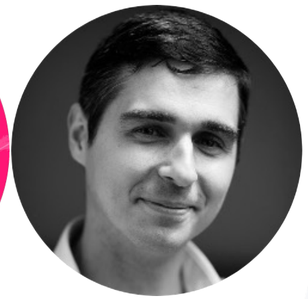
Working with Numbers

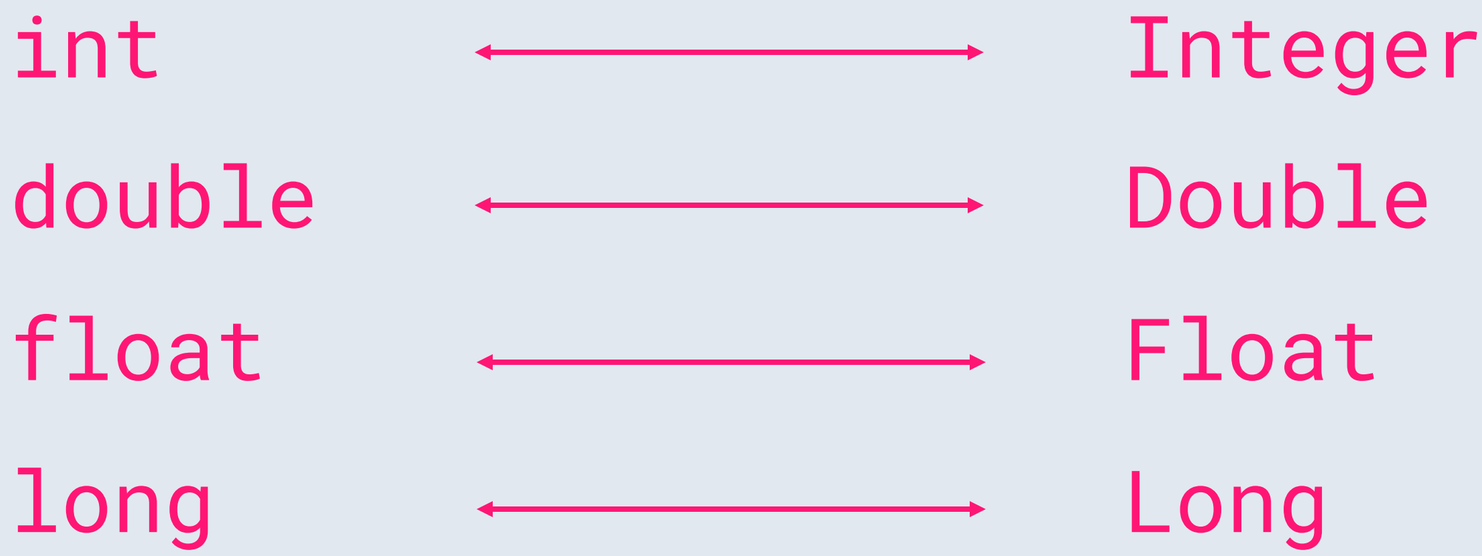


Andrejs Doronins
Software Developer in Test

Overview

- Converting
- Checking
- Formatting
- Localizing
- Rounding
- Comparing

Primitives and Their Wrappers



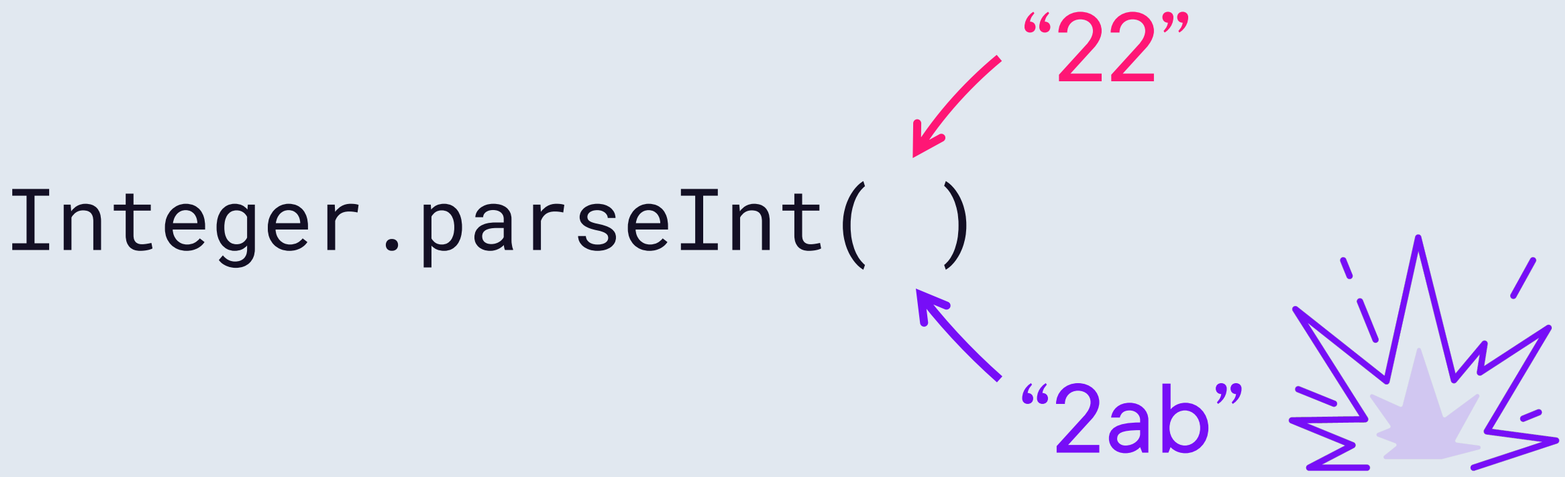
```
List<Integer> ints;
```

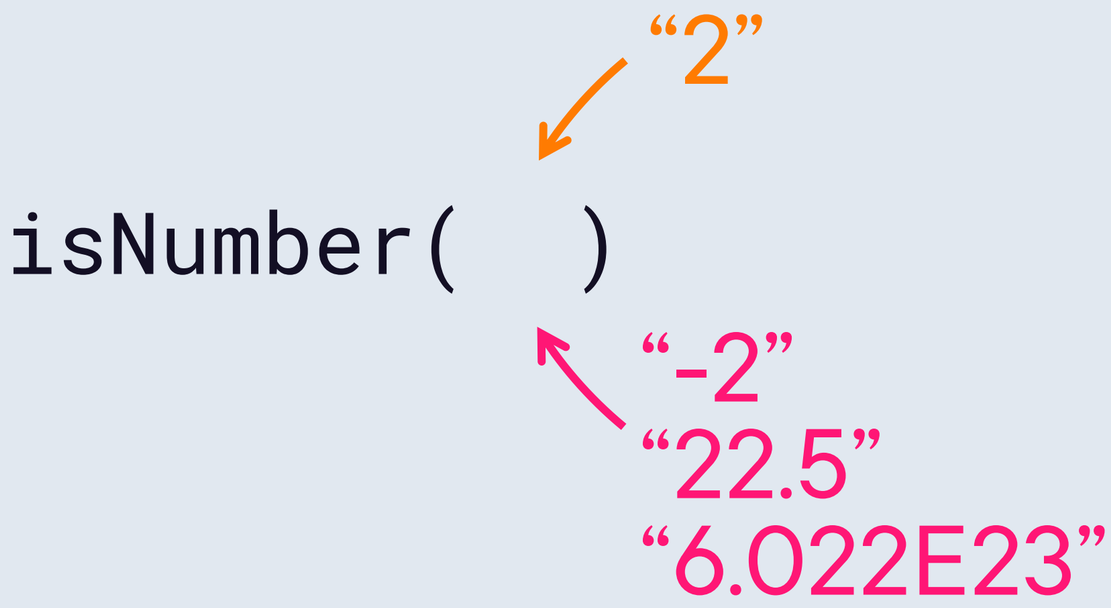
```
List<int> ints;
```

```
class Integer {

    int MIN_VALUE = 0x80000000;
    int MAX_VALUE = 0x7fffffff;

    String toBinaryString(int i);
}
```


“22”
Integer.parseInt( )
“2ab”


“2”
isNumber( )
“-2”
“22.5”
“6.022E23”

Joshua Bloch, Effective Java, 3rd Edition

“Item 69: Use exceptions only for exceptional conditions”

Understand the requirements:

- **Acceptable input**
- **Where it comes from**

Clean your data (prevent) instead of hacking around dirty data (react)

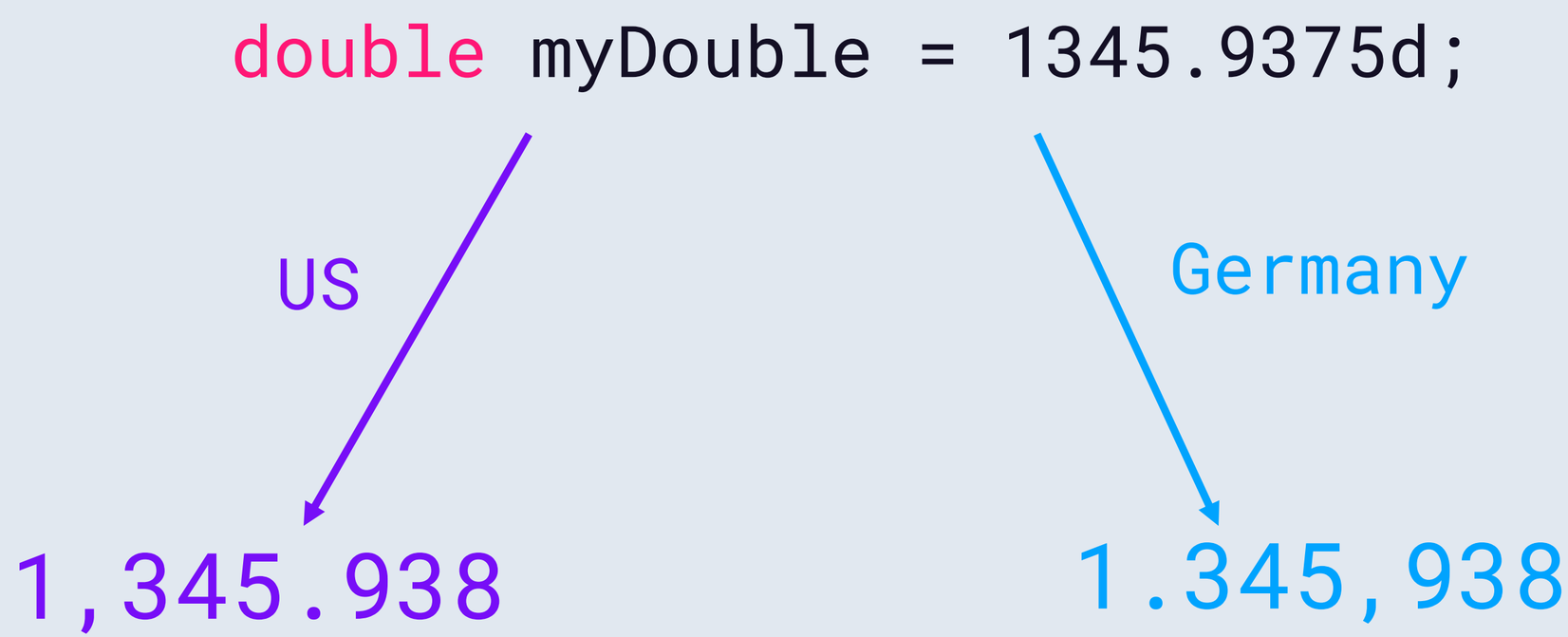
String > Number

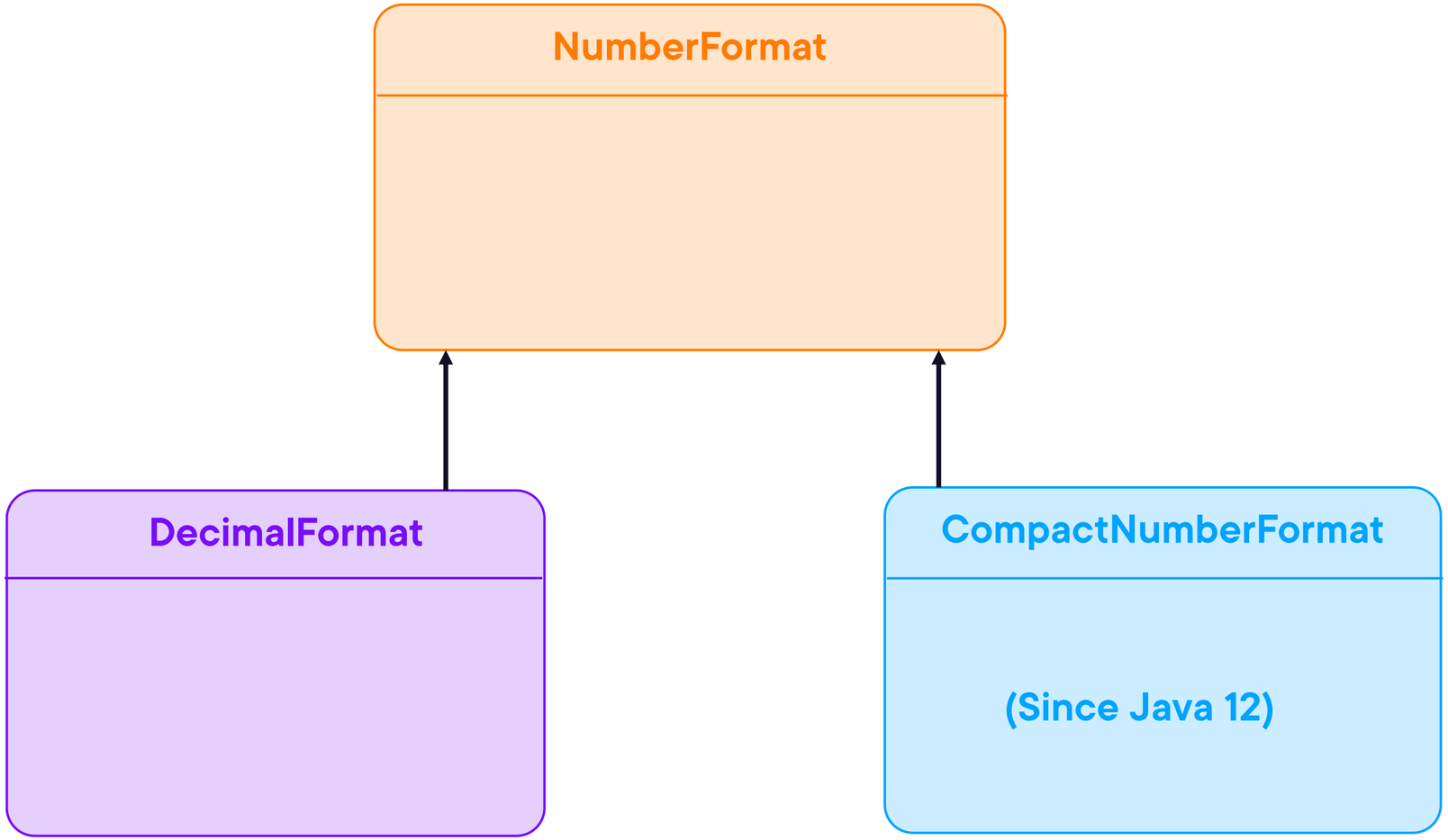
Number > String

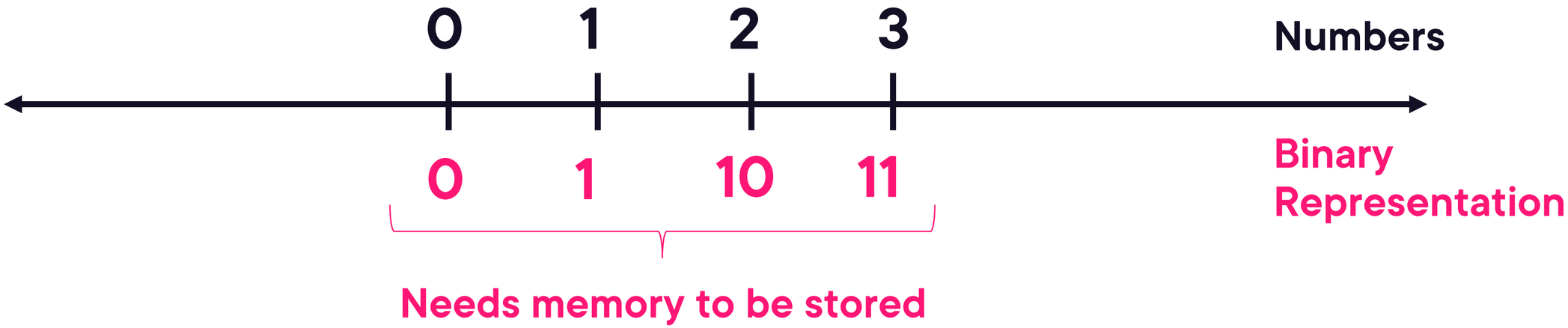
```
System.out.println(int i);
System.out.println(boolean i);
System.out.println(double i);
// etc.
```

How do I influence the format of my numbers

Commas instead of dots
Add or remove leading or trailing zeroes
Rounding


double myDouble = 1345.9375d;
US
Germany
1,345.938
1.345,938


NumberFormat
DecimalFormat
CompactNumberFormat
(Since Java 12)


0
1
2
3
Numbers
0
1
10
11
Binary
Representation
Needs memory to be stored

0.1
0.01
0.001
∞

- Need infinite memory to store
- Approximate representation

```
class Money {

    BigDecimal amount;
    Currency currency;
}
```

JSR 354 - Money and Currency

Be aware of issues with floating-point arithmetic

Consider the loss of precision

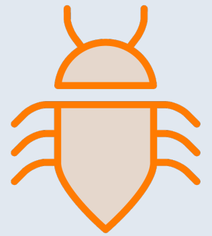
Prefer BigDecimal

For Money - use a dedicated API (amount + currency)


round
3.12567022

```
/**
 * Returns the closest long to the argument
 * with ties rounding to positive infinity
 */
round(double a)
```

4.4 -> 4
4.5 -> 5



BUT! -4.5 -> -4

```
Math.round();

DecimalFormat {}

BigDecimal {}
```

Banker’s Rounding

Problem: Always rounding up from 0.5 creates a disbalance of results (upward)

Solution: RoundingMode.HALF_EVEN

round(1.5) → (up)

round(2.5) → (down)

round(3.5) → (up)

round(4.5) → (down)

Banker's Rounding

Before Java 7: Random class

After Java 7: ThreadLocalRandom

- higher quality random numbers
- Very fast

Higher quality random numbers

- **New solution: the probability of randomness is higher**
- **Old solution: certain number ranges are more likely to be produced than others**

Summary

Most common number operations

Up Next:

Solving Tasks with Dates and Times