JAVA 기본 입력과 출력

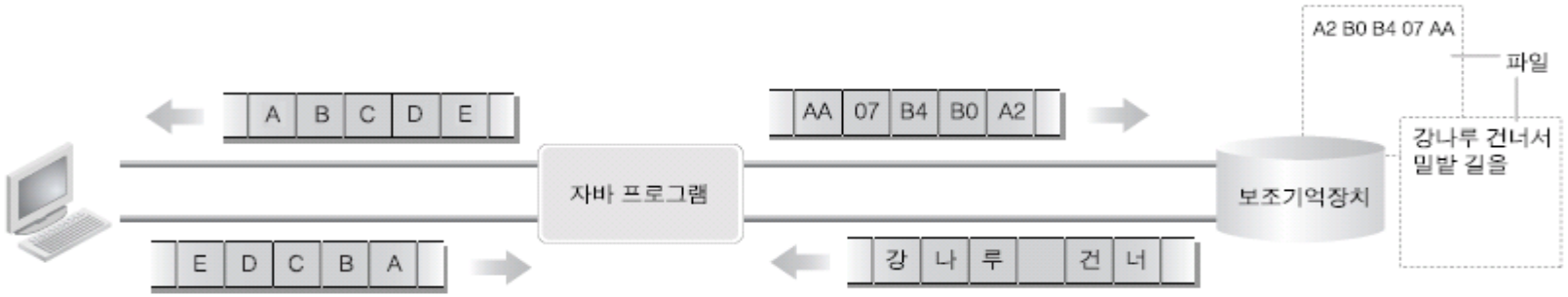
JAVA Programming

강 성 관

목 차

- System 클래스와 기본 출력 처리
- 입력과 출력 시 예외 처리
- Keyboard를 통한 기본 입력 처리

- 스트림이란?
  - 일차원적인 데이터의 연속적인 흐름



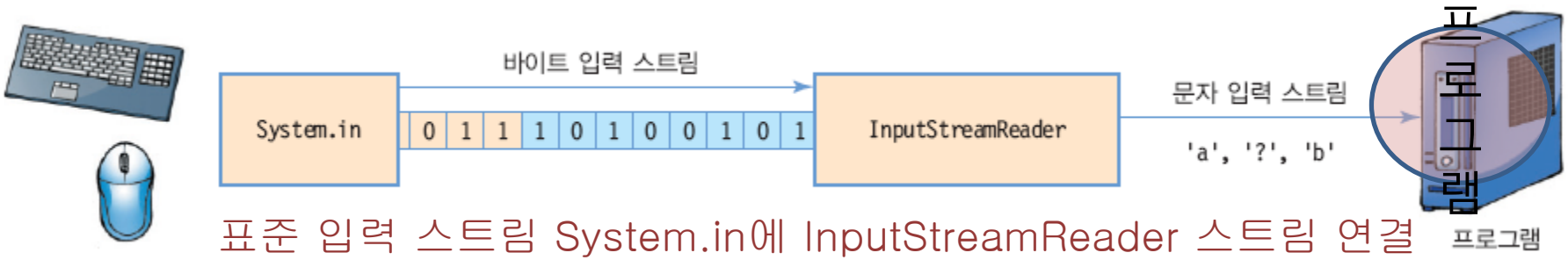
- 흐름의 방향에 따른 분류
  - 입력 스트림(input stream)
  - 출력 스트림(output stream)
- 데이터의 형태에 따른 분류
  - 문자 스트림(character stream)
  - 바이트 스트림(byte stream)

스트림

- 스트림이란?
  - 바이트 단위의 데이터의 연속적인 흐름
  - 입출력 스트림은 입출력 장치와 프로그램 사이의 일련의 데이터 흐름을 의미
  - 스트림을 통해 흘러가는 데이터의 기본 단위는 바이트

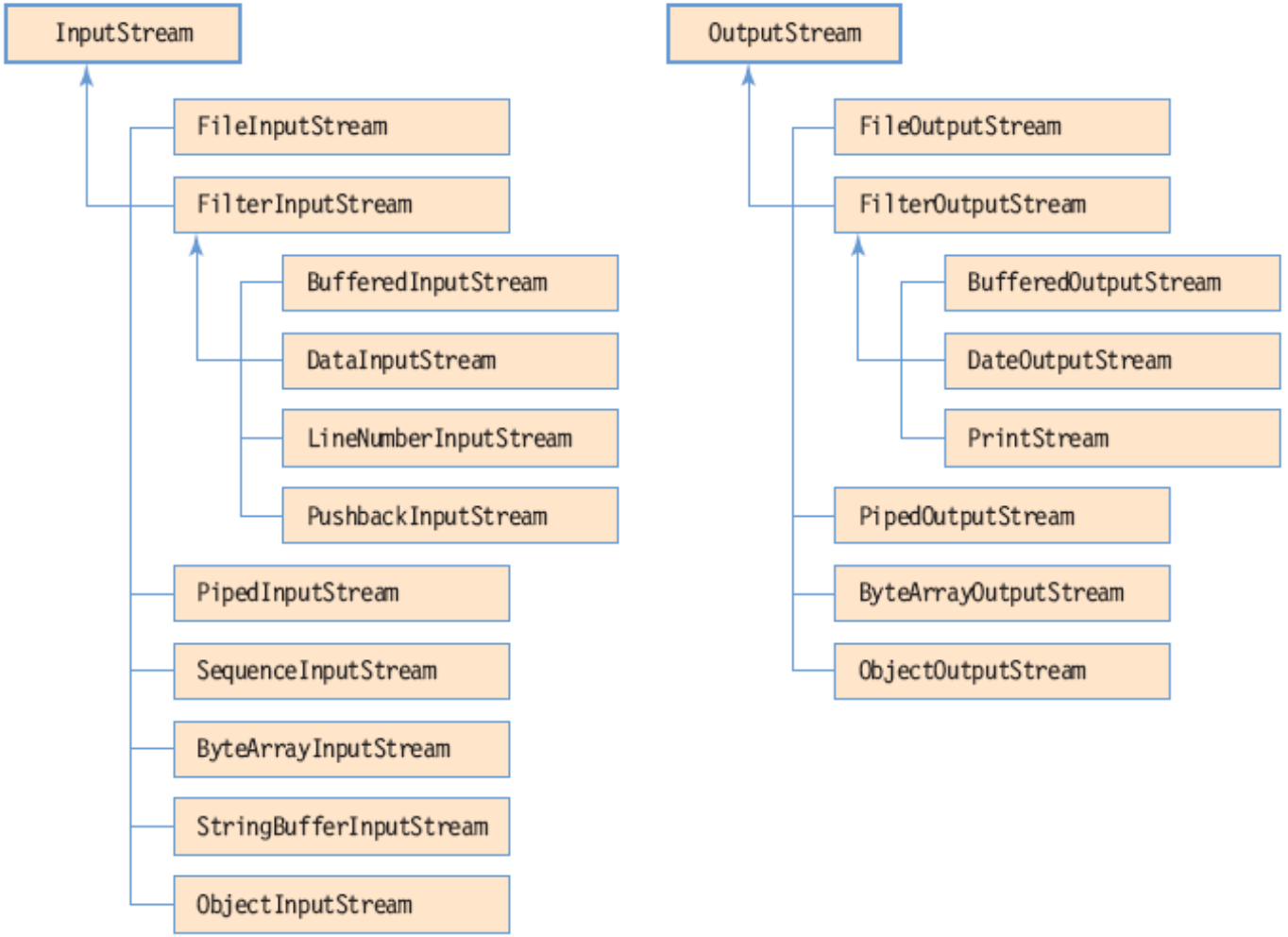
스트림의 특징

- 스트림의 특징
  - 스트림은 단방향이다.
    - 방향에 따라 입력 스트림과 출력 스트림으로 나뉜다.
  - 스트림은 선입선출, 즉 FIFO 구조이다.
  - 스트림은 연결될 수 있다.
  - 스트림은 지연될 수 있다.
    - 입력 스트림이 흐르는 통로인 파이프가 비어 있다면, 컴퓨터는 읽어갈 데이터가 없으므로 기다리게 되고, 반대로 출력 스트림이 흐르는 통로인 파이프에 데이터가 꽉 차 있다면 컴퓨터는 빈 공간이 생길 때까지 기다린다.

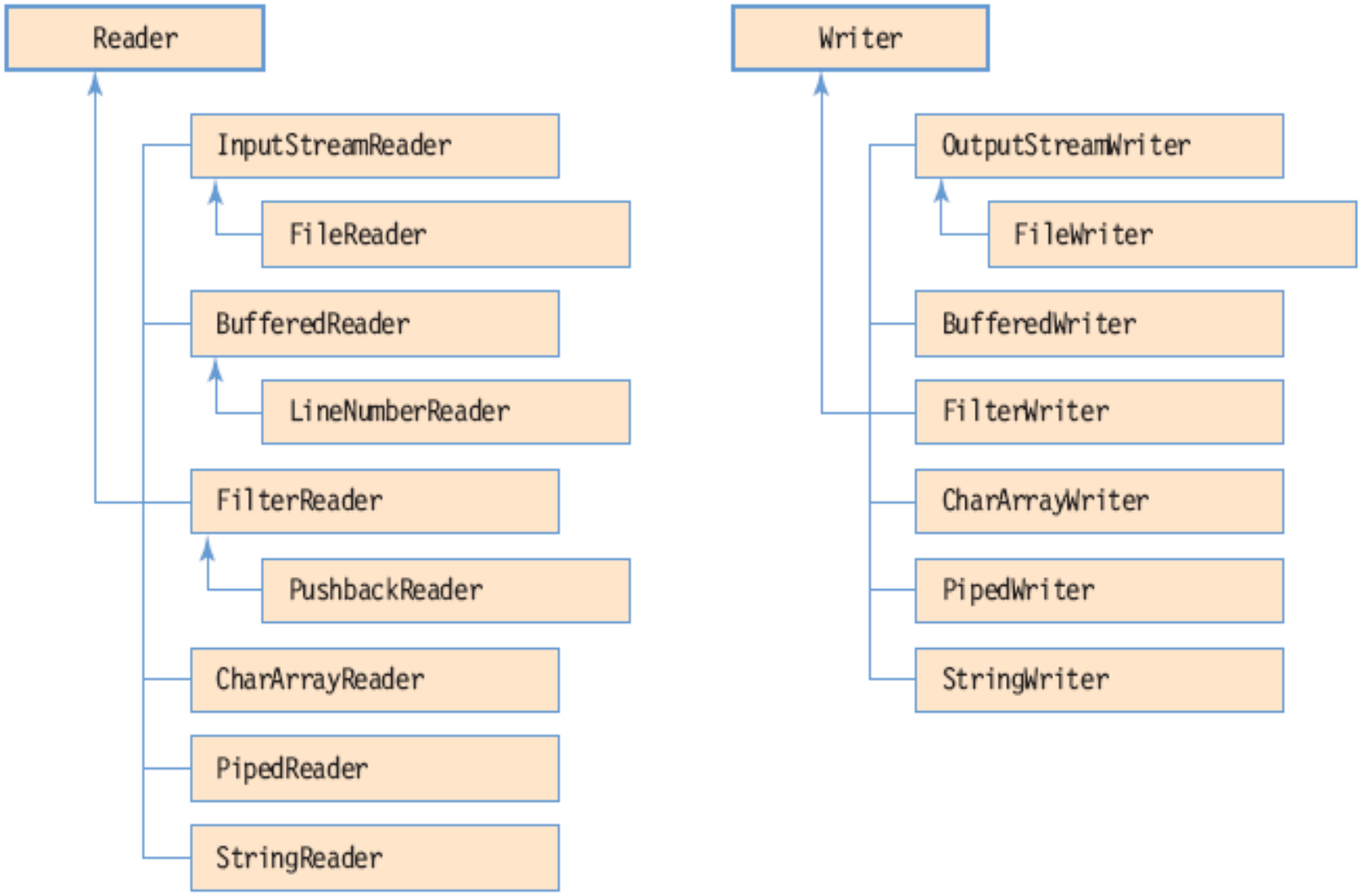


표준 입력 스트림 System.in에 InputStreamReader 스트림 연결

바이트 스트림 클래스 계층 구조



문자 스트림 클래스 계층 구조



바이트 스트림

- 바이트 스트림
  - 바이트 단위의 바이너리 값을 읽고 쓰는 스트림
  - 모든 바이트 스트림은 InputStream과 OutputStream의 서브 클래스
- 바이트 스트림 클래스
  - InputStream/OutputStream
    - java.io 패키지에 포함
    - 추상 클래스로서 바이트 입출력 스트림을 나타내는 모든 클래스의 수퍼 클래스
  - FileInputStream/FileOutputStream
    - 파일로부터 바이트 데이터를 읽거나 파일로 저장하는 클래스
    - 바이너리 데이터인 raw 데이터의 입출력.
  - DataInputStream/DataOutputStream
    - 자바의 기본 데이터 타입의 값을 바이너리 값으로 입출력
    - String을 바이너리 값으로 입출력

문자 스트림

- 문자 스트림
  - 유니 코드로 된 문자 단위의 데이터가 흐르는 스트림
  - 로컬 문자 집합으로 데이터를 자동 변환
  - 모든 문자 스트림은 Reader과 Writer의 서브 클래스이다.
- 바이트 스트림과 차이점
  - 바이트 스트림은 이미지 파일, 동영상 파일 등과 같은 바이너리 데이터 스트림 처리에 적합
  - 문자 스트림은 문자 데이터 스트림만 처리할 수 있다.
- 문자 입력 스트림의 종류
  - Reader, InputStreamReader, FileReader 등
- 문자 출력 스트림의 종류
  - Writer, OutputStreamWriter, FileWriter 등

문자 스트림의 종류

- Reader/Writer
  - java.io 패키지에 포함
  - 추상 클래스로
  - 문자 입출력 스트림을 나타내는 모든 클래스의 수퍼 클래스
- InputStreamReader/OutputStreamWriter
  - 바이트 스트림과 문자 스트림을 연결시켜주는 브릿지 역할
  - 지정된 문자집합을 이용하여 입력 스트림에서 바이트를 읽어 문자로 인코드 또는 문자를 바이트로 디코드하여 출력 스트림으로 출력
- FileReader/FileWriter
  - 텍스트 파일로 문자 데이터를 입출력 한다.
  - FileInputStream과 FileOutputStream을 이용하여 바이트 스트림을 문자 스트림으로 변환

입출력 기능/성능을 향상시키는 클래스들

- 버퍼를 이용해서 입출력 성능을 향상시키는 클래스들
  - 스트림의 종류에 따라 4개의 클래스가 있음

클래스 이름	설명
BufferedInputStream	바이트 입력 스트림을 버퍼링하는 클래스
BufferedOutputStream	바이트 출력 스트림을 버퍼링하는 클래스
BufferedReader	문자 입력 스트림을 버퍼링하는 클래스
BufferedWriter	문자 출력 스트림을 버퍼링하는 클래스

1. System클래스와 기본 출력 처리

- **System 클래스 Member Field**
  - in : InputStream과 연결된 객체로 키보드로부터의 입력을 처리할 수 있음.
  - out : PrintStream 과 연결된 객체로 콘솔로의 출력을 처리할 수 있음.
  - err : 역시 PrintStream 과 연결된 객체로 콘솔로의 출력을 처리할 수 있으나 일반적으로 에러 메시지를 표시할 때 많이 사용함.
- **기본 출력 (PrintStream 클래스)**
  - write Method
  - print Method
  - println Method
  - printf Method

2. 입력과 출력 시 예외 처리

- 자바에서의 안정성과 예외처리
- java.io.IOException 클래스와 throws
- 예외 발생 및 예외 전가

자바에서 키 입력, System.in

- System.in
  - 자바의 표준 입력 스트림
  - java.io의 InputStream 클래스 타입
  - InputStream이 바이트 스트림이므로 문자 스트림으로 변환하려면 InputStreamReader 클래스를 이용
  - 입력 동안 IOException이 발생가능, 예외 처리 필요.

3. Keyboard를 통한 기본 입력처리

- **문자 및 숫자 입력**
  - **아스키코드 = System.in.read();**
  - **숫자 = System.in.read() – 48 or '0';**
  - **문자 = (char)System.in.read();**
- **문자열 입력**
  - **System.in.read(byte[]);**
  - **BufferedReader in = new BufferedReader(new InputStreamReader(System.in));**
    - **(예) String str = in.readLine() ;**
      - **readLine()함수는 한 라인에 있는 모든 데이터(공백류 문자들 포함)를 읽어 문자열 형태로 반환한다.**

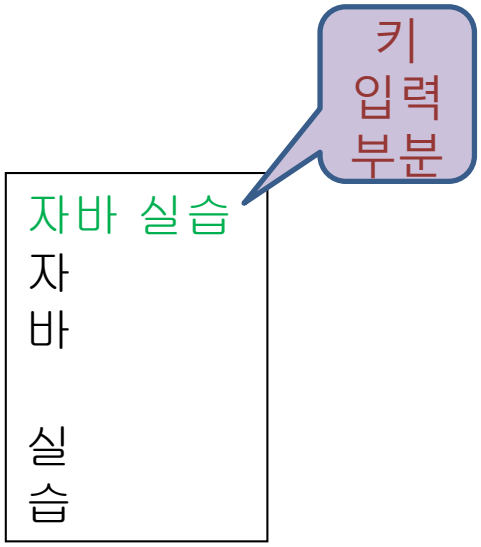
      ❖공백류 문자(White Space)

      ❖ SpaceBar에 의한 공백문자, Tab 키에 의한 문자, EnterKey에 의한 문자.

예제 : 표준 입력 스트림을 이용한 키 입력

다음 소스의 실행 결과는 무엇인가?
System.in을 InputStreamReader에 연결하여 사용자로부터 키를 입력받는다. 입력받은 문자를 화면에 출력하고 사용자가 ctrl-z를 누르면 읽기가 종료된다.

```
import java.io.*;
public class InputExample {
   public static void main (String args[]) {
      InputStreamReader  rd = new
InputStreamReader(System.in);
      try {
         while (true) {
            int a = rd.read();
            if (a == -1)// ctrl-z가 입력되면 read()는 -1을 리턴
                break;
            System.out.println((char)a);  // 입력된 문자 출력
         }
      }
      catch (IOException e) {
         System.out.println("입력 에러 발생");
      }
   }
}
```



Scanner를 이용한 키 입력

- Scanner 클래스
  - java.util.Scanner 클래스
  - Scanner 객체 생성

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

  - import문 필요
    - 소스 맨 앞줄에 사용

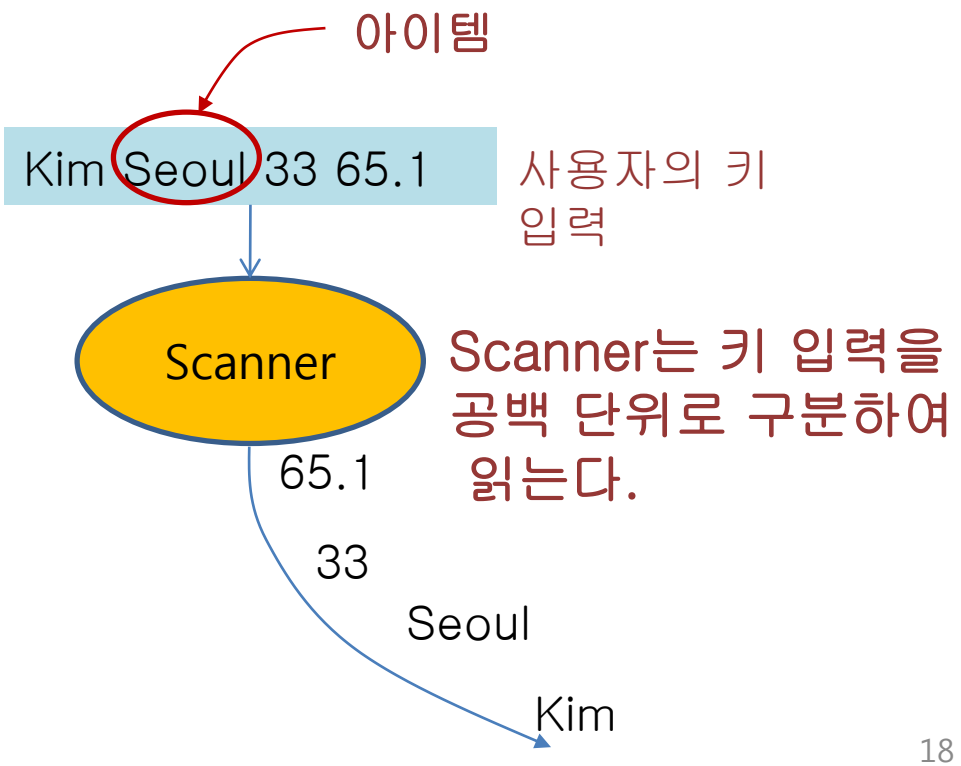
```
import    java.util.Scanner;
```

  - Scanner에서 키 입력 받기
    - **Scanner는 입력되는 키 값을 공백 (‘₩t’, ‘₩f’, ‘₩r’, ‘ ’, ‘₩n’) 으로 구분되는 아이템 단위로 읽음**

Scanner를 이용한 키 입력

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

String name = scanner.next();  // "Kim"
String addr = scanner.next(); // "Seoul"
int age = scanner.nextInt(); // 23
double weight = scanner.nextDouble();// 65.1
```



Scanner 주요 메소드

생성자/메소드	설명
String next()	다음 아이템을 찾아 문자열로 반환
boolean nextBoolean()	다음 아이템을 찾아 boolean으로 변환하여 반환
byte nextByte()	다음 아이템을 찾아 byte로 변환하여 반환
double nextDouble()	다음 아이템을 찾아 double로 변환하여 반환
float nextFloat()	다음 아이템을 찾아 float로 변환하여 반환
int nextInt()	다음 아이템을 찾아 int로 변환하여 반환
long nextLong()	다음 아이템을 찾아 long으로 변환하여 반환
short nextShort()	다음 아이템을 찾아 short로 변환하여 반환
String nextLine()	한 라인 전체('₩n' 포함)를 문자열 타입으로 반환

예제 : Scanner를 이용한 키 입력 연습

Scanner를 이용하여 나이, 체중, 신장 데이터를 키보드에서 입력 받아 다시 출력하는 프로그램을 작성해보자.

```
import java.util.Scanner;
public class ScannerExam {
  public static void main (String args[]) {
    Scanner a = new Scanner(System.in);
    System.out.println("나이, 체중, 신장을 빈칸으로 분리하여 순서대로 입력하세요");
    System.out.println("당신의 나이는 " + a.nextInt() + "살입니다.");
    System.out.println("당신의 체중은 " + a.nextDouble() +"kg입니다.");
    System.out.println("당신의 신장은 " + a.nextDouble()+ "cm입니다.");
  }
}
```

키 입력 부분

```
나이, 체중, 신장을 빈칸으로 분리하여 순서대로 입력하세요
35 75 175
당신의 나이는 35살입니다.
당신의 체중은 75.0kg입니다.
당신의 신장은 175.0cm입니다.
```

println, print, printf

① println() 메서드
- ln은 라인의 약어로서 메소드 내에 기술한 내용을 출력한 후 자동으로 개행(줄을 바꿈)한다.

② print() 메소드
- 메소드내에 기술한 내용만을 출력할뿐 줄 바꿈을 하지 않는다.

③ printf() 메소드
- 자바 5.0에서 추가되었다.
- printf의 f는 format의 약어로서 형식 지정자를 기술하여 원하는 자료 형태로 출력할 수 있는 메소드.
- 문자 데이터를 문자 형태로 출력하기 위해서는 %c라는 형식 지정자를 사용한다.
- 형식 지정자는 %기호 다음에 영문자를 기술하는데 형식 지정자 %d는 정수형 10진수 형태로 출력하게 된다. d는 decimal의 약어이다. 저장된 문자 데이터의 아스키 코드 값으로 출력하려면 %d 사용

printf() 메서드에서 사용하는 서식 지정자(1)

서식의 구성 형태	'%' + 옵션 + 출력 데이터 지정 문자
%c	char 문자 1개를 출력하는 서식
%숫자c	char 문자 1개를 해당 숫자의 자리 수만큼 확보한 상태로 출력하는 서식(숫자가 음수이면 좌측 정렬, 숫자가 양수이면 우측 정렬)
%d(%o, %x)	byte, short , int, long형의 데이터를 10진(8진, 16진)수로 출력하는 서식
%숫자d(%숫자o, %숫자x)	byte, short , int, long형의 데이터를 10진(8진, 16진)수로 해당 숫자의 자리 수만큼 확보한 상태로 출력하는 서식(숫자가 음수이면 좌측 정렬, 숫자가 양수이면 우측 정렬)
%0숫자d(%0숫자o, %0숫자x)	byte, short , int, long형의 데이터를 10진(8진, 16진)수로 해당 숫자의 자리 수만큼 확보한 상태로 출력하는 서식인데 확보한 자리가 공백이면 0 값을 공백에 채우게 하는 서식(숫자가 음수이면 런타임 에러 발생)

CONTINUE

printf() 메서드에서 사용하는 서식 지정자(2)

서식의 구성 형태	'%' + 옵션 + 출력 데이터 지정 문자
%f(%g, %e)	float, double형의 데이터를 실수(근사, 지수)로 출력하는 서식
%숫자f(%숫자g, %숫자e)	float, double형의 데이터를 실수(근사, 지수)로 해당 숫자의 자리 수만큼 확보한 상태로 출력하는 서식(숫자가 음수이면 좌측 정렬, 숫자가 양수이면 우측 정렬). 또한 숫자는 소수 이하 1자리를 나타낼 수도 있다. 예를 들면 '%10.2f' 라는 서식은 전체 10자리를 확보하고 그 중에서 소수점 이하 2자리를 나타내라는 뜻이다. 만약 소수 이하 세 번째 자리에서 반올림이 가능하면 반올림을 하라는 뜻도 된다.
%0숫자f(%0숫자g, %0숫자e)	float, double형의 데이터를 실수(근사, 지수)로 해당 숫자의 자리 수만큼 확보한 상태로 출력하는 서식인데 확보한 자리가 공백이면 0 값을 공백에 채우게 하는 서식(숫자가 음수이면 런타임 에러 발생). 또한 숫자는 소수 이하 1자리를 나타낼 수도 있다. 예를 들면 '%10.2f' 라는 서식은 전체 10자리를 확보하고 그 중에서 소수점 이하 2자리를 나타내라는 뜻이다. 만약 소수 이하 세 번째 자리에서 반올림이 가능하면 반올림을 하라는 뜻도 된다.
%s	문자열(String) 데이터를 출력하는 서식
%숫자s	문자열(String) 데이터를 해당 숫자의 자리 수만큼 확보한 상태로 출력하는 서식(숫자가 음수이면 좌측 정렬, 숫자가 양수이면 우측 정렬)

printf()에서 사용되는 확장 특수 출력 문자(escape sequence)

종 류	의 미
‘₩a'	경고음이 난다.
‘₩n'	엔터 키의 기능을 갖는다. 줄을 바꾼다(new line).
'₩t‘	수평 탭으로 일정한 간격을 띄운다(tab).
'₩b'	백스페이스로 뒤로 한 칸 후진한다(back space).
'₩r‘	동일한 줄맨 앞칸으로 커서만 옮긴다(carriage return).
'₩f'	출력 용지를 한 페이지 넘긴다(form feed).
'₩₩'	₩문자를 의미한다.
'₩''	'문자를 의미한다(single quote).
'₩"'	"문자를 의미한다(double quote).
'₩0'	널 문자를 의미한다(null).

문자 여러개를 집합으로 관리하는 문자열형(String)

```
'AB'          //잘못된 표현

"AB"

char x="AB"; //잘못된 표현

String y="AB";

String y='A'; //잘못된 표현
```

<예제> 문자열 저장하기

```
01:public class Data08 {
02:   public static void main(String[] args){
03:      String y;
04:      y="AB";
05:      System.out.println(y);
06:      y="A";
07:      System.out.println(y);
08:   }
09:}
```