

Назад

Меню

Вперед

## Класс Big Decimal

Класс BigDecimal расположен в пакете `java.math`.

Каждый объект этого класса хранит два целочисленных значения: мантиссу вещественного числа в виде объекта класса `BigInteger`, и неотрицательный десятичный порядок числа типа `int`.

Например, для числа 76.34862 будет храниться мантисса 7 634 862 в объекте класса `BigInteger`, и порядок 5 как целое число типа `int`. Таким образом, мантисса может содержать любое количество цифр, а порядок ограничен значением константы `integer.MAX_VALUE`. Результат операции над объектами класса `BigDecimal` округляется по одному из восьми правил, определяемых следующими статическими целыми константами:

`ROUND_CEILING` — округление в сторону большего целого;

`ROUND_DOWN` — округление к нулю, к меньшему по модулю целому значению;

`ROUND_FLOOR` — округление к меньшему целому;

`ROUND_HALF_DOWN` — округление к ближайшему целому, среднее значение округляется к меньшему целому;

`ROUND_HALF_EVEN` — округление к ближайшему целому, среднее значение округляется к четному числу;

Work and  
Collaborate  
Where  
You Want

Work and  
Collaborate  
Where  
You Want

ROUND\_HALF\_UP — округление к ближайшему целому, среднее значение округляется к большему целому;  
ROUND\_UNNECESSARY — предполагается, что результат будет целым, и округление не понадобится;

- ROUND\_UP — округление от нуля, к большему по модулю целому значению.

В классе `BigDecimal` четыре конструктора:

- `BigDecimal (BigInteger bi)` — объект будет хранить большое целое `bi`, порядок равен нулю;
- `BigDecimal (BigInteger mantissa, int scale)` — задается мантиса `mantissa` и неотрицательный порядок `scale` объекта; если порядок `scale` отрицателен, возникает исключительная ситуация;
- `BigDecimal (double d)` — объект будет содержать вещественное число удвоенной точности `d`; если значение `d` бесконечно или `NaN`, то возникает исключительная ситуация;
- `BigDecimal (String val)` — число задается строкой символов `val`, которая должна содержать запись числа по правилам языка Java.

При использовании третьего из перечисленных конструкторов возникает неприятная особенность, отмеченная в документации. Поскольку вещественное число при переводе в двоичную форму представляется, как правило, бесконечной двоичной дробью, то при создании объекта, например, `BigDecimal(0.1)`, мантисса, хранящаяся в объекте, окажется очень большой. Она показана на рис. 4.5. Но при создании такого же объекта четвертым конструктором, `BigDecimal ("0.1")`, мантисса будет равна просто 1.

В Классе переопределены методы `doubleValue()`, `floatValue()`, `intValue()`, `longValue()`.

Большинство методов этого класса моделируют операции с вещественными числами. Они возвращают объект класса `BigDecimal`. Здесь буква `x` обозначает объект класса `BigDecimal`, буква `n` — целое значение типа `int`, буква `r` — способ округления, одну из восьми перечисленных выше констант:

`abs()` — абсолютное значение объекта `this`;

`add(x)` — операция `this + x`;

`divide(x, r)` — операция `this / x` с округлением по способу `r` ;

`divide(x, n, r)` — операция `this / x` с изменением порядка и округлением по способу `r` ;

`max(x)` — наибольшее из `this` и `x` ;

`min(x)` — наименьшее из `this` и `x` ;

`movePointLeft(n)` — сдвиг влево на `n` разрядов;

`movePointRight(n)` — сдвиг вправо на `n` разрядов;

`multiply(x)` — операция `this * x` ;

`negate()` — возвращает объект с обратным знаком;

`scale()` — возвращает порядок числа;

`setscale(n)` — устанавливает новый порядок `n` ;

`setscale(n, r)` — устанавливает новый порядок `n` и округляет число при необходимости по способу `r` ;

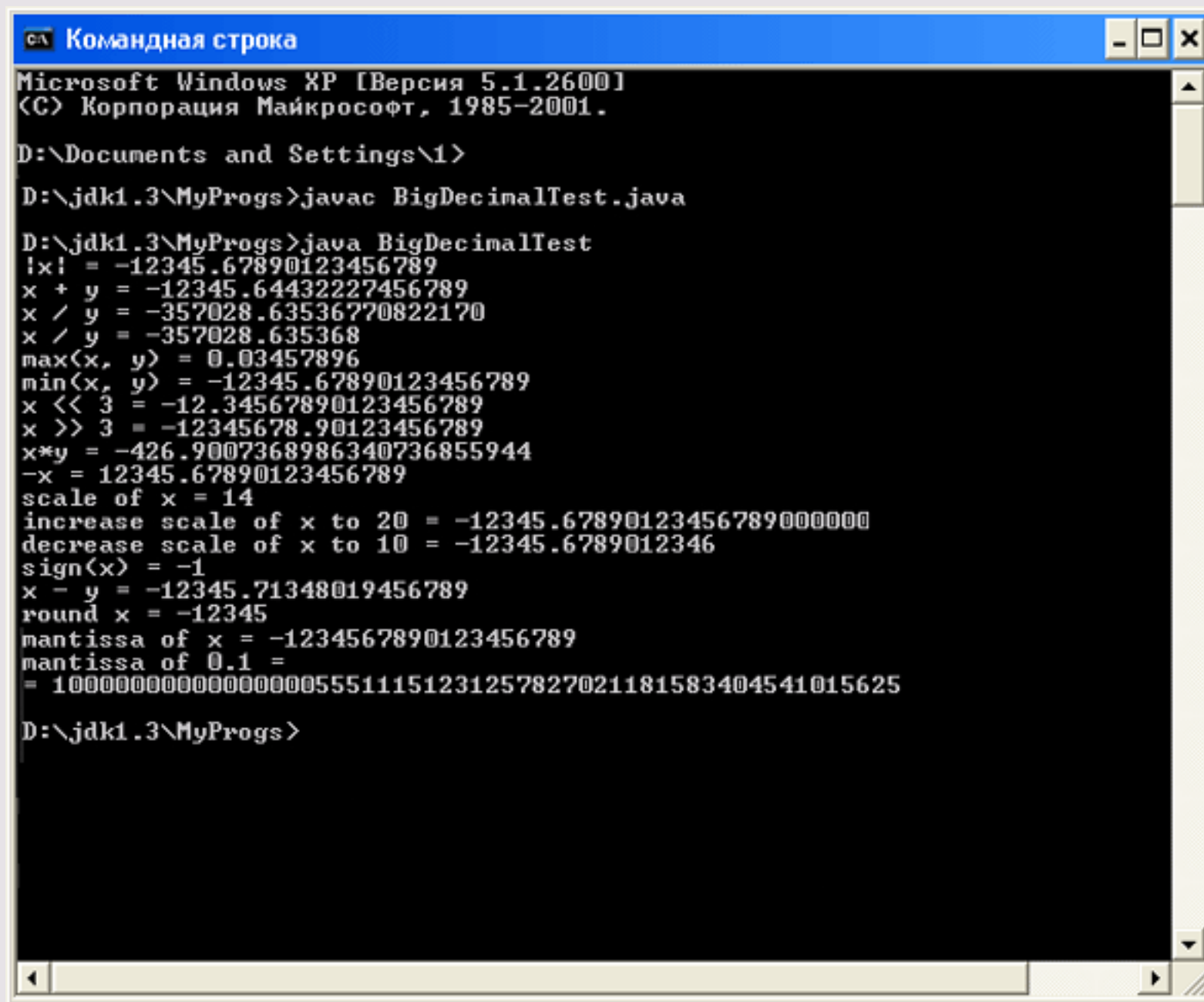
`signum` — знак числа, хранящегося в объекте;

`subtract(x)` — операция `this - x` ;

`toBigInteger()` — округление числа, хранящегося в объекте;

`unscaledvalue()` — возвращает мантиссу числа.

Листинг 4.4 показывает примеры использования этих методов, а рис. 4.5 — вывод результатов.



```
Командная строка
Microsoft Windows XP [Версия 5.1.2600]
(C) Корпорация Майкрософт, 1985-2001.

D:\Documents and Settings\1>
D:\jdk1.3\MyProgs>javac BigDecimalTest.java

D:\jdk1.3\MyProgs>java BigDecimalTest
!x! = -12345.67890123456789
x + y = -12345.64432227456789
x / y = -357028.63536770822170
x \ y = -357028.635368
max(x, y) = 0.03457896
min(x, y) = -12345.67890123456789
x << 3 = -12.34567890123456789
x >> 3 = -12345678.90123456789
x*y = -426.9007368986340736855944
-x = 12345.67890123456789
scale of x = 14
increase scale of x to 20 = -12345.67890123456789000000
decrease scale of x to 10 = -12345.6789012346
sign(x) = -1
x - y = -12345.71348019456789
round x = -12345
mantissa of x = -1234567890123456789
mantissa of 0.1 =
= 1000000000000000000055511151231257827021181583404541015625

D:\jdk1.3\MyProgs>
```

**Рис. 4.5.** Методы класса `BigDecimal` в программе `BigDecimalTest`

**Листинг 4.4.** Методы класса `BigDecimal` в программе `BigDecimalTest`

```
import java.math.*;

class BigDecimalTest{

    public static void main,( String [] args) {

        BigDecimal x = new BigDecimal("-12345.67890123456789");

        BigDecimal y = new BigDecimal("345.7896e-4");

        BigDecimal z = new BigDecimal(new BigInteger("123456789"),8);

        System.out.println("|x| = " + x.abs());

        System.out.println("x + y = " + x.add(y));

        System.out.println("x / y = " + x.divide(y, BigDecimal.ROUND__DOWN));

        System.out.println("x / y = " +

            x.divide(y, 6, BigDecimal.ROUND_HALF_EVEN));

        System.out.println("max(x, y) = " + x.max(y));

        System.out.println("min(x, y) = " + x.min(y));

        System.out.println("x « 3 = " * x.movePointLeft(3));

        System.out.println("x » 3 = " + x.mpvePQintRight(3));

        System.out.println("x * y = " + x.multiply(y));

        System.out.println("-x = " + x.negate());
```

```

System.out.println("scale of x = " + x.scale());

System.out.println("increase scale of x to 20 = " + x.setScale(20));

System.out.println("decrease scale of x to 10 = " +

    x.setScale (10, BigDecimal.ROUND_HALF__UP)) ;

System.out.println("sign(x) = " + x.signum());

System.out.println("x - y = " + x.subtract(y));

System.out.println("round x = " + x.toBigInteger());

System.out.println("mantissa of x = " + x.unscaledValue());

System.out.println("mantissa of 0.1 =\n= " +

    new BigDecimal(0.1).unscaledValue()); } }

```

Приведем еще один пример. Напишем простенький калькулятор, выполняющий четыре арифметических действий с числами любой величины. Он работает из командной строки. Программа представлена в листинге 4.5, а примеры использования калькулятора — на рис. 4.6.

#### Листинг 4.5. Простейший калькулятор

```

import Java.math.*;

class Calc{

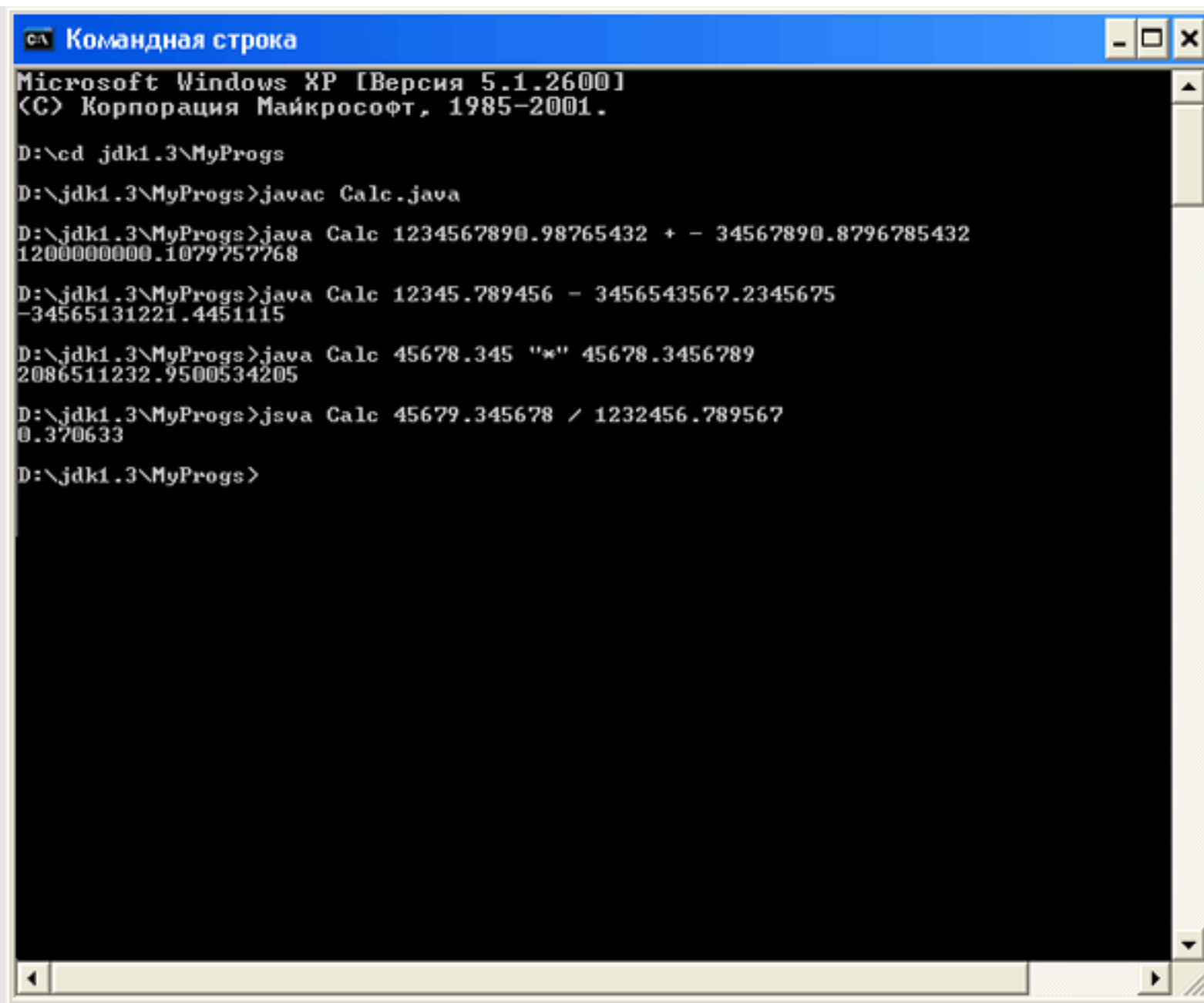
    public static void main(String[] args){

        if (args.length < 3){

```

```
System.err.println("Usage: Java Calc operand operator operand");  
  
return;  
  
}  
  
BigDecimal a = new BigDecimal(args[0]);  
  
BigDecimal b = new BigDecimal(args[2]);  
  
switch (args[1].charAt(0)){  
  
    case '+': System.out.println(a.add(b)); break;  
  
    case '-': System.out.println(a.subtract(b)); break;  
  
    case '*': System.out.println(a.multiply(b)); break;  
  
    case '/': System.out.println(a.divide(b,  
  
                                     BigDecimal.ROUND_HALF_EVEN)); break;  
  
    default : System.out.println("Invalid operator");  
  
}  
  
}  
  
}
```

Почему символ умножения — звездочка — заключен на рис. 4.6 в кавычки? "Юниксоидам" это понятно, а для других дадим краткое пояснение.



```
Командная строка
Microsoft Windows XP [Версия 5.1.2600]
(C) Корпорация Майкрософт, 1985-2001.

D:\>cd jdk1.3\MyProgs
D:\jdk1.3\MyProgs>javac Calc.java
D:\jdk1.3\MyProgs>java Calc 1234567890.98765432 + - 34567890.8796785432
12000000000.1079757768
D:\jdk1.3\MyProgs>java Calc 12345.789456 - 3456543567.2345675
-34565131221.4451115
D:\jdk1.3\MyProgs>java Calc 45678.345 "*" 45678.3456789
2086511232.9500534205
D:\jdk1.3\MyProgs>jsva Calc 45679.345678 / 1232456.789567
0.370633
D:\jdk1.3\MyProgs>
```

**Рис. 4.6.** Результаты работы калькулятора

Это особенность операционной системы, а не языка Java. Введенную с клавиатуры строку вначале



просматривает командная оболочка (shell) операционной системы, а звездочка для нее — указание подставить на это место все имена файлов из текущего каталога. Оболочка сделает это, и интерпретатор Java получит от нее длинную строку, в которой вместо звездочки стоят имена файлов через пробел.

Звездочка в кавычках понимается командной оболочкой как обычный символ. Командная оболочка снимает кавычки и передает интерпретатору Java то, что надо.

Назад Меню Вперед



Реклама от Google

▶ [Java coding](#)

▶ [Java double](#)

▶ [Math java](#)

▶ [Python java](#)

Copyright © [Realcoding.NET](#) 2003-2007. При перепечатке материалов ссылка на автора материала обязательна.

Сообщить об ошибке или написать письмо администрации [через форму контактов](#).

4 113  
2 925  
1 059

