java.math

Enum RoundingMode

java.lang.Object
  java.lang.Enum<RoundingMode>
    java.math.RoundingMode

All Implemented Interfaces:

Serializable, Comparable<RoundingMode>

```
public enum RoundingMode
extends Enum<RoundingMode>
```

Specifies a *rounding behavior* for numerical operations capable of discarding precision. Each rounding mode indicates how the least significant returned digit of a rounded result is to be calculated. If fewer digits are returned than the digits needed to represent the exact numerical result, the discarded digits will be referred to as the *discarded fraction* regardless the digits' contribution to the value of the number. In other words, considered as a numerical value, the discarded fraction could have an absolute value greater than one.

Each rounding mode description includes a table listing how different two-digit decimal values would round to a one digit decimal value under the rounding mode in question. The result column in the tables could be gotten by creating a `BigDecimal` number with the specified value, forming a `MathContext` object with the proper settings (`precision` set to `1`, and the `roundingMode` set to the rounding mode in question), and calling `round` on this number with the proper `MathContext`. A summary table showing the results of these rounding operations for all rounding modes appears below.

Summary of Rounding Operations Under Different Rounding Modes

	Result of rounding input to one digit with the given rounding mode							
Input Number	**UP**	**DOWN**	**CEILING**	**FLOOR**	**HALF_UP**	**HALF_DOWN**	**HALF_EVEN**	**UNNECESSARY**
5.5	6	5	6	5	6	5	6	throw ArithmeticException
2.5	3	2	3	2	3	2	2	throw ArithmeticException
1.6	2	1	2	1	2	2	2	throw ArithmeticException
1.1	2	1	2	1	1	1	1	throw ArithmeticException
1.0	1	1	1	1	1	1	1	1
-1.0	-1	-1	-1	-1	-1	-1	-1	-1
-1.1	-2	-1	-1	-2	-1	-1	-1	throw ArithmeticException
-1.6	-2	-1	-1	-2	-2	-2	-2	throw ArithmeticException
-2.5	-3	-2	-2	-3	-3	-2	-2	throw ArithmeticException
-5.5	-6	-5	-5	-6	-6	-5	-6	throw ArithmeticException

This enum is intended to replace the integer-based enumeration of rounding mode constants in `BigDecimal` (`BigDecimal.ROUND_UP`, `BigDecimal.ROUND_DOWN`, etc.).

Since:

1.5

See Also:

`BigDecimal`, `MathContext`

Enum Constant Summary

Enum Constants

Enum Constant and Description
CEILING Rounding mode to round towards positive infinity.
DOWN Rounding mode to round towards zero.
FLOOR Rounding mode to round towards negative infinity.
HALF_DOWN Rounding mode to round towards "nearest neighbor" unless both neighbors are equidistant, in which case round down.
HALF_EVEN Rounding mode to round towards the "nearest neighbor" unless both neighbors are equidistant, in which case, round towards the even neighbor.
HALF_UP Rounding mode to round towards "nearest neighbor" unless both neighbors are equidistant, in which case round up.
UNNECESSARY Rounding mode to assert that the requested operation has an exact result, hence no rounding is necessary.
UP Rounding mode to round away from zero.

Method Summary

Methods

Modifier and Type	Method and Description
static **RoundingMode**	**valueOf**(int rm) Returns the RoundingMode object corresponding to a legacy integer rounding mode constant in **BigDecimal**.
static **RoundingMode**	**valueOf**(**String** name) Returns the enum constant of this type with the specified name.
static **RoundingMode**[]	**values**() Returns an array containing the constants of this enum type, in the order they are declared.

Methods inherited from class java.lang.Enum

clone, compareTo, equals, finalize, getDeclaringClass, hashCode, name, ordinal, toString, valueOf

Methods inherited from class java.lang.Object

getClass, notify, notifyAll, wait, wait, wait

Enum Constant Detail

UP

```
public static final RoundingMode UP
```

Rounding mode to round away from zero. Always increments the digit prior to a non-zero discarded fraction. Note that this rounding mode never decreases the magnitude of the calculated value.

Example:

Input Number	Input rounded to one digit with UP rounding
5.5	6
2.5	3
1.6	2
1.1	2
1.0	1
-1.0	-1
-1.1	-2
-1.6	-2
-2.5	-3
-5.5	-6

DOWN

```
public static final RoundingMode DOWN
```

Rounding mode to round towards zero. Never increments the digit prior to a discarded fraction (i.e., truncates). Note that this rounding mode never increases the magnitude of the calculated value.

Example:

Input Number	Input rounded to one digit with DOWN rounding
5.5	5
2.5	2
1.6	1
1.1	1
1.0	1
-1.0	-1
-1.1	-1
-1.6	-1
-2.5	-2
-5.5	-5

CEILING

```
public static final RoundingMode CEILING
```

Rounding mode to round towards positive infinity. If the result is positive, behaves as for `RoundingMode.UP`; if negative, behaves as for `RoundingMode.DOWN`. Note that this rounding mode never decreases the calculated value.

Example:

Input Number	Input rounded to one digit with CEILING rounding
5.5	6
2.5	3
1.6	2
1.1	2
1.0	1
-1.0	-1
-1.1	-1
-1.6	-1
-2.5	-2
-5.5	-5

FLOOR

```
public static final RoundingMode FLOOR
```

Rounding mode to round towards negative infinity. If the result is positive, behave as for `RoundingMode.DOWN`; if negative, behave as for `RoundingMode.UP`. Note that this rounding mode never increases the calculated value.

Example:

Input Number	Input rounded to one digit with FLOOR rounding
5.5	5
2.5	2
1.6	1
1.1	1
1.0	1
-1.0	-1
-1.1	-2
-1.6	-2
-2.5	-3
-5.5	-6

HALF_UP

```
public static final RoundingMode HALF_UP
```

Rounding mode to round towards "nearest neighbor" unless both neighbors are equidistant, in which case round up. Behaves as for `RoundingMode.UP` if the discarded fraction is ≥ 0.5; otherwise, behaves as for `RoundingMode.DOWN`. Note that this is the rounding mode commonly taught at school.

Example:

Input Number	Input rounded to one digit with HALF_UP rounding
5.5	6

2.5	3
1.6	2
1.1	1
1.0	1
-1.0	-1
-1.1	-1
-1.6	-2
-2.5	-3
-5.5	-6

HALF_DOWN

```
public static final RoundingMode HALF_DOWN
```

Rounding mode to round towards "nearest neighbor" unless both neighbors are equidistant, in which case round down. Behaves as for `RoundingMode.UP` if the discarded fraction is > 0.5; otherwise, behaves as for `RoundingMode.DOWN`.

Example:

Input Number	Input rounded to one digit with `HALF_DOWN` rounding
5.5	5
2.5	2
1.6	2
1.1	1
1.0	1
-1.0	-1
-1.1	-1
-1.6	-2
-2.5	-2
-5.5	-5

HALF_EVEN

```
public static final RoundingMode HALF_EVEN
```

Rounding mode to round towards the "nearest neighbor" unless both neighbors are equidistant, in which case, round towards the even neighbor. Behaves as for `RoundingMode.HALF_UP` if the digit to the left of the discarded fraction is odd; behaves as for `RoundingMode.HALF_DOWN` if it's even. Note that this is the rounding mode that statistically minimizes cumulative error when applied repeatedly over a sequence of calculations. It is sometimes known as "Banker's rounding," and is chiefly used in the USA. This rounding mode is analogous to the rounding policy used for `float` and `double` arithmetic in Java.

Example:

Input Number	Input rounded to one digit with `HALF_EVEN` rounding
5.5	6
2.5	2

1.6	2
1.1	1
1.0	1
-1.0	-1
-1.1	-1
-1.6	-2
-2.5	-2
-5.5	-6

UNNECESSARY

```
public static final RoundingMode UNNECESSARY
```

Rounding mode to assert that the requested operation has an exact result, hence no rounding is necessary. If this rounding mode is specified on an operation that yields an inexact result, an `ArithmeticException` is thrown.

Example:

Input Number	Input rounded to one digit with UNNECESSARY rounding
5.5	throw `ArithmeticException`
2.5	throw `ArithmeticException`
1.6	throw `ArithmeticException`
1.1	throw `ArithmeticException`
1.0	1
-1.0	-1
-1.1	throw `ArithmeticException`
-1.6	throw `ArithmeticException`
-2.5	throw `ArithmeticException`
-5.5	throw `ArithmeticException`

Method Detail

values

```
public static RoundingMode[] values()
```

Returns an array containing the constants of this enum type, in the order they are declared. This method may be used to iterate over the constants as follows:

```
for (RoundingMode c : RoundingMode.values())
    System.out.println(c);
```

Returns:

an array containing the constants of this enum type, in the order they are declared

valueOf

```
public static RoundingMode valueOf(String name)
```

Returns the enum constant of this type with the specified name. The string must match *exactly* an identifier used to declare an enum constant in this type. (Extraneous whitespace characters are not permitted.)

Parameters:

`name` - the name of the enum constant to be returned.

Returns:

the enum constant with the specified name

Throws:

`IllegalArgumentException` - if this enum type has no constant with the specified name

`NullPointerException` - if the argument is null

valueOf

```
public static RoundingMode valueOf(int rm)
```

Returns the `RoundingMode` object corresponding to a legacy integer rounding mode constant in `BigDecimal`.

Parameters:

`rm` - legacy integer rounding mode to convert

Returns:

`RoundingMode` corresponding to the given integer.

Throws:

`IllegalArgumentException` - integer is out of range