BIGINTEGER & BIGDECIMAL

- BigInteger

BigInteger class is used for the mathematical operation which involves very big integers calculations that are outside the limits of all available primitive data types

Key-Points

Arbitary Precision - Can represent integers of any size, limited only by available memory.

Immutable - Operations on BigInteger returns new BigInteger objects; the orignal object remains unchanged.

Wide Range of Operations - Supports arithmetic, bitwise, modular and other mathematical operations

```
Object
↳ java.lang.Number
    ↳ java.math.BigInteger
      extends number &
      implements comparable
```

Constructors

BigInteger bi = new BigInteger(String val);

Constructs a BigInteger from a string representation.

BigInteger bi = new BigInteger (String val, int radix);

Constructs a BigInteger from a string representation in the specified base (radix).

BigInteger bi = new BigInteger(int numBits, Random rnd);

Constructs a randomly generated BigInteger with the specified bit length

BigInteger bi = new BigInteger(int bitLength, int certainty, Random rnd);

Constructs a probable prime 'BigInteger' with the specified bit length.

Methods

Arithmetic operations:

add (BigInteger val)
subtract (BigInteger val)
multiply (BigInteger val)
divide (BigInteger val)
mod (BigInteger val)

Bitwise Operations:

and (BigInteger val)
or (BigInteger val)
xor (BigInteger val)
not (BigInte)
shiftLeft (int n);
shiftRight (int n)

Comparison:

compareTo (BigInteger val)
equals (Object obj)

Others:

gcd (BigInteger val);
isProbablePrime (int certainty)
nextProbablePrime()

• BigDecimal

BigDecimal is a class that provides support for arbitary-precision floating point numbers. It is used for precision calculation especially important in financial and scientific application where accuracy is crucial.

Key-Points

Arbitrary Precision - Can represent decimal numbers of any size and precision.

Immutable - Operations on 'BigDecimal' returns new BigDecimal objects the orignal object remains unchanged.

Rounding Modes - Provide various rounding modes such as HALF_UP, HALF_DOWN, CEILING, FLOOR etc.

```
Object
↳ java.lang.Number
    ↳ java.math.BigDecimal
      extends number &
      implements comparable
```

Constructors

BigDecimal bd = new BigDecimal(String val);
Constructs a BigDecimal from a string representation

BigDecimal bd = new BigDecimal(BigInteger val);
Constructs a BigDecimal from a BigInteger.

BigDecimal bd = new BigDecimal(double val);
Constructs a BigDecimal from a double (note use: BigDecimal. valueOf(double) to avoid precision issues.)

Big Decimal bd = new BigDecimal(int val);
Constructs a BigDecimal from integer.

Methods
Arithmetic Operations.
add (BigDecimal val)
subtract (BigDecimal val)
multiply (BigDecimal val)
divide (BigDecimal val)
reminder (BigDecimal val)
pow (BigDecimal val)
Comparison:
compareTo (BigDecimal val)
equals (Object obj)
Scale / Precision:
setScale (int newScale)
precision()
Rounding:
divide (BigDecimal divisor, int scale, RoundingMode roundingMode)
setScale(int newScale, RoundingMode roundingMode);

Rounding Modes
RoundingMode.UP - Rounds away from zero
RoundingMode.DOWN - Rounds towards zero
RoundingMode.CEILING - Round's toward positive infinity
RoundingMode.FLOOR - Rounds toward negative infinity
RoundingMode.HALF_UP - Rounds towards nearest neighbor unless both neighbor are equivalent/dista
In which case rounds up.
RoundingMode.HALF_DOWN - Rounds towards nearest neighbor unless both neighbor equidistant
In which case rounds down
RoundingMode.HALF_EVEN Rounds towards nearest neighbor unless both neighbor are equidistant
In which case rounds toward even neighb