Numbers and Maths

Java Developer

StayAhead Training; January, 2023

Strings to Numbers

Numbers and Maths

- Apps often consume data as Strings, even when the values are numeric
- Each of the wrapper classes (Integer, Double, etc.) have static methods for creating primitive numbers from String objects, e.g.

```
var wholeNumber = Integer.parseInt("42");

var fractionalNumber = Double.parseDouble("2.95");
```

- If the String argument is not a valid number then an exception is thrown

Arithmetic and Unary Operators

Numbers and Maths

- Arithmetic operators: `+` `-` `*` `/` `%` (modulus/remainder)
- Unary operators are those that require only one operand
- `-` (negation) yields a version of the operand with its sign (+/-) flipped
- `++` (increment) increments the operand by one
- `--` (decrement) decrements the operand by one
- `!` (not) yields a version of the (boolean) operand that is inverted

Arithmetic and Unary Operators

Numbers and Maths

- The increment and decrement unary operators may be prefix or postfix
- When included as a part of a composite expression the choice matters, e.g.

```
// prefix
var y = 1;
var result1 = 3 * ++y; // y is incr. first; result1 is 6

// postfix
var z = 1;
var result2 = 3 * z++; // z incr. last; result2 is 3
```

The Math Class

Maths and Numbers

- The Math class (java.lang) contains many static fields and methods
- Fields: PI and E (the base of the natural logarithms)
- Methods (a selection):
  - `abs` returns the absolute value of the arg
  - `pow` returns the value of the first arg raised to the power of the second
  - `sqrt` returns the rounded positive square root of the arg
  - `max` returns the larger of the two args
  - `min` returns the smaller of the two args
  - `random` returns a random double >= 0 and < 1

The Math Class

Maths and Numbers

- Methods (a selection, cont.):
  - `signum` returns 1 if the arg is > 0, -1 if the arg < 0, and 0 if the arg is 0
  - `sin` returns the trigonometric sine of the arg (angle in radians)
  - `cos` ditto the cosine
  - `tan` ditto the tangent
  - `toDegrees` returns the arg (radians) in degrees
  - `toRadians` returns the arg (degrees) in radians
  - `ceil` returns the smallest double >= arg
  - `floor` returns the largest double <= arg
  - `round` returns the closest integer to the arg

The BigDecimal Class

Numbers and Maths

- Primitives should **not** be used for monetary values, e.g.

```
var x = 1;
var y = 0.3;
var result = x / y; // 3.3333333333333335
```

- The reason is complicated and has to do with the way floating point numbers are stored in memory

- The BigDecimal class (java.lang) should be used for high-precision arithmetic and for calculations requiring control over scale and rounding

The BigDecimal Class

Numbers and Maths

- BigDecimal objects can be created in a variety of ways but your best bet is to use the constructor that accepts a String, e.g.

```
var bd1 = new BigDecimal("0.1");
```

- If you must create a BigDecimal from a number value then use valueOf, e.g.

```
var bd2 = BigDecimal.valueOf(0.1);
```

- Note that `0.1` has no exact representation as a floating point number - try `0.1 * 3` in Java code and review the result

The BigDecimal Class

Numbers and Maths

- The BigDecimal class has methods for performing computations, e.g.

```
var sum = bd1.add(bd2);

var difference = bd1.subtract(bd2);

var product = bd1.multiply(bd2);

var quotient = bd1.divide(bd2);
```

- The return value in each case is a new BigDecimal

The BigDecimal Class

Numbers and Maths

- An operation, the result of which cannot be represented exactly, throws an exception, e.g.

```
var bd1 = new BigDecimal("1");

var bd2 = new BigDecimal("0.3");

var result = bd1.divide(bd2); // throws an exception
```

- The divide method is overloaded to accept a second arg - RoundingMode

```
var result = bd1.divide(bd2, RoundingMode.DOWN); // 3
```

The BigDecimal Class

Numbers and Maths

- The BigDecimal class has a setScale method that accepts a scale and RoundingMode, e.g.

```
var bd1 = new BigDecimal("1234.5678");

var bd2 = bd1.setScale(2, RoundingMode.UP); // 1234.57
```

- NB: BigDecimal instances/objects are immutable and cannot be modified, that is why every BigDecimal instance method returns a new BigDecimal

Number Formatting

Numbers and Maths

- The NumberFormat class (java.text) provides for the formatting of numbers
- NumberFormat instances are not created in the usual way
- To get an instance you must call a static method, e.g. getInstance
- The instance will use the default Locale to do the formatting
- The Locale may be specified when you obtain the instance
- The format method returns a String representation of the given number

Number Formatting

Numbers and Maths

- Number formatting, e.g.

```
var formatter = NumberFormat.getInstance();
var result = formatter.format(123456); // "123,456"
```

- With a specified Locale, e.g.

```
var locale = new Locale("DE");
var formatter = NumberFormat.getInstance(locale);
var result = formatter.format(123456); // "123.456"
```

Number Formatting

Numbers and Maths

- Number formatting with scaling and rounding, e.g.

```
var formatter = NumberFormat.getInstance();

formatter.setMaximumFractionDigits(2);

formatter.setRoundingMode(RoundingMode.UP);

var result = formatter.format(123.456); "123.46"
```

Number Formatting

Numbers and Maths

- Currency formatting, e.g.

```
var formatter = NumberFormat.getCurrencyInstance();
var result = formatter.format(123.456); // "£123.46"
```

- With a specified Locale, e.g.

```
var locale = new Locale("DE");
var formatter = NumberFormat.getCurrencyInstance(locale);
var result = formatter.format(123.456); // "123,46 €"
```

Number Formatting

Numbers and Maths

- Compact formatting (since Java 12), e.g.

```
var formatter = NumberFormat.getCompactNumberInstance();
var result = formatter.format(2000); // "2K"
```

- With a specified Locale and style, e.g.

```
var locale = new Locale("DE");
var formatter = NumberFormat.getCompactNumberInstance(
    locale, NumberFormat.Style.LONG);
var result = formatter.format(2000); // "2 Tausend"
```