Some floating point number basics

Precision defines the number of digits in a decimal number.

Precision includes digits to both the left, and the right of the decimal point.

Scale is the number of digits to the right of the decimal point in a number.

Examples	Precision	Scales
15.456	5	3
8	1	0
100000.000001	12	6
.123	3	3

Fractional Numbers may not be able to be accurately represented by floating point numbers

First, many floating-point numbers can only approximate the decimal number they're supposed to represent.

Think about the number 1/3, which has an infinitely repeating decimal.

This fraction, by necessity will get rounded at some point.

It's scale wasn't big enough as a float, when we applied it to our fairly large life insurance benefit.

The double worked well enough for this calculation, but it was still off by a penny, when reconciling a rounded dollar amount, with a calculated double amount.

BigDecimal

The BigDecimal class stores a floating point number in two integer fields.

The first field holds an **unscaled value**, with a type of BigInteger, another class in the java.math package, that can store numbers bigger than even long values.

The second field is the **scale**, which can be positive, 0 or negative.

A positive or 0 scale defines how many digits in the unscaled value, are after the decimal point.

You can use a negative scale as well, which means the unscaled value is multiplied by ten to the power of the negation of the scale.

BigDecimal Examples

Using the examples I showed you on the first slide, if I created BigDecimal instances, using those values, I would get the unscaled value, and scale, shown here.

Examples	Unscaled Value	Scale	Precision
15.456	15456	3	5
8	8	0	1
100000.000001	100000000001	6	11
.123	123	3	3

The unscaled value has all the digits and no decimal point.

The decimal point's placement is determined by the scale, and again precision identifies the number of digits.