链滴

java.math.BigDecimal

作者：helly

原文链接：https://ld246.com/article/1501296009951

来源网站：链滴

一、IEEE754标准浮点格式

1. 格式：数符 + 阶码 + 尾数

①. 32位（单精度浮点数）： 1 + 8 + 23

数符是尾数的符号位；

阶码移码表示，但偏置量为127，不是128（2^7）；

尾数用原码表示；

隐含约定尾数最高位20 ，即1。

②. 64位（双精度浮点数）： 1 + 11 + 52

二、IEEE754的四种舍入方式

1. **最近舍入（Round to Nearest）**（**默认**的舍入方式，也称为向偶舍入，java浮点数的舍入方式）

Round(0.5) = 0; Round(1.5) = 2; Round(2.5) = 2;

2. **向0舍入**

(int) 1.3 = 1，(int) -1.3 = -1;

3. **向下舍入**

floor(1.3) = 1，floor(-1.3) = -2

4. **向上舍入**

ceil(1.3) = 2。Ceil(-1.3) = -1;

三、为什么浮点计算不准确

2进制的小数无法精确的表达某些10进制小数。

四、BigDecimal

BigDecimal可以表示任意精度的小数，并对它们进行计算。

由于 BigDecimal 对象是不可变的，加、减、乘、除中的每一个都会产生新的BigDecimal 对象。

1. 构造方法

```
bigDecimal = new BigDecimal(new char[]{'1','2'});//char[]
bigDecimal = new BigDecimal(1.2);//double
bigDecimal = new BigDecimal(5);//int
bigDecimal = new BigDecimal(5L);//long
bigDecimal = new BigDecimal("1.2");//String
```

2. 加减乘除运算

```
BigDecimal a = new BigDecimal("5.5");
BigDecimal b = new BigDecimal("1.2");
BigDecimal c = null;
c = a.add(b);      //c = a + b
c = a.subtract(b); //c = a - b
c = a.multiply(b); //c = a * b
c = a.divide(b);   //c = a / b
```

原文链接：java.math.BigDecimal

3. 小数舍入

setScale(newScale, roundingMode) // 设置设置小数位数及取舍方式，newScale为小数位数

roundingMode有如下几种：

BigDecimal.ROUND_CEILING; // 向正无穷方向取整
BigDecimal.ROUND_DOWN; // 向0方向取整
BigDecimal.ROUND_FLOOR; // 向负无穷方向取整
BigDecimal.ROUND_HALF_DOWN; // 小数>0.5向非0方向进位，其他向0方向进位
BigDecimal.ROUND_HALF_EVEN; // 小数偶数同ROUND_HALF_DOWN，奇数同ROUND_HALF_U

BigDecimal.ROUND_HALF_UP; // 小数>=0.5向非0方向进位，其他向0方向进位
BigDecimal.ROUND_UNNECESSARY;//断言所请求的操作具有精确的结果，因此不需要舍入。 如在产生不精确结果的操作上指定此舍入模式，则抛出ArithmeticException
BigDecimal.ROUND_UP; //向非0方向取整

4. 补充

首选String类型的构造函数（double类型的误差导致BigDecimal接收到的值会有误差）

除法操作需要设置小数位数（因为如果结果出现无限小数，会抛出异常）

不要使用equals判断是否相等（比较两个BigDecimal数值是否相等，应使用compareTo() 方法而不equals()方法。）

参考：

https://zhuanlan.zhihu.com/p/24997415