BigDecimal

**Precision is the total number of digits in a number.** For example,

|  |  |
| --- | --- |
| **Number** | **Precision** |
| 246 | 3 |
| 56.256 | 5 |
| 1.0074256 | 8 |

When we apply a precision of 3 to the value 3.14159265358, using new MathContext(3), we get a result of 3.14.  
  
When precision is applied to a **BigDecimal**, the number may undergo rounding (default being RoundingMode.HALF\_UP) and the value may change. Below table shows how **BigDecimal** rounds-off numbers to given precisions:

|  |  |  |
| --- | --- | --- |
| **Number** | **Precision** | **Rounded Value** |
| 2.34 | 2 | 2.3 |
| 2.36 | 2 | 2.4 |
| 24.6 | 2 | 25 |
| 56.246 | 4 | 56.25 |
| 1.0074256 | 3 | 1.01 |

In **BigDecimal**, **Scale is the number of digits to the right of the decimal point.** For example, in 3.14159265358, scale is 11. When a precision of 3 is applied, the value is rounded to 3.14 and scale becomes 2. When scale is not explicitly set in **BigDecimal**, it automatically calculates scale as the number of digits after the decimal point in the given value.

Scale is used to represent the number of digits after the decimal point.

Below examples show how a value changes when precision and scale are applied together:

|  |  |  |  |
| --- | --- | --- | --- |
| **Number** | **Precision** | **Scale** | **Final Value** |
| 24.6 | 2 | 5 | 25.00000 |
| 56.256 | 4 | 1 | 56.3 |
| 1.0074256 | 3 | 10 | 1.0100000000 |

When precision 4 is applied on 56.246, it becomes 56.25. When a scale of 1 is applied on 56.25 using BigDecimal.setScale(int newScale), it is further rounded up to 56.3, by providing a rounding behaviour of RoundingMode.HALF\_UP.  
  
The method **setScale**(int newScale) assumes that the value does not change after scale adjustment (for example, applying a scale of 2 on 2.3400 results in 2.34, where value has not changed).  
  
We see a loss in value of 0.05 when a scale of 1 is applied on 56.25. In such cases, we should use setScale(int newScale, int roundingMode) with appropriate rounding behaviour to avoid ArithmeticException. Precise rounding behaviour can be specified by using various options specified in the Enum [RoundingMode](http://docs.oracle.com/javase/7/docs/api/java/math/RoundingMode.html" \t "_blank).  
  
It is important to understand scaling behaviour for different numbers, especially when scientific notation is used. Refer to [BigDecimal](http://docs.oracle.com/javase/7/docs/api/java/math/BigDecimal.html" \t "_blank) for more details.  
  
Retype and submit the below code to understand how precision is applied in **BigDecimal**.