# Number representation in Java

Thijs Dorssers, Pieter van den Hombergh

Fontys Hogeschool voor Techniek en Logistiek

June 15, 2017





## It is just a bunch of bits

But what they represent depends on the interpretation.
eg `0b 01100010 01110101 01110011 01111001`:

- is 1651864441 when interpreted as integer
- is 1,13194324E21 when interpreted as floating point (float)
- "busy" when interpreted as a string.

So the meaning of a set of bits depend on their intended interpretation. These bits themselves have no clue.
Remember that.



## Scientific notation

How is 0,1 written in scientific notation?
Scientific notation is:

$$mantissa * base^{exponent}$$

In the decimal number system thus base = 10, mantissa always has exactly 1 digit before the decimal mark.
This decimal mark depends on the country you are:

- US and UK this is a . (dot)
- Germany and the Netherlands this is a , (comma)

Examples:

- 100 = 1E2
- avogadro's number 602.214.130.000.000.000.000.000 = 6,0221413e+23

See: https://www.youtube.com/watch?v=Dme-G4rc6NI

Some details will be written on the blackboard, thus jot down notes.



## See NumberRepresentation in demos repository folder

In this java code the following topics will be discussed:

1. Scientific notation
2. Inaccurate double and float calculations
3. Accurate calculations with big decimals
4. Infinite versus finite
5. Binary representation of integers
6. Integer addition, subtraction, multiplication, division (decimal and binary)
7. Binary representation of floats

Some details will be written on the blackboard, thus jot down notes.

## Binary representation of integers

An integer in java is a so-called "32-bit signed two's complement integer".
First bit is not a sign bit, although being 0 for positive numbers and 1 for negative numbers. Some people call it a sign bit, but as you know as a mathematician one has to be very precise.
Based on the circle model the 2 complements 4 bits number system will be explained on the black board, here you can easily deduct the bit representation of negativ numbers.
A trick:
To get the bit representation of a negative number do:

1. Determine bit representation of the belonging positive number
2. Replace all zero's by ones and all one's by zero's
3. Add 1 to the result of step 2

Some details will be written on the blackboard, thus jot down notes.





## Binary representation of floats IEEE 754

On the black board: 32 bit internal representation of 0,1
Decimal 0,1 = 1/16 + 1/32 + 1/256 (see spreadsheet 0.1_in_binary_form.xlsx) and in binary form 0,00011001....
These dots express an infinite series of 0 and 1.

So not every finite series of decimals can be expressed with a finite series of bits.
Computer internal representation of 0,1:

- (1 bit) sign bit = 0
- (23 bits) mantissa = 1,1001... and normalized mantissa = 1001..
- (8 bits) exponent = -4, and biased exponent = -4+127 is decimal 127 - 4 = binary 01111111 - 100 = 01111011

Check this with http://www.binaryconvert.com
Some details will be written on the blackboard, thus jot down notes.





## More examples

On the black board:

- convert IEEE754 binary 11000001111000000000000000000000 to decimal
- convert decimal 12.375 to IEEE754 binary
- use `http://www.h-schmidt.net/FloatConverter/IEEE754.html` to find the accuracy of the 32 bit IEEE754 representation of 0.3
- exam examples





## Solutions of more examples

- signbit=-, exponent: 10000011=131, 131-127=4, mantissa=1.110000000000000=1.75 result=-28
- 12.375=1100.011, mantissa=1.100011, normalized=100011, exponent=3, biased exponent=130=10000010, signbit=0 result=01000001010001100000000000000000
- transformed back we get 0.30000001192092896, accuracy 7 decimals after the decimal dot

## Links and exercises





For more information about float representation see http://introcs.cs.princeton.edu/java/91float/ up to "Rounding of decimal fractions".
A handy helper app is provided by Harald Schmidt on his site:
http://www.h-schmidt.net/FloatConverter/IEEE754.html.
Exercises:

1. Given is the binary representation: -0.01$\overline{0011}$
2. Convert the binary representation of item 1 with pencil and paper to a 32 bit IEEE 754 floating point representation.
3. Calculate the binary representation of -16.375 with pencil and paper
4. Convert the result of exercise 3 to a 32 bit IEEE 754 floating point representation.
5. Suppose you want to add the following 2 binary IEEE 754 floating point numbers, what's than the problem?
   0 10000000 1010000 00000000 00000000
   0 10000010 1001100 00000000 00000000



## Solution of exercises





1. -
2. signbit=1, exponent=-2 biased exponent=125=64+32+16+8+4+1=01111101, mantissa=0.01$\overline{0011}$ normalized=$\overline{0011}$ in 23 bit 00110011001100110011001l, result=10111110100110011001100110011
3. -16.375=-(16+0.25+0.125), binair=-10000.011, transformation to IEEE754 is solved next
4. signbit=1, exponent=4 biased=131=10000011, mantissa=1.0000011, normalized=0000011, result=11000001100000110000000000000
5. The problem is how to deal with the different exponents?