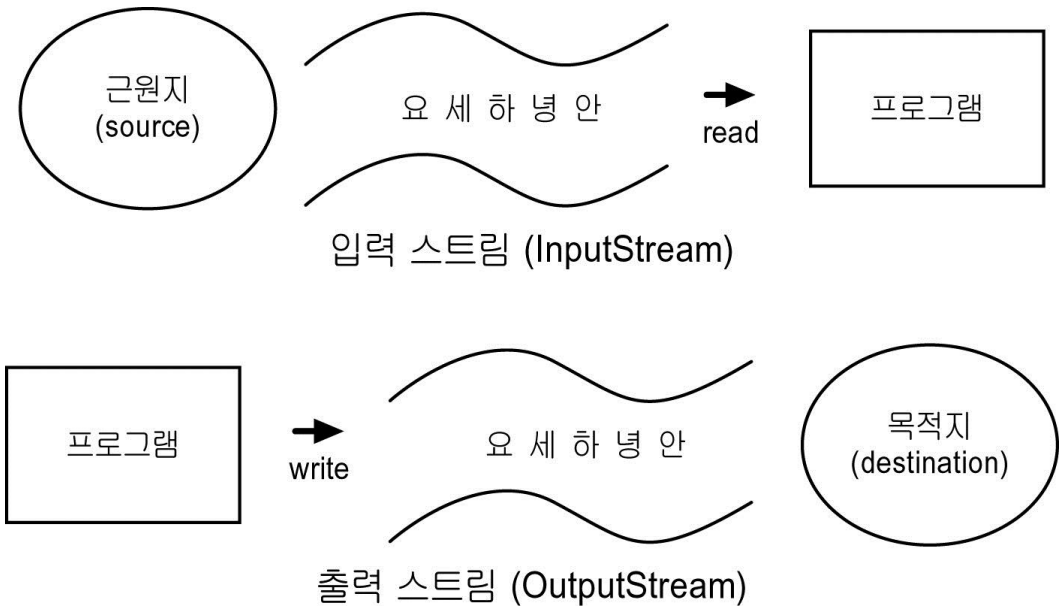
정해진 포맷을 사용하여 문자 또는 바이트 형식으로 전송 되는 연속적인 흐름. (순서가 있으며 길이가 정해져 있지 않다.)

스트림 입출력을 통해서 메모리, 파일, 네트워크 등 다양한 경로로의 입출력을 구현

스트림의 종류

근원지에서 흘러 들어오는 데이터 - 입력 스트림

목적지로 흘러가는 데이터 - 출력 스트림



입출력 단위

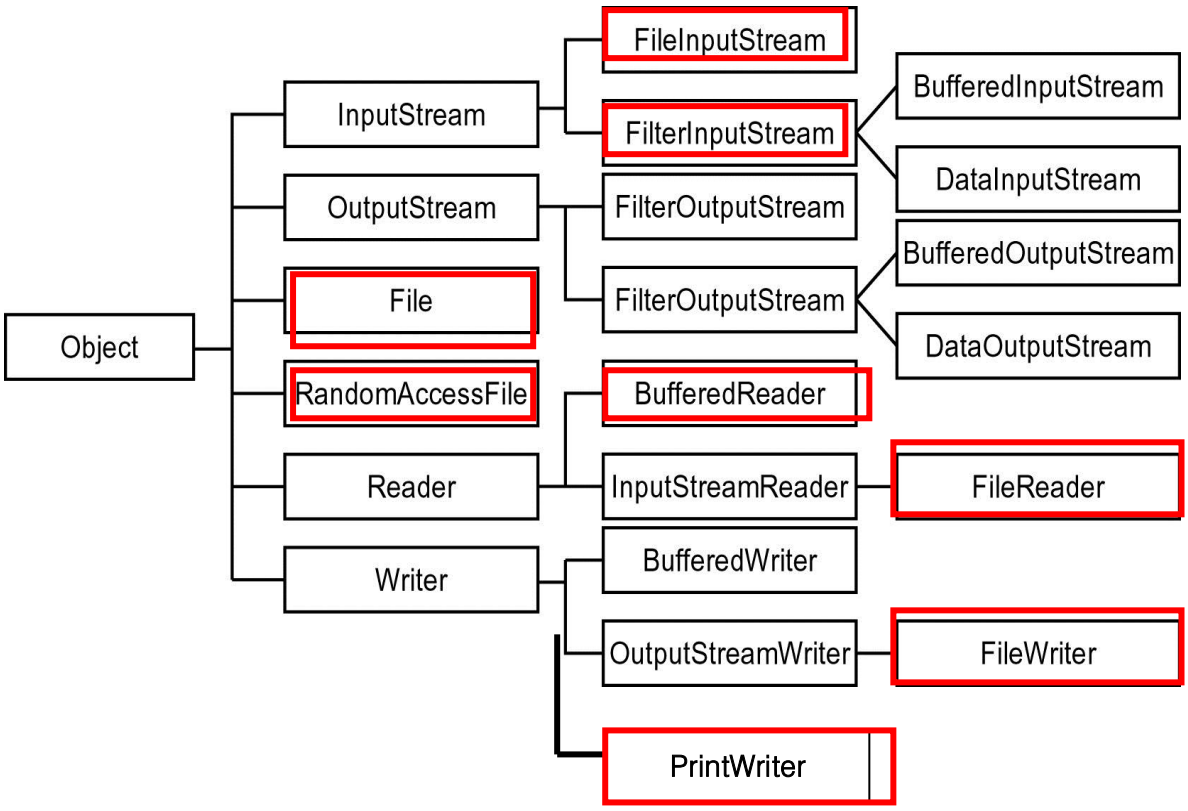
- 바이트 단위 : 바이트 스트림 → 8 비트의 바이트 코드
- 문자 단위 : 문자 스트림 → 16비트의 문자나 문자열

스트림 입출력의 특징

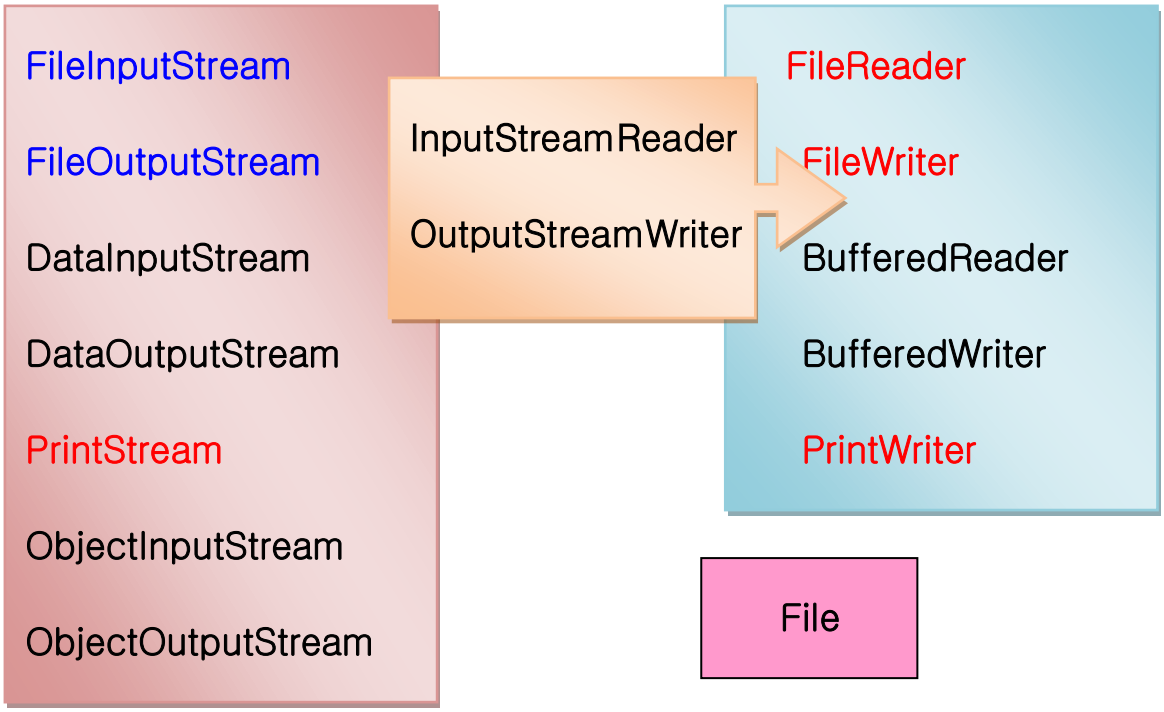
- 순차적인 데이터의 흐름, 데이터의 무작위적인 접근 불가능
- 단방향 흐름
  - 입력 장치로부터 입력되는 데이터를 의미하는 입력 스트림
  - 출력 장치에서 출력되는 데이터를 의미는 출력 스트림
- 모든 데이터의 입출력이 정해진 포맷으로 전송
- 데이터의 근원지나 목적지가 파일이든 네트워크이든 구분할 필요 없이 항상 같은 방법으로 프로그램 작성

java.io 패키지

- 입출력을 위해 스트림을 생성하고 다루는 클래스들
- 입출력 방향과 데이터의 단위 그리고 데이터 타입의 종류 등 여러 조건에 따라 사용할 수 있는 다양한 종류의 클래스들
- 파일이나 폴더를 다루는 데 이용되는 File 클래스
- 파일에 특정 위치의 접근이 가능하게 하는 RandomAccessFile



Java 의 I/O 프로그래밍



File 클래스

- 기존의 파일이나 폴더에 대한 다양한 기능을 처리하는데 사용
- 파일과 폴더의 접근 권한, 마지막 수정 일자, 길이, 폴더에 존재하는 파일 이름, 파일과 폴더의 삭제, 이름 변경 등 파일에 대해 직접 입 출력하는 기능을 제외한 파일과 관련된 다양한 기능을 지원하는 클래스

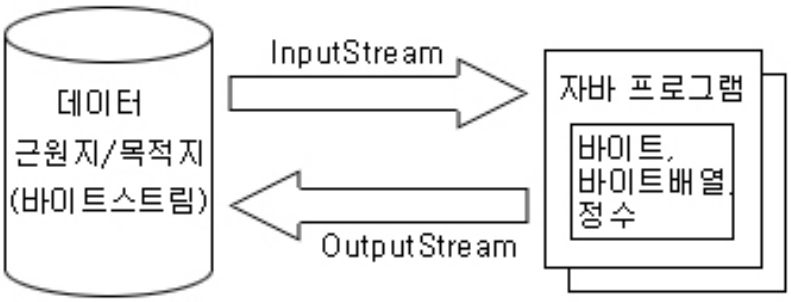
```
boolean delete()
boolean exists()
String getParent()
String getName()
long length()
long lastModified()
boolean isDirectory()
boolean isFile()
boolean mkdir()
boolean renameTo(File dest)
```

바이트 스트림 API

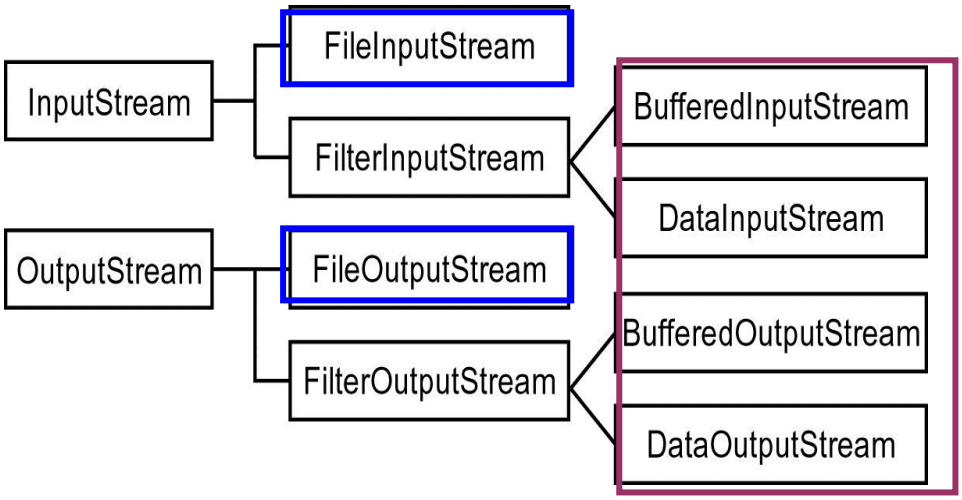
- 1바이트 단위로 데이터를 읽고 쓰기 위한 스트림

- InputStream/OutputStream의 하위 클래스

- 이미지 파일 등 바이너리 파일을 읽고 쓸 때



바이트 스트림 클래스는 모두 InputStream/OutputStream의 하위 클래스이고 포함되는 클래스와 그 계층 관계는 아래와 같다.



입출력 장치를 대상으로 직접 입출력

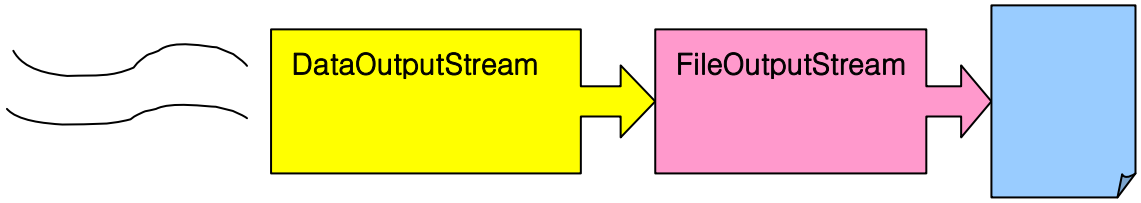
FileInputStream, FileOutputStream

다른 스트림 객체를 대상으로 입출력

DataInputStream, DataOutputStream

2개 이상의 스트림 객체를 연결한 입출력

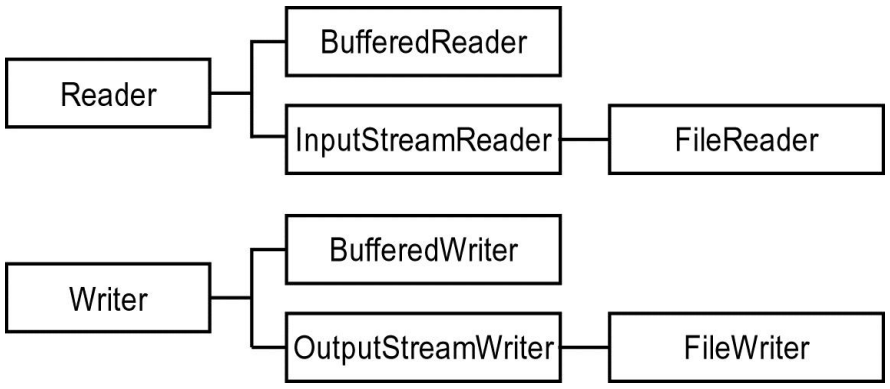
```
try {
        FileOutputStream fos = new FileOutputStream("test.out");
        DataOutputStream dos = new DataOutputStream(fos);
        dos.writeDouble(123.456);
        dos.writeInt(55);
        dos.writeChar('K');
        dos.close();
        fos.close();
}catch(IOException e) { }
```



문자 스트림 개요

- 16비트 문자나 문자열들을 읽고 쓰기 위한 스트림
- Reader/Writer의 하위 클래스
- 영어 이외의 문자에 대한 처리와 문자 인코딩을 내부에서 처리
- 유니코드를 지원하는 Java 특성에 맞게 2바이트 크기 입출력

Reader/Writer 클래스에서부터 시작되는 문자 스트림 클래스들의 상속관계를 나타내는 그림



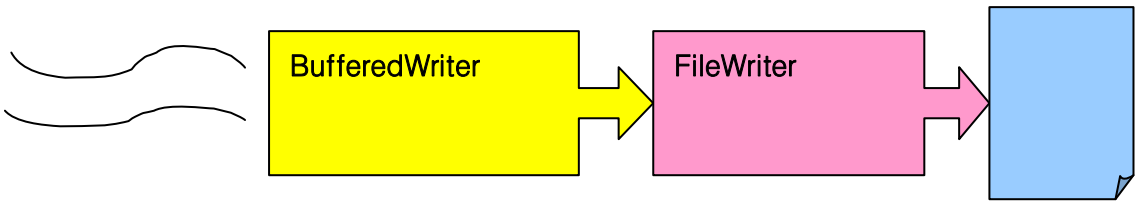
입출력 장치를 대상으로 직접 입출력

FileReader, FileWriter

다른 스트림 객체를 대상으로 입출력

BufferedReader, BufferedWriter

```
try{
        FileWriter fw = new FileWriter("test.txt");
        BufferedWriter bw = new BufferedWriter(fw);
        bw.write("abc"); bw.newLine();
        bw.write("가마다"); bw.newLine();
        bw.close();
        fw.close();
}catch(IOException e) { }
```



InputStreamReader/OutputStreamWriter

- 바이트 스트림에서 문자 스트림으로의 변환을 제공하는 입출력 스트림 API
- 바이트를 단위로 읽어서 지정된 문자 인코딩에 따라 문자로 변환하는데 사용(System.in 을 가지고 문자단위 입력 시 사용)
- 문자로 변환하는 경우 인코딩 방식은 특정 방식으로 지정할 수도 있고 경우에 따라서는 플랫폼의 기본 인코딩을 이용

표준 입출력

스크린과 키보드를 통한 입출력 방법

java.lang.System의 멤버 변수인 in, out, err을 이용

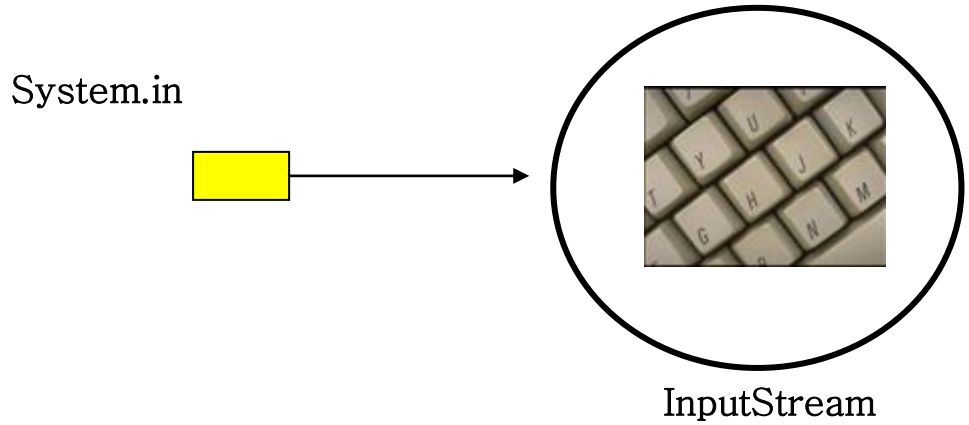
표준 입력 : System.in 은 InputStream 객체 변수

```
int input = System.in.read();
```

표준 출력 : System.out 은 PrintStream 객체 변수

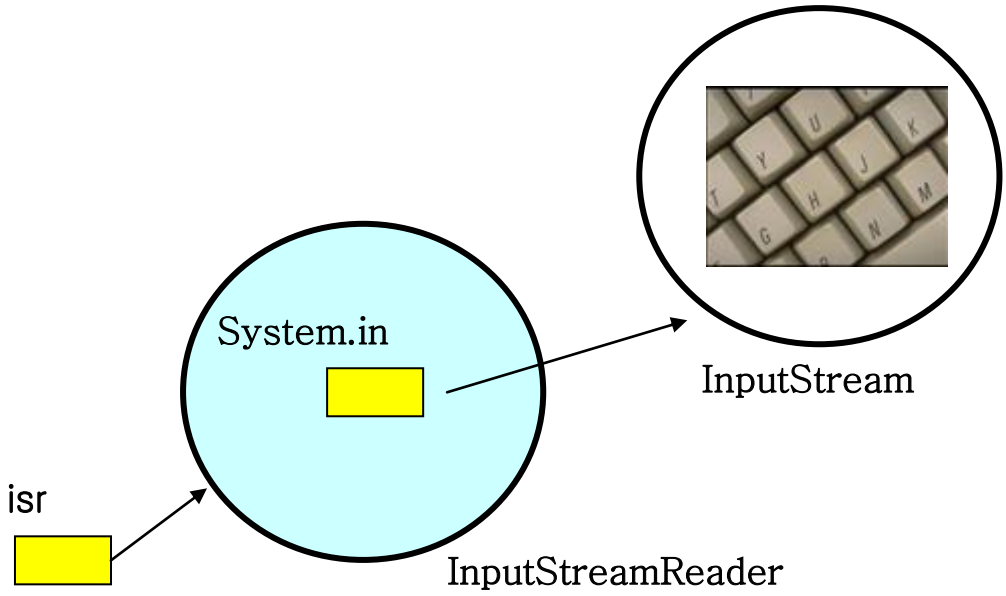
```
System.out.print(……);

System.out.println(……);
```

[키보드로부터 한 바이트 입력]



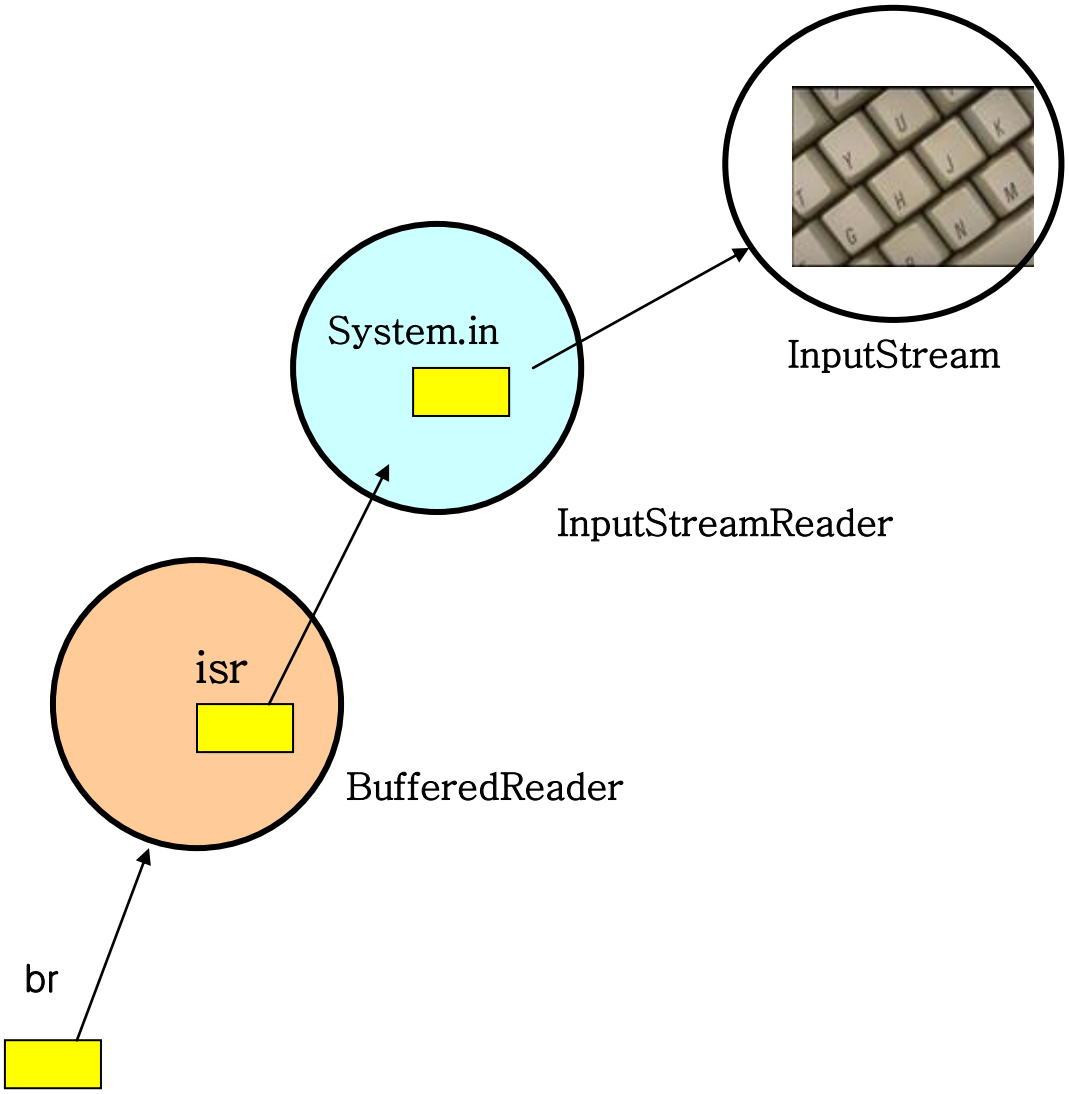

```
char input = (char)System.in.read();
```

[키보드로부터 한 문자 입력]




```
InputStreamReader isr = new InputStreamReader(System.in);

char input = (char)isr.read();
```

[키보드로부터 한 행 입력]




```
InputStreamReader isr = new InputStreamReader(System.in);

BufferedReader br = new BufferedReader(isr);

String input = br.readLine();
```

```
// 키보드로부터 한 바이트만 읽는다.
int inputbyte = System.in.read();

// 키보드로부터 한 문자만 읽는다.
InputStreamReader isr = new InputStreamReader(System.in);
int inputchar = isr.read();

// 키보드로부터 한 행을 읽는다.
InputStreamReader isr = new InputStreamReader(System.in);
BufferedReader br = new BufferedReader(isr);
String inputline = br.readLine();
```

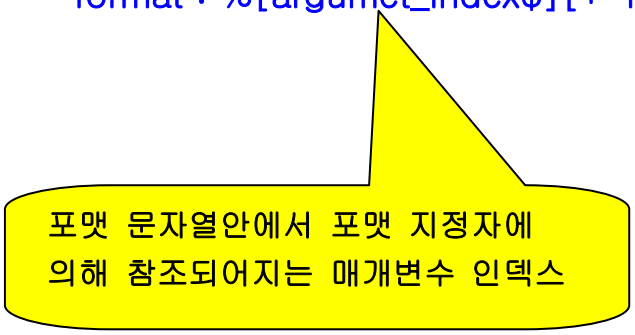
포매팅 I/O (JDK 5.0)

- JDK 5.0 부터 지원
- PrintStream 의 printf() 메서드 사용
- %.. 으로 출력 포맷 지정
- Java2 에서 문자열 결합 연산자(Concatenation) + 를 이용하던 불편을 다소 해결

문법 구조

```
printf(format, arguments)
```

format : %[argumet_index$][+-flag][width]conversion



```
int a = 20;

long b = 20000L;

float c = 34.56F;

double d = 123.123;

char e = 'A';

Calendar today = Calendar.getInstance();


System.out.println("1 : "+a +" "+b+" "+c+" "+d+" "+ e);
```

→ JAVA2 방식

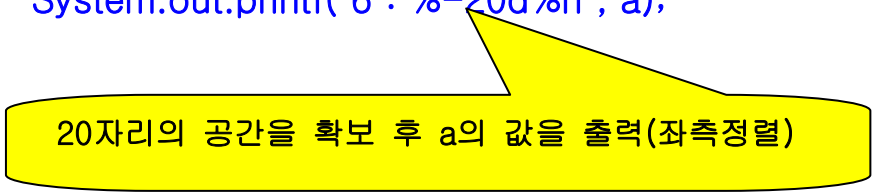
```
System.out.printf("2 : %d %d %f %f %c %n", a, b, c, d, e);

System.out.printf("3 : %f %1$a %1$e %1$f %1$g ₩n",

                  32.3453453453);
```

아규먼트 갯수가 포맷지정자와 차이가 날 경우 위치 지정

```
System.out.printf("4 : %1$h %1$d %2$f %3$c ₩n", a, d, e);

System.out.printf("5 : %1$20d%n", a);
```

20자리의 공간을 확보 후 a의 값을 출력(우측정렬)

```
System.out.printf("6 : %-20d%n", a);
```




```
System.out.printf("7 : %1$20.10f%n", 2345.0123456789);

System.out.printf("8 : 오늘의
    날짜는 %1$tY 년 %1$tm 월 %1$td 일 입니다.₩n", today);
```

%형식		입력	출력	%형식		입력	출력
문자	%c	‘A’	A(char)	시간요일	%tm	Calendar 타입	01~12(월)
정수	%c	65	A(char)		%tl, %tI		1~12, 01~12(시간)
	%d	65	65(10진수)		%tK, %tH		0~23, 00~23(시간)
	%h, %x	65	41(16진수)		%tM		00~59(분)
	%o	65	101(8진수)		%tS		00~60(초)
실수	%f	65.65	65.650000		%tb, %th		January, February…(1월, …)
	%a	65.65	0x1.069999999ap6		%tB		Jan, Feb,..(1월, …)
	%e	65.65	6.565000e+01		%tY, %ty		2006, 06(년도)
	%g	65.65	65.6500		%te, %td		1~31, 01~31(일)
시간요일	%tA	Calendar 타입	Sunday(일요일)		%tT		“%tH:%tM:%tS”
	%ta		Sun(일)		%tD		“%tm/%td/%ty”
	%tj		001~366(일)		%tF		“%tY-%tm-%td”
	%tR		“%tH:%tM”		%tc		“%ta %tb %td %tT %tZ %tY

Scanner 를 이용한 텍스트 스캔하기

java.util.Scanner 클래스를 이용함으로써 일반 표현문을 사용하는 스트링과 기본형 타입을 읽고 파싱(parsing)하는 좀 더 쉬워짐

JDK 5.0 이전의 소스

```
FileReader reader = new FileReader("xxx");
BufferedReader in = new BufferedReader(reader);
String string;
while ((string = in.readLine()) != null) {
    System.out.println(string);
}
```

Scanner 를 이용하여 재구현

```
Scanner scanner = new Scanner(new File(fileName));
scanner.useDelimiter(System.getProperty("line.separator"));
while (scanner.hasNext()) {
    System.out.println(scanner.next());
}
```

표준 입력으로부터 정수형 숫자 읽어 들이기

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

int i = sc.nextInt();
```

분리자를 이용하여 구분하여 읽기

```
String input = "1 fish 2 fish red fish blue fish";

Scanner s = new Scanner(input).useDelimiter("₩₩s*fish₩₩s*");

System.out.println(s.nextInt());

System.out.println(s.nextInt());

System.out.println(s.next());

System.out.println(s.next());

s.close();
```

Scanner 의 활용 예

[data.txt 파일의 내용]
100,100,100
200,200,200
300,300,300
400,400,400
500,500,500

```
Scanner sc = new Scanner(new File("data.txt"));
int cnt = 0;
int totalSum = 0;
while (sc.hasNextLine()) {
   String line = sc.nextLine();
   Scanner sc2 = new Scanner(line).useDelimiter(",");
   int sum = 0;
   while(sc2.hasNextInt()) {
       sum += sc2.nextInt();
   }
   System.out.println(line + ", sum = "+ sum);
   totalSum += sum;
   cnt++;
}
System.out.println("Line: " + cnt + ", Total: " + totalSum);
```