***부동소수점 -> 2진수 규칙**

IEEE 754 명세에 따라 2진수로 전환한다.

모든 값들은 다 숫자 = 정수 = 2진수로 바꿀 수 있다면 저장할 수 있다.

12.375

12를 2진수로 1100

0.375 *2 = 0.750 => 일단 소수점 앞 0을 기록
0.750 * 2 = 1.500 = 앞의 1을 뺀다
0.500 * 2 = 1.000 = 1
=> 0.011

더한다
1100.011
항상 왼쪽에 숫자 한개만 남겨두어야한다
#.###

1.100 001 * 2의 3승 (2진수의 정규화)
|
1100.011 같은 값

32bit(4byte) 메모리라면
부호비트(+ 는 0 - 는 1) , 지수부 (8bit), 가수부(23bit)

지수부는 express -k에 따라 127을 더해 저장한다
가수부는 부호 절대값 sign-Magnitude 그대로 저장한다

같은 수를 저장하더라도 float, double
 몇 바이트에 저장하느냐에 따라 진수 표현이 다르다.

부동소수점이라도 정규화할 때 2진수로 딱 떨어지지 않는 경우가 있다.
그래서 부동소수점은 정수와 다르게 정확하게 2진수로 변환할 수 없는 경우가 있따.
메모리에 정확하게 저장되지 않을 수 있다.

0.0000 .....1 의 오차가 있을 수 있다.

4byte(float) 부동소수점의 유효자릿수는 소수점을 뺀 후 7자리 숫자까지 정상적으로 저장된다.
유효자리가 7자리를 넘어가면 잘려서 저장 될 수 있다.

8byte(double) 부동소수점의 유효자릿수는 소수점을 뺀 후 16자리 숫자까지 정상적으로 저장된다.
유효자리가 7자리를 넘어가면 잘려서 저장 될 수 있다.

그래서 32비트를 단정도라고 부르고 single-precision이라고 부르고 64비트는 두배 정밀하게 값을 저장해서 배정도 double-precision이라고 부른다,

*부동소수점과 메모리크기
부동소수점은 정확하게 최대 최소를 정할 수 없다.
32bit 메모리 4byte는 부호비트, 지수부, 가수부로 구분되는데 = 2진수로 전환하기 때문에
- ? ~ + ? 인지 결정할 수 없다

결론
IEEE 754 규칙에 따라 2진수로 변환하기 때문에 소수점 이상/이하 자릿수를 명시 할 수 없다.
소수점을 제외한 숫자의 자릿수로 가능범위를 표현한다.
=' 유효자릿수'라고 부른다.

유효자릿수 범위의 부동소수점을 2진수로 변환하면 최소의 오차로 값을 저장할 수 있다.

***문자 -> 2진수**
1)ASCII (7bit)
A = 100 0001
B = 100 0010
C = 100 0011
a = 110 0001
? = 011 1111
공백 = 010 0000

문제는 미국식이라 유럽문자를 쓸 수 없습니다.

국제표준 규칙을 만들어
2)ISO-8859-x (8bit)
ASCII 문자 + 유럽 문자
한글 저장 못함

3)Euc-KR (16bit) 최대 65536자
한글 2350자 + 알파
영어는 ISO-8859-1 사용
현대 통용되는 한글 음절 11172자를 모두 표현할 수 없다.
ex)윈도우 3.1까지 똠,똡은 정의가 안되어있어서 표현할 수 없었다
완성된 문자에 대해 2진수를 정의해서 완성형이라고 부른다.

4)조합형(16bir)
모든 한글표현 가능 36,000자 사용가능
국제표준은 아니다. 그래서 안쓴다. 아래아 한글(HWP 에디터)에서만 쓰인다.

초성; ㄱ ㄲ ㄴ ㄷ ㄸ ㄹ ... =>5bit사용
중성; ㅏ ㅐ ㅑ ㅒ ㅓ ... =>5bit 사용
종성; => ㄱ ㄲ ㄳ ㄴ ㄵ ㄶ ㄷ ... => 5bit 사용

관공서는 모든 문자를 저장해야하기 때문에 아래아 한글을 씁니다.
전세계 시장 90퍼센트에서 아래아 한글을 사용했고 그 여파로 MS949 등장

5)MS949 (CP949) (16bit)
=EUCKR(2350자 +) + 알파
기존 EUC-KR은 그대로 사용하고 문자 규칙을 추가한다
출판 업계에서 유통되고있는 문자를 대부분 지원한다.

6)Unicode UTF-16 (16bit)
영어 대,소문자
숫자
특수기호 모두 2byte로 표현
한글은 현대 한글 음절 11172자를 새로 정의 -> 기존 규칙과 호환되지 않는다.
Java에서 문자를 다룰 때 이 규칙을 사용합니다.

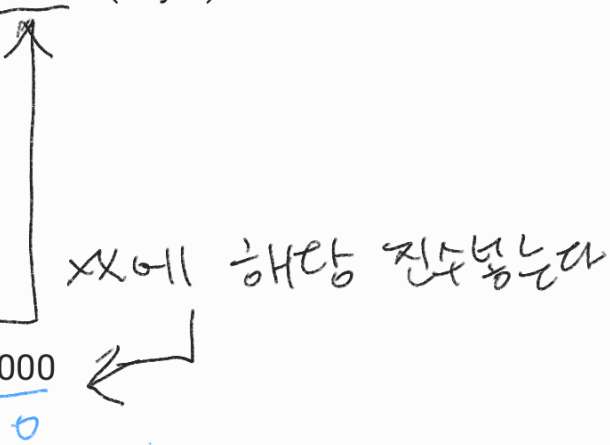
알파벳이 2바이트를 차지하는 것이 마음에 들지 않았다.
기존 영어 문서와 호환이 되지 않았다

영어를 그대로 ISO8859-x로 쓰도록 규칙을 바꿉니다.

7) UTF-8 (8 ~ 32 bit - 1byte ~ 4byte)
Unicode 변형 포맷이라고 부릅니다
Unicode (Universial Character Set) Transformation Format
기존의 ASCII 규칙을 그대로 사용하기 위해 만든 규칙입니다.
이전에 작성한 영어 문서를 그대로 사용할 수 있다.

0000 ~ 007f (ASCII) = 그대로 1byte 사용
0080 ~ 07ff = 110xxxxx 10xxxxxx (2byte) - 유럽문자
0800 ~ ffff = 1110xxxx 10xx xxxx 10xx xxxx (3byte) - 한글 등
10000 ~ ㅡ = 4byte

예) 가
ms949(EUC-KR + 알파) : B0A1 (2byte)
조합형 : 8861 (2byte)
Unicode (UTF-16) : AC00 (2byte)
2진수로 변환 = 1010 1100 0000 0000
UTF-8 : 1110 1010 1011 0000 1000 0000
0x E A B 0 8 0



*문자 - > 2진수 규칙
character set(문자집합) -> 문자를 2진수로 바꾸는 규칙
아스키, iso, euc-kr, 조합형, ms949, utf-16, utf-8

* javac -encoding UTF-8 Hello.java
javac.exe에서 ms949로 저장했다면 Hello.java 를 컴파일 할 때 UTF-8로 인코딩한다.
소스코드는 무조건 utf-8로 저장해야한다.
많은 개발자들이 맥을 사용한다.

***자바로 문자 출력하기**
자바는 '유니코드(UTF-16BE; UCS2)' 라는 문자집합을 사용한다.
\ u 뒤에 문자에 부여된 코드 값을 16진수로 적는다.
u는 소문자로 적어야 한다.

\ 바로 뒤에 8진수로 적을 수 있다.
단, 0 ~ 377 범위의 8진수만 가능하다.
System.out.println('\101'); // A
System.out.println('\122'); // R
System.out.println('\377'); // ÿ
System.out.println('\477'); // 컴파일 오류!

유니코드 값을 직접 적는 경우는
보통 키보드로 문자를 직접 입력할 수 없을 때이다.
```
System.out.println('\u4eba'); // 사람을 뜻하는 '인(人)'이라는 한자의 문자 코드
System.out.println('\u00a9'); // copyright를 뜻하는 '©' 문자 코드
System.out.println('\u03c0'); // 수학의 파이 기호 'π'
System.out.println('\u03a3'); // 수학의 시그마 기호 'Σ'
```

문자의 리터럴 - 문자 집합(character set)
자바는 문자를 다룰 때 2바이트 값으로 다룬다
자바가 사용하는 문자 집합은 '유니코드(Unicode)'이다

유니코드
전 세계의 모든 문자를 컴퓨터에서 일관되게 처리할 목적으로 정의된 산업 표준 규칙이다.
유니코드는 '문자를 2진수로 표현(인코딩; encoding)'할 때, 두 가지 방식(UTF와 UCS)을 사용한다.
자바는 메모리에 문자를 저장할 때는 'UCS(국제 문제 집합)'를 사용하고, 외부로 입출력 할 때는 'UTF'를 사용한다.

*** 'A' 출력하기**
println()에 전달하는 값이 일반적인 수가 아니라 문자 코드임을 지정해야 한다.
숫자 앞에 (char) 를 붙여 이 숫자가 평범한 숫자가 아니라 문자의 UCS-2 코드 값임을 알려줘라.
(char)문자코드
```
System.out.println((char)65);
```

보통 문자 코드를 표현할 때 16진수를 사용한다.

```
System.out.println((char)0x41);
```

***문자의 리터럴 - 작은따옴표('')와 문자 코드**

문자 리터럴을 표현할 때 사용하는 작은 따옴표는 문자의 코드 값을 알려주는 도구이다.

유니코드를 직접 넘겨주기
단, 유니코드임을 알려주기 위해 (char)를 앞에 붙인다.

```
System.out.println((char)65); //A
```

작은 따옴표를 사용하여 문자 코드를 넘겨주기
(char)붙일 필요가 없다.

```
System.out.println('A'); // 65
```

***이스케이프 문자**

화면에 출력하는 문자가 아니라 문자 출력을 제어하는 문자이다.

제어 문자
- \n - Line Feed(LF), 0x0a
- \r - Carrage Return(CR), 0x0d
- \f - Form Feed, 0x0c
- \t - Tab, 0x09
- \b - Backspace, 0x08
- \' - Single Quote, 0x27
- \" - Double Quote, 0x22
- \\ - Backslash, 0x5c

***줄바꿈과 코드 값**

\r - Carrage Return(CR), 0x0d
커서를 맨 처음으로 돌리는 문자
하지만 이 코드는 이클립스로 실행할 시 줄바꿈으로 인식한다.
명령창에서 실행하면 커서를 맨 앞으로 옮긴다.

편집기
ABC
abc
가각간

코드 값
41 42 43 0D 0A 61 62 63
A B C 엔터 뉴라인 가각간

편집기에 엔터가 있으면 0D 0A 코드가 자동으로 삽입된다,

12.375

2진수⇒ 1100.011

1100.011

↓정규화

2진수정규화 1.100011×2^{3} ← 지수부

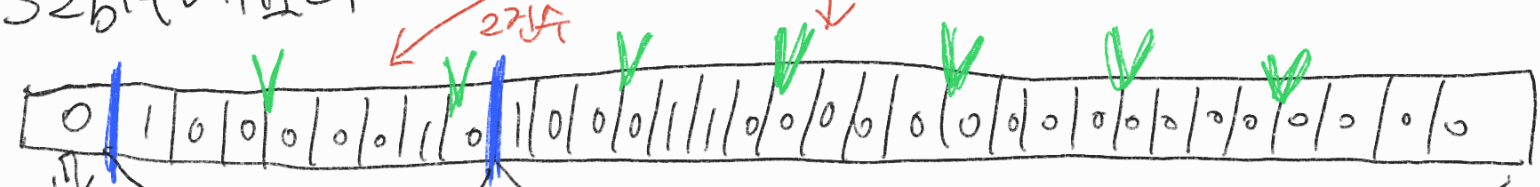
가수부

excess-k⇒ 3+127=130

Sign-magnitude

23자리

32bit메모리



부호비트 (+0, −1)

지수부 (8bit)

가수부(23bit)

이 32bit메모리를 16진수로 짧게 표현

✔ 표시 ⇒ 16진수로 끊어읽기

※ 0x41460000