Table of Contents

Java in a Nutshell, 5th Edition By David Flanagan ISBN: 0596007736 Publisher: O'Reilly
Print Publication Date: 2005/03/01

Chapter 11. java.math

Package java.math

Java 1.1

The `java.math` package contains the `BigInteger` class for arbitrary-precision integer arithmetic, which is useful for cryptography. It also contains the `BigDecimal` class for arbitrary precision decimal floating-point arithmetic, which is useful for financial applications that need to be careful about rounding errors. The `BigDecimal` class is greatly enhanced in Java 5.0 and is accompanied by the new types `MathContext` and `RoundingMode`.

Enumerated Types

```
public enum RoundingMode;
```

Classes

```
public class BigDecimal extends Number implements Comparable<BigDecimal>;
public class BigInteger extends Number implements Comparable<BigInteger>;
public final class MathContext implements Serializable;
```

BigDecimal java.math

Java 1.1 *serializable comparable*

This subclass of `java.lang.Number` represents a floating-point number of arbitrary size and precision. Because it uses a decimal rather than binary floating-point representation, it is not subject to the rounding errors that the `float` and `double` types are. This makes `BigDecimal` well-suited to financial and similar applications.

`BigDecimal` provides `add( )` , `subtract( )`, `multiply( )`, and `divide( )` methods to support basic arithmetic. In Java 5.0, this class has been expanded to define many more methods, including `pow( )` for exponentiation. Many of the new methods use a `MathContext` to specify the desired precision of the result and the `RoundingMode` to be used to achieve that precision.

Java in a Nutshell, 5th Edition By David Flanagan ISBN: 0596007736 Publisher: O'Reilly
Print Publication Date: 2005/03/01

`BigDecimal` extends `Number` and implements the `Comparable` interface. The `compareTo( )` method compares the value of two `BigDecimal` objects and returns -1, 0, or 1 to indicate the result of the comparison. Use this method in place of the <, <=, >, and >= operators that you'd use with `float` and `double` values.

A `BigDecimal` object is represented as an integer of arbitrary size and an integer scale that specifies the number of decimal places in the value. When working with `BigDecimal` values, you can explicitly specify the precision (i.e., the number of decimal places) you are interested in. Also, whenever a `BigDecimal` method can discard precision (e.g., in a division operation), you are required to specify what sort of rounding should be performed on the digit to the left of the discarded digit or digits. The eight constants defined by this class specify the available rounding modes. In Java 5.0, however, the preferred way to specify a rounding mode is with the enumerated type `RoundingMode`.

Figure 11-1. java.math.BigDecimal


```
public class BigDecimal extends Number implements Comparable<BigDecimal> {
// Public Constructors
     public BigDecimal(BigInteger val);
5.0  public BigDecimal(int val);
5.0  public BigDecimal(long val);
     public BigDecimal(String val);
5.0  public BigDecimal(char[ ] in);
     public BigDecimal(double val);
5.0  public BigDecimal(long val, MathContext mc);
5.0  public BigDecimal(int val, MathContext mc);
5.0  public BigDecimal(double val, MathContext mc);
5.0  public BigDecimal(String val, MathContext mc);
5.0  public BigDecimal(char[ ] in, MathContext mc);
     public BigDecimal(BigInteger unscaledVal, int scale);
5.0  public BigDecimal(BigInteger val, MathContext mc);
5.0  public BigDecimal(BigInteger unscaledVal, int scale, MathContext mc);
5.0  public BigDecimal(char[ ] in, int offset, int len);
5.0  public BigDecimal(char[ ] in, int offset, int len, MathContext mc);
// Public Constants
5.0  public static final BigDecimal ONE;
     public static final int ROUND_CEILING;                              =2
     public static final int ROUND_DOWN;   =1
     public static final int ROUND_FLOOR;  =3
     public static final int ROUND_HALF_DOWN;                            =5
     public static final int ROUND_HALF_EVEN;                            =6
     public static final int ROUND_HALF_UP;                              =4
     public static final int ROUND_UNNECESSARY;                          =7
     public static final int ROUND_UP;     =0
5.0  public static final BigDecimal TEN;
5.0  public static final BigDecimal ZERO;
// Public Class Methods
     public static BigDecimal valueOf(long val);
5.0  public static BigDecimal valueOf(double val);
     public static BigDecimal valueOf(long unscaledVal, int scale);
// Public Instance Methods
     public BigDecimal abs( );
5.0  public BigDecimal abs(MathContext mc);
     public BigDecimal add(BigDecimal augend);
5.0  public BigDecimal add(BigDecimal augend, MathContext mc);
5.0  public byte byteValueExact( );
```


Java in a Nutshell, 5th Edition By David Flanagan ISBN: 0596007736 Publisher: O'Reilly
Print Publication Date: 2005/03/01

```
     public int compareTo(BigDecimal val);                Implements:Comparable
5.0  public BigDecimal divide(BigDecimal divisor);
     public BigDecimal divide(BigDecimal divisor, int roundingMode);
5.0  public BigDecimal divide(BigDecimal divisor, RoundingMode roundingMode);
5.0  public BigDecimal divide(BigDecimal divisor, MathContext mc);
     public BigDecimal divide(BigDecimal divisor, int scale, int roundingMode);
5.0  public BigDecimal divide(BigDecimal divisor, int scale, RoundingMode roundingMode);
5.0  public BigDecimal[ ] divideAndRemainder(BigDecimal divisor);
5.0  public BigDecimal[ ] divideAndRemainder(BigDecimal divisor, MathContext mc);
5.0  public BigDecimal divideToIntegralValue(BigDecimal divisor);
5.0  public BigDecimal divideToIntegralValue(BigDecimal divisor, MathContext mc);
5.0  public int intValueExact( );
5.0  public long longValueExact( );
     public BigDecimal max(BigDecimal val);
     public BigDecimal min(BigDecimal val);
     public BigDecimal movePointLeft(int n);
     public BigDecimal movePointRight(int n);
     public BigDecimal multiply(BigDecimal multiplicand);
5.0  public BigDecimal multiply(BigDecimal multiplicand, MathContext mc);
     public BigDecimal negate( );
5.0  public BigDecimal negate(MathContext mc);
5.0  public BigDecimal plus( );
5.0  public BigDecimal plus(MathContext mc);
5.0  public BigDecimal pow(int n);
5.0  public BigDecimal pow(int n, MathContext mc);
5.0  public int precision( );
5.0  public BigDecimal remainder(BigDecimal divisor);
5.0  public BigDecimal remainder(BigDecimal divisor, MathContext mc);
5.0  public BigDecimal round(MathContext mc);
     public int scale( );
5.0  public BigDecimal scaleByPowerOfTen(int n);
     public BigDecimal setScale(int newScale);
     public BigDecimal setScale(int newScale, int roundingMode);
5.0  public BigDecimal setScale(int newScale, RoundingMode roundingMode);
5.0  public short shortValueExact( );
     public int signum( );
5.0  public BigDecimal stripTrailingZeros( );
     public BigDecimal subtract(BigDecimal subtrahend);
5.0  public BigDecimal subtract(BigDecimal subtrahend, MathContext mc);
     public BigInteger toBigInteger( );
5.0  public BigInteger toBigIntegerExact( );
5.0  public String toEngineeringString( );
5.0  public String toPlainString( );
5.0  public BigDecimal ulp( );
1.2  public BigInteger unscaledValue( );
// Methods Implementing Comparable
     public int compareTo(BigDecimal val);
// Public Methods Overriding Number
     public double doubleValue( );
     public float floatValue( );
     public int intValue( );
     public long longValue( );
// Public Methods Overriding Object
     public boolean equals(Object x);
     public int hashCode( );
     public String toString( );
}
```

Passed To

javax.xml.datatype.DatatypeFactory.{newDuration( ), newXMLGregorianCalendar( ),newXMLGregorianCalendarTime( )}, javax.xml.datatype.Duration.multiply( ), javax.xml.datatype.XMLGregorianCalendar.{setFractionalSecond( ), setTime( )}

Java in a Nutshell, 5th Edition By David Flanagan ISBN: 0596007736 Publisher: O'Reilly
Print Publication Date: 2005/03/01

Returned By

`java.util.Scanner.nextBigDecimal( )`,
`javax.xml.datatype.XMLGregorianCalendar.getFractionalSecond( )`

BigInteger java.math

Java 1.1 ***serializable comparable***

This subclass of `java.lang.Number` represents integers that can be arbitrarily large (i.e., integers that are not limited to the 64 bits available with the `long` data type). `BigInteger` defines methods that duplicate the functionality of the standard Java arithmetic and bit-manipulation operators. The `compareTo( )` method compares two `BigInteger` objects and returns -1, 0, or 1 to indicate the result of the comparison. The `gcd( )` ,`modPow( )`,`modInverse( )`, and `isProbablePrime( )` methods perform advanced operations and are used primarily in cryptographic and related algorithms.

Figure 11-2. java.math.BigInteger


```
public class BigInteger extends Number implements Comparable<BigInteger> {
// Public Constructors
     public BigInteger(byte[ ] val);
     public BigInteger(String val);
     public BigInteger(String val, int radix);
     public BigInteger(int signum, byte[ ] magnitude);
     public BigInteger(int numBits, java.util.Random rnd);
     public BigInteger(int bitLength, int certainty, java.util.Random rnd);
// Public Constants
1.2  public static final BigInteger ONE;
5.0  public static final BigInteger TEN;
1.2  public static final BigInteger ZERO;
// Public Class Methods
1.4  public static BigInteger probablePrime(int bitLength, java.util.Random rnd);
     public static BigInteger valueOf(long val);
// Public Instance Methods
     public BigInteger abs( );
     public BigInteger add(BigInteger val);
     public BigInteger and(BigInteger val);
     public BigInteger andNot(BigInteger val);
     public int bitCount( );
     public int bitLength( );
     public BigInteger clearBit(int n);
     public int compareTo(BigInteger val);               Implements:Comparable
     public BigInteger divide(BigInteger val);
     public BigInteger[ ] divideAndRemainder(BigInteger val);
     public BigInteger flipBit(int n);
     public BigInteger gcd(BigInteger val);
     public int getLowestSetBit( );
     public boolean isProbablePrime(int certainty);
     public BigInteger max(BigInteger val);
     public BigInteger min(BigInteger val);
     public BigInteger mod(BigInteger m);
     public BigInteger modInverse(BigInteger m);
```


Java in a Nutshell, 5th Edition By David Flanagan ISBN: 0596007736 Publisher: O'Reilly
Print Publication Date: 2005/03/01

```
    public BigInteger modPow(BigInteger exponent, BigInteger m);
    public BigInteger multiply(BigInteger val);
    public BigInteger negate( );
5.0 public BigInteger nextProbablePrime( );
    public BigInteger not( );
    public BigInteger or(BigInteger val);
    public BigInteger pow(int exponent);
    public BigInteger remainder(BigInteger val);
    public BigInteger setBit(int n);
    public BigInteger shiftLeft(int n);
    public BigInteger shiftRight(int n);
    public int signum( );
    public BigInteger subtract(BigInteger val);
    public boolean testBit(int n);
    public byte[ ] toByteArray( );
    public String toString(int radix);
    public BigInteger xor(BigInteger val);
// Methods Implementing Comparable
    public int compareTo(BigInteger val);
// Public Methods Overriding Number
    public double doubleValue( );
    public float floatValue( );
    public int intValue( );
    public long longValue( );
// Public Methods Overriding Object
    public boolean equals(Object x);
    public int hashCode( );
    public String toString( );
}
```

Passed To

Too many methods to list.

Returned By

Too many methods to list.

Type Of

`java.security.spec.RSAKeyGenParameterSpec.{F0,F4}`

MathContext java.math

Java 5.0 ***serializable***

This simple class represents a precision (number of significant digits) and a `RoundingMode` to be used in `BigDecimal` arithmetic. The constants are predefined `MathContext` objects that can be used to select unlimited precision arithmetic or to select specific operating modes that match decimal floating-point modes defined by the IEEE 754R standard.

Figure 11-3. java.math.MathContext


```
public final class MathContext implements Serializable {
// Public Constructors
    public MathContext(int setPrecision);
    public MathContext(String val);
```


Java in a Nutshell, 5th Edition By David Flanagan ISBN: 0596007736 Publisher: O'Reilly
Print Publication Date: 2005/03/01

```
     public MathContext(int setPrecision, RoundingMode setRoundingMode);
// Public Constants
     public static final MathContext DECIMAL128;
     public static final MathContext DECIMAL32;
     public static final MathContext DECIMAL64;
     public static final MathContext UNLIMITED;
// Public Instance Methods
     public int getPrecision( );
     public RoundingMode getRoundingMode( );
// Public Methods Overriding Object
     public boolean equals(Object x);
     public int hashCode( );
     public String toString( );
}
```

Passed To

Too many methods to list.

RoundingMode java.math

Java 5.0 ***serializable comparable enum***

The constants defined by this enumerated type represent possible ways of rounding numbers. `UP` and `DOWN` specify rounding away from zero or toward zero. `CEILING` and `FLOOR` represent rounding toward positive infinity and negative infinity. `HALF_UP`, `HALF_DOWN`, and `HALF_EVEN` all round toward the nearest value and differ only in what they do when two values are equidistant. In this case, they round up, down, or to the "even" neighbor. `UNNECESSARY` is a special rounding mode that serves as an assertion that an arithmetic operation will have an exact result and that rounding is not needed. If this assertion fails—that is, if the operation does require rounding—an `ArithmeticException` is thrown.

Figure 11-4. java.math.RoundingMode


```
public enum RoundingMode {
// Enumerated Constants
     UP,
     DOWN,
     CEILING,
     FLOOR,
     HALF_UP,
     HALF_DOWN,
     HALF_EVEN,
     UNNECESSARY;
// Public Class Methods
     public static RoundingMode valueOf(int rm);
     public static RoundingMode valueOf(String name);
     public static final RoundingMode[ ] values( );
}
```


Java in a Nutshell, 5th Edition By David Flanagan ISBN: 0596007736 Publisher: O'Reilly
Print Publication Date: 2005/03/01

Passed To

`BigDecimal.{divide( ),setScale( )},MathContext.MathContext( )`

Returned By

`MathContext.getRoundingMode( )`

Java in a Nutshell, 5th Edition By David Flanagan ISBN: 0596007736 Publisher: O'Reilly
Print Publication Date: 2005/03/01