**Module** java.base
**Package** java.math

# Enum Class RoundingMode

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

**All Implemented Interfaces:**

Serializable, Comparable<RoundingMode>, Constable

---

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

Specifies a *rounding policy* 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.

More generally, a rounding policy defines a mapping from the real numbers to a subset of representable values. In the case of BigDecimal, the representable values are a function of the precision being used in the computation. Assuming the mathematical result is within the exponent range of BigDecimal, the mathematical result will be exactly representable in the result precision or will fall between two adjacent representable values. In the case of falling between two representable values, the rounding policy determines which of those two bracketing values is the result. For in-range real numbers, for a given set of representable values, a rounding policy maps a continuous segment of the real number line to a single representable value where the real number numerically equal to a representable value is mapped to that value.

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**

| Input Number | Result of rounding input to one digit with the given rounding mode | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | 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. ).

**API Note:**

Five of the rounding modes declared in this class correspond to rounding-direction attributes defined in the *IEEE Standard for Floating-Point Arithmetic*. Where present, this correspondence will be noted in the documentation of the particular constant.

**See *Java Language Specification*:**