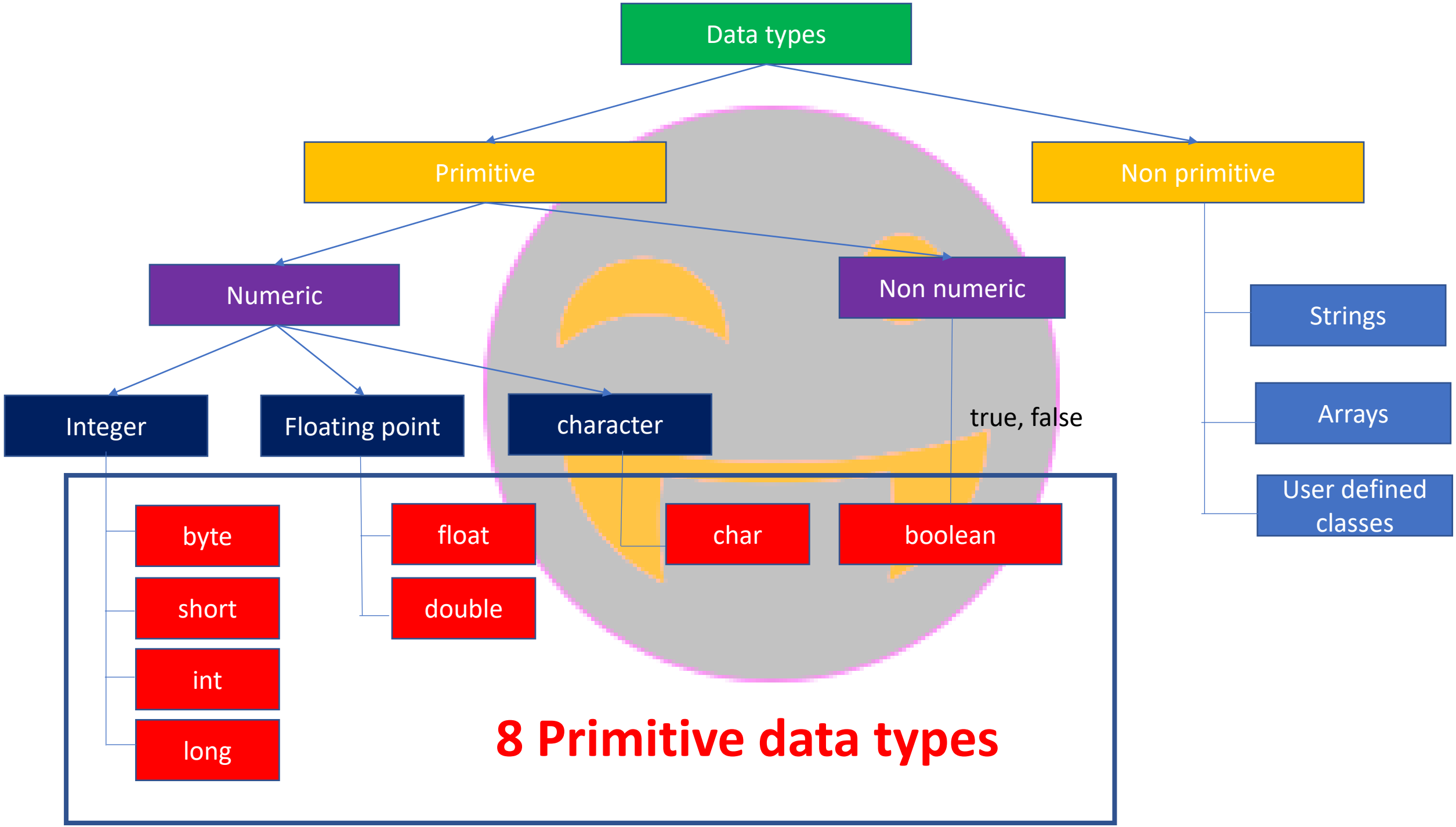
Chapter 22

Data types in Java

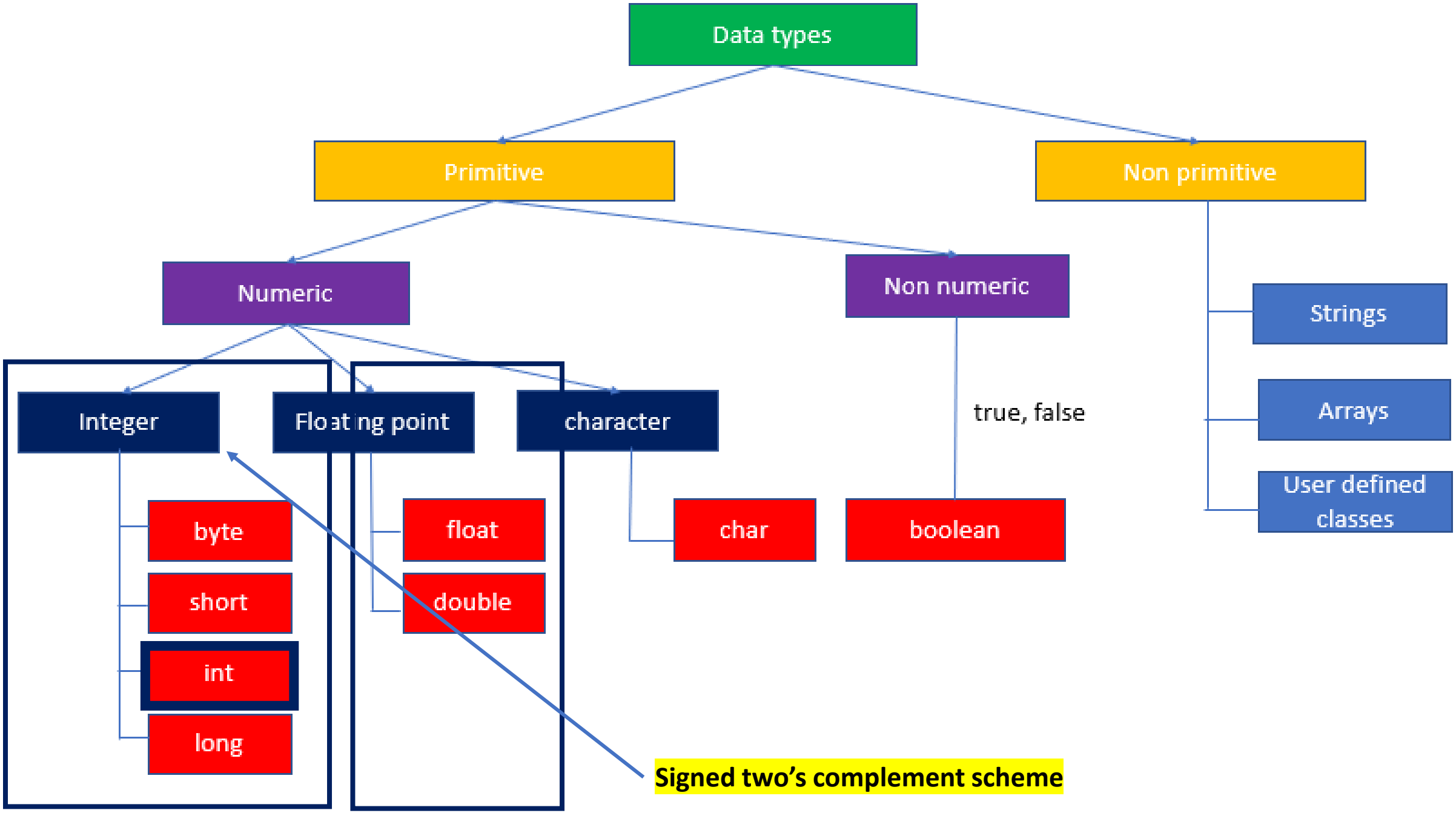
- Practical
- Floating point numbers


Data types
Primitive
Non primitive
Numeric
Non numeric
Strings
Arrays
User defined classes
Integer
Floating point
character
true, false
byte
short
int
long
float
double
char
boolean
8 Primitive data types

Integer types

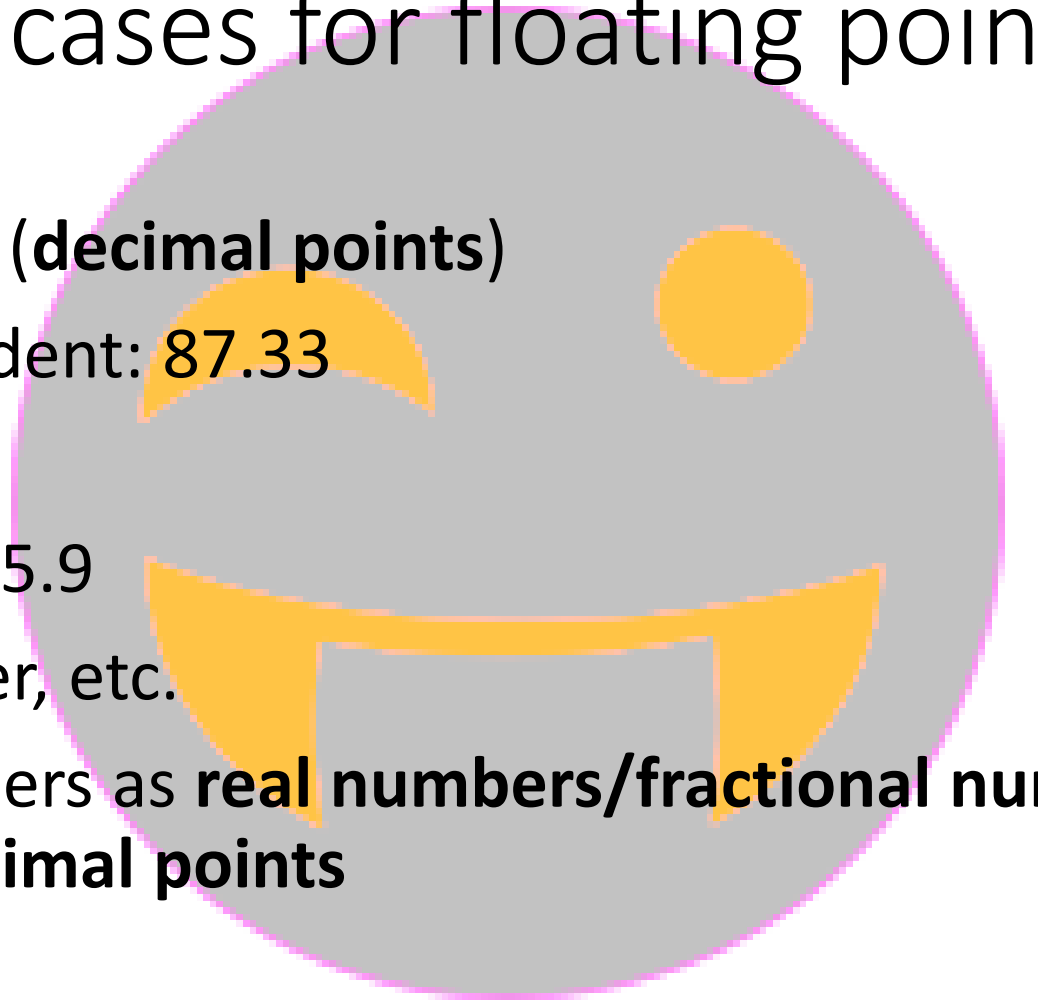
Type	Bit depth	Range	Default values
byte	1 byte(8 bits)	-2^7 to 2^7-1	0
short	2 bytes(16 bits)	-2^15 to 2^15-1	0
int	4 bytes(32 bits)	-2^31 to 2^31-1	0
long	8 bytes(64 bits)	-2^63 to 2^63-1	0

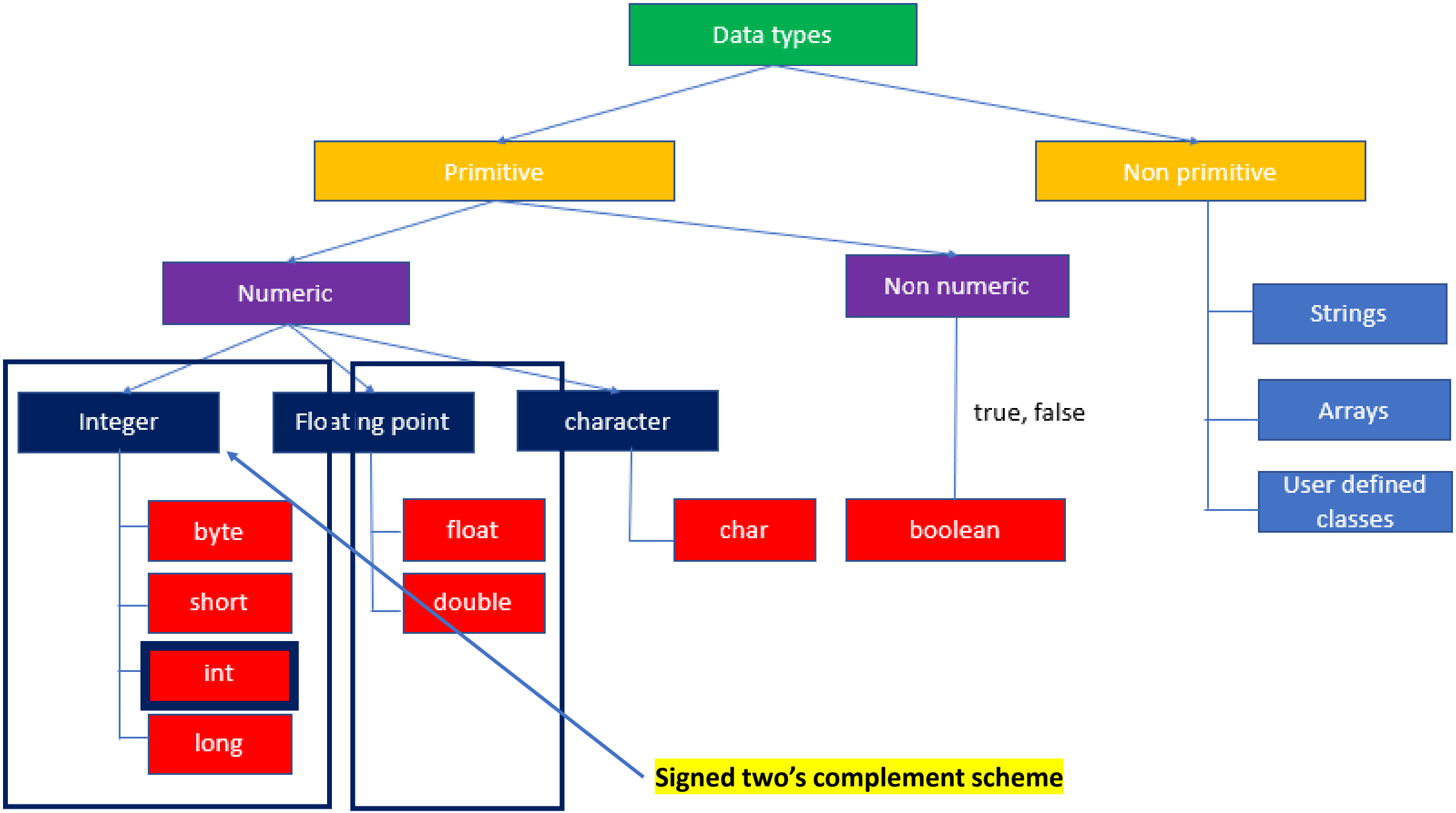
Signed two's complement scheme


Data types
Primitive
Non primitive
Numeric
Non numeric
Strings
Arrays
User defined classes
Integer
Floating point
character
true, false
byte
short
int
long
float
double
char
boolean
Signed two's complement scheme

Common use cases for floating point values

- Fractional numbers (**decimal points**)
- Percentage of a student: 87.33
- Movie ratings: 4.3
- Height of a person: 5.9
- Average of a number, etc.
- We call these numbers as **real numbers/fractional numbers/floating point numbers/decimal points**




Data types
Primitive
Non primitive
Numeric
Non numeric
Strings
Arrays
User defined classes
Integer
Floating point
character
true, false
byte
short
int
long
float
double
char
boolean
Signed two's complement scheme

Float

- **Uses 32 bits** to store floating point number in the IEEE 754 format
- **Single-precision** 32 bit IEEE 754 floating point
- 4 bytes of size
- -2^31 to 2^31-1
- ~~-2,14,74,83,648 to 2,14,74,83,647~~
- **-3.4E38 to 3.4E38**

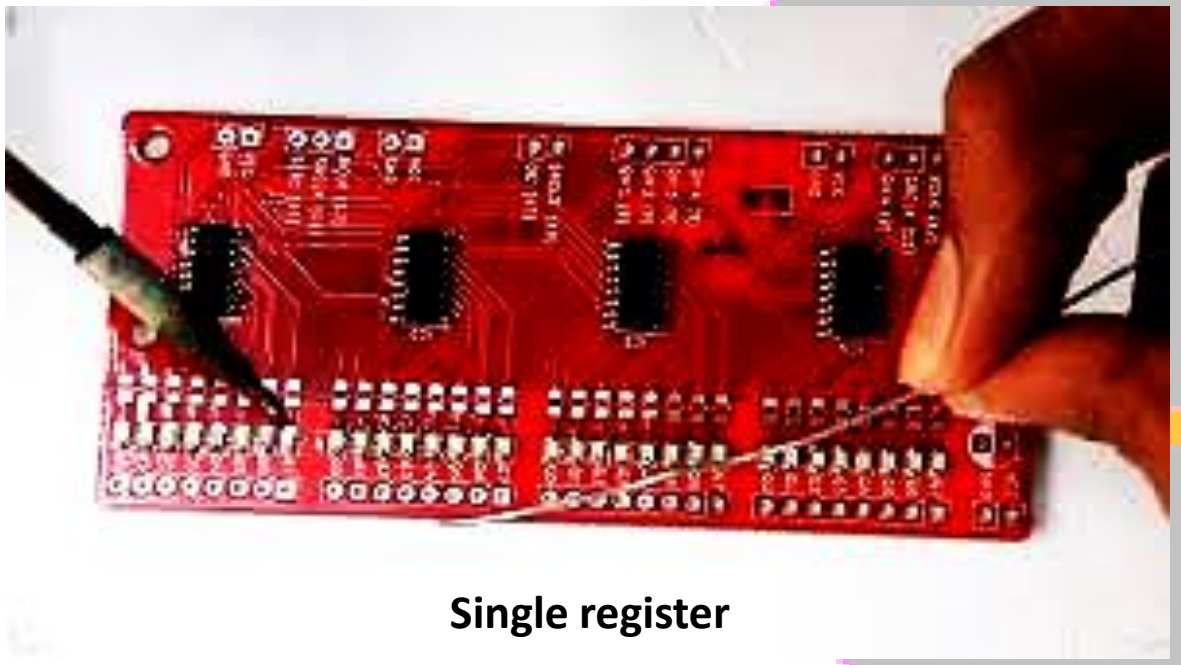
Why is it called single and double?
why the range is different?

Double

- **Uses 64 bits** to store floating point number in the IEEE 754 format
- **Double-precision** 64 bit IEEE 754 floating point
- 8 bytes of size
- -2^63 to 2^63-1
- ~~-92,23,37,20,36,85,47,75,808 to 92,23,37,20,36,85,47,75,807~~
- **-1.7E308 to 1.7E308**

Registers

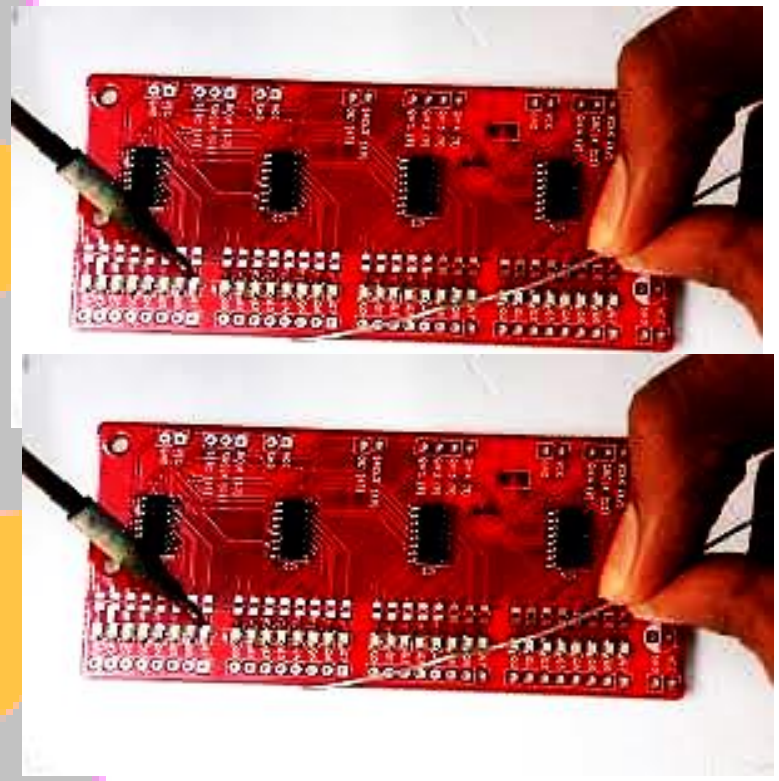
16 bit registers can represent half Precision floating point numbers



Single register

32 bit register

Single precision float will fit in one register



2 registers

64 bit register

Double precision float will fit in two registers

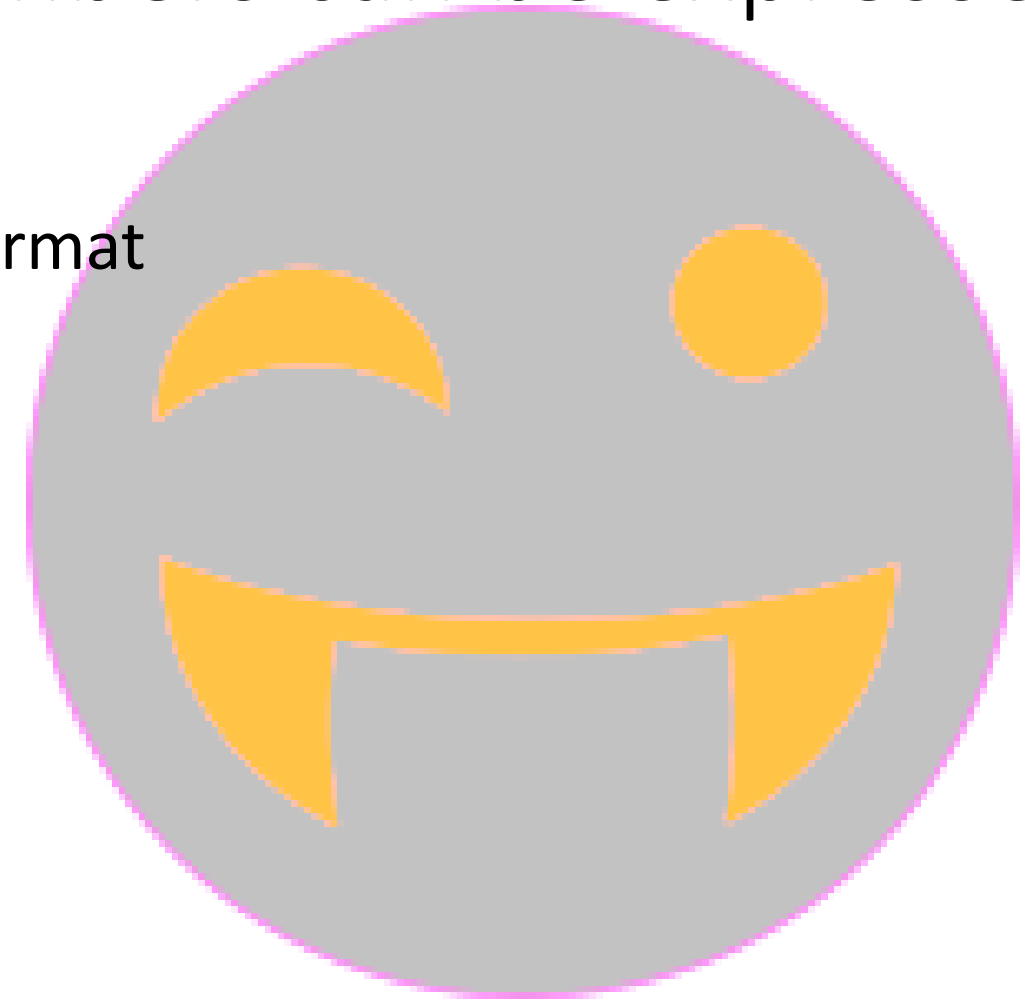
Precision

- **Precision** of a floating point value indicates how **many digits the value can have after the decimal point.**
- precision of **float is 6-7** decimals
- precision of **double is 15-16** decimals

- Definition: Single precision or double precision refer to the number of machine words (4 bytes/32 bit blocks) used to store a real number with varying degrees of accuracy

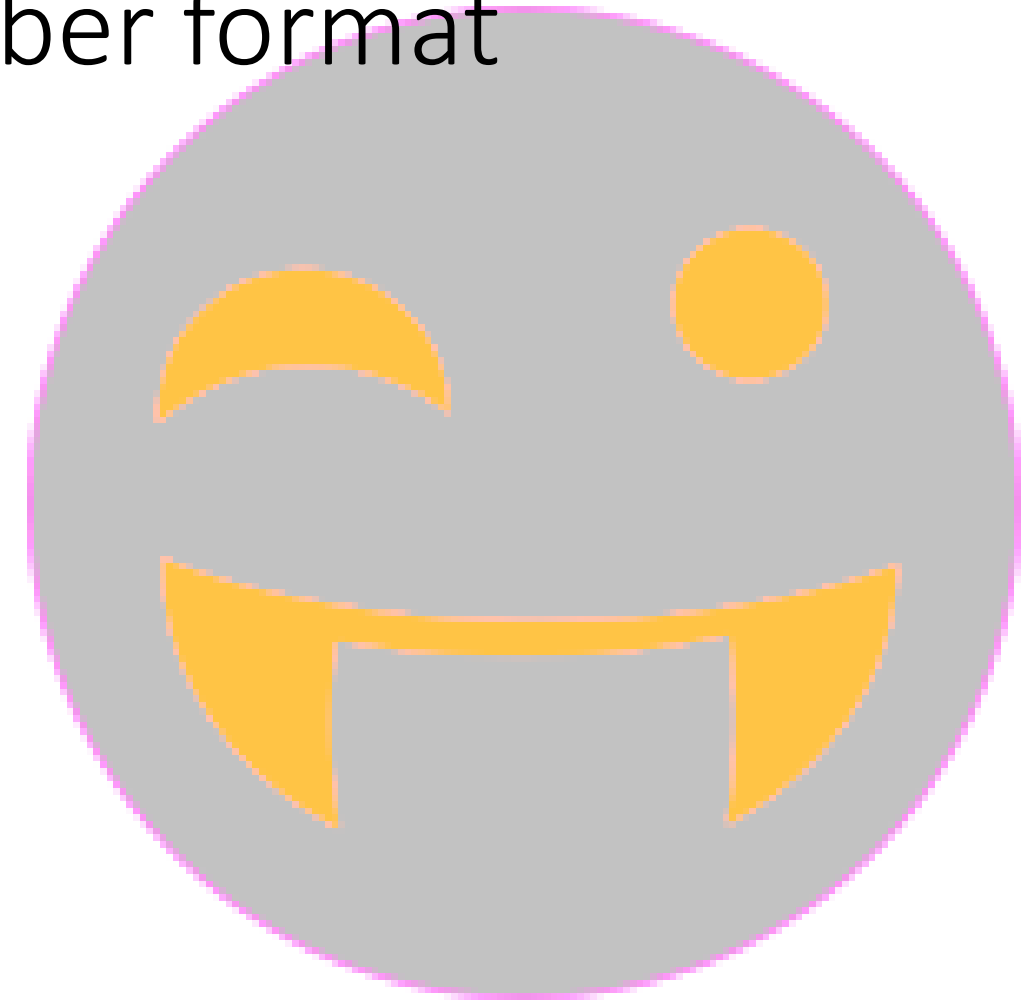
A floating numbers can be expressed in two formats

- Decimal number format
- Scientific notation



Decimal number format

- 5F or 5f
- 9.f or 9.0f or 9.F
- 2.59F



Scientific notation

- Scientific notation is used when working with large numbers
- Ex: mass of earth: 5.9700000000000000000000000
- 5.97*10^24
- How to represent this exponents in programming?
- we use scientific notation as: 5.97e24
- e/E represents **“10 to the power of”**
- 5.97*10^-24
- 5.97e+24
- 5.97e+24

34.776 × 10^6

1000000

1000000

0.000034 × 10^6

0.000034e6

5.67 × 100/100

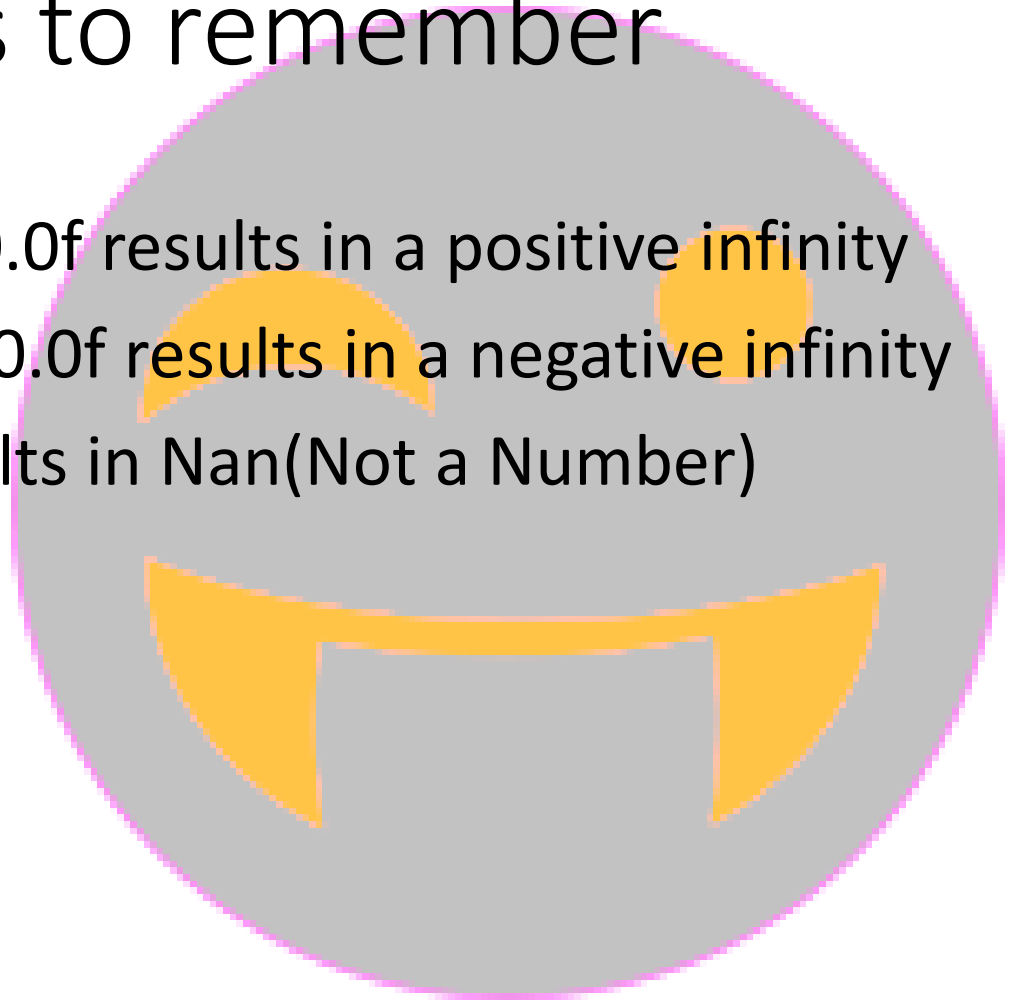
567 × 1/100

567 × 10^-2

567e-2

Special values to remember

- Dividing a float by 0.0f results in a positive infinity
- Dividing a float by -0.0f results in a negative infinity
- Dividing 0 by 0 results in Nan(Not a Number)



Float special values

float Constants	Meaning
Float.POSITIVE_INFINITY	Positive infinity of type float
Float.NEGATIVE_INFINITY	Negative infinity of type float
Float.NaN	Not a Number of type float
Float.MAX_VALUE	The largest positive value that can be represented in a float variable. This is equal to 3.4 x 10^38 approx.
Float.MIN_VALUE	The smallest positive value greater than zero that can be represented in a float variable. This is equal to 1.4 x 10^-45.

Program

```
    floatBasket = 6.838338399992 82f;
    System.out.println(floatBasket);
    floatBasket = 653e9f;
    System.out.println(floatBasket);
     float f1 = 3.25f;
float f2 = 32.5e-1f;
float f3 = 0.325e+1f ;
float f4 = 0.325e1f;
float f5 = 0.0325e2f;
float f6 = 0.0325e2f;
float f7 = 3.25e0f;

System.out.println(f1/-0f);// w  ww .j a v a2 s. c  o m
System.out.println(f2);
System.out.println(f3);
System.out.println(f4);
System.out.println(f5);
System.out.println(f6);
System.out.println(f7);
```

Float

- We can assign an integer to float data type
- Automatically converted to float
- By default fractional point values are treated as double
- Should be suffixed with f or F
- Size - 32 bits

Size: 4 bytes

Range: -3.4E38 to 3.4E38

Default value: 0.0f = 0.0

Precision: 6-7 digits

```
void floatDemo(){
    floatVar = 1.22;
    System.out.println(floatVar);
}
```

```
void floatDemo(){
    floatVar = 10;
    System.out.println(floatVar);
}
```

```
java:63: error: incompatible types: possible lossy conversion from double to float
  floatVar = 1.22;
```

Double

- By default fractional point values are treated as double
- Optionally suffixed with d or D
- We can assign an integer to double data type
- Size - 64 bits
- 1.7 double 34, 38 and 308 to remember
- E can be either upper or lower case

Size: 8 bytes

Range: -1.7E308 to 1.7E308

Default value: 0.0d

Precision: 15-16 digits

Issues with floating point values

- Avoid using float and double if you want definite results
- Perform addition and subtraction on 1.0f and 0.8f
- It is not just with Java, but with other programming languages as well
- Use **BigDecimal** if you want definite results

```java
void floatIssues(){
    heading("Float issues");
    float a = 1.0f;
    float b = 0.8f;
    float add = a+b;//good
    float sub = a-b;//issue

    BigDecimal a1 = new BigDecimal("1.0");
    BigDecimal b1 = new BigDecimal("0.8");
    System.out.println(a1.subtract(b1));
}
```

What next?

IEEE 754 scheme to store fractional numbers

చిన్న బ్రేక్ చిటికలో వచ్చేస్తా