链滴

BigDecimal 常用锦囊

作者：sirwsl

原文链接：https://ld246.com/article/1630570306917

来源网站：链滴

知识点

CompareTo

算式	使用	compareTo 值
a == b	a.compareTo(b)	0
a > b	a.compareTo(b)	1
a < b	a.compareTo(b)	-1

推理：

例子	推荐code	不推荐code
a > b	a.compareTo(b) > 0	a.compareTo(b) == 1
a >= b	a.compareTo(b) >= 0	a.compareTo(b) > -1
a < b	a.compareTo(b) < 0	a.compareTo(b) == -1
a <= b	a.compareTo(b) <=0	a.compareTo(b) < 1
a == b	a.compareTo(b) == 0	a.compareTo(b) == 0

加减乘除

+-*/	表达式
加	add(BigDecimal)
减	subtract(BigDecimal)
乘	multiply(BigDecimal)
除	divide(BigDecimal)

equals

a 值	b 值	a.equals(b)值
BigDecimal(1000)	BigDecimal(1000)	ure
BigDecimal(1000)	BigDecimal("1000")	rue
BigDecimal(100.01)	BigDecimal("100.01")	alse
BigDecimal(100.01)	BigDecimal(100.0100)	rue
BigDecimal("100.01")	BigDecimal("100.0100")	alse

推理：

equals比较时候相等时候

采用double构建和小数位数无关，采用String构建与小数位数有关

double与String构建的BigDecimal，值一样但也是false

故事

就说说最近吧，最近上班经常用到了一个东西BigDecimal，然后对我我这不爱动脑的人来说就很要了，为什么这种说（接下来就尽情嘲笑我吧），我每次用到比较大小都去百度一次，有个哥们那个博怕被我访问了100来次，但是还是记不住，就是因为他用了上面我写了不推荐的那个写法，搞得我头眼花，今天咬咬牙，来写写，加深印象。

我每次用到BigDecimal的时候，经常要比较大小，这时候呢，我这脑子就不好用了，分不清谁大谁，当时我脑子大概是这样想的：compareTo等于1是左边大还是又边大，等于-1了，如果要大于等于么办。

搞事情（理论）

compareTo源码分析

```
public int compareTo(BigDecimal val) {
    //判断小数位数，区别出是否是字符串构造
    //如果是int或者long则为0，double 则scale是46，但如果是字符串，则有几位算几位
    if (scale == val.scale) {//相等说明是int或者long
```

```
        //xs ys 通过移位以及其他运算得出可用于计算的值
        long xs = intCompact;
        long ys = val.intCompact;
        if (xs != INFLATED && ys != INFLATED) //判断时候超过INFLATED  -2的63次方
            //这就是为什么返回的是 0 1 -1的原因
            return xs != ys ? ((xs > ys) ? 1 : -1) : 0;
    }
    //判断signum函数 具体signum函数是什么鬼自己google
    //具体为正负0:0，正：1，负：-1   NaN：NaN
    int xsign = this.signum();
    int ysign = val.signum();
    if (xsign != ysign)
        return (xsign > ysign) ? 1 : -1;
    if (xsign == 0)
        return 0;
    //忽略符号进行比较  无符号的compareTo
    int cmp = compareMagnitude(val);
    return (xsign > 0) ? cmp : -cmp;
}
```

equals源码分析

```
@Override
public boolean equals(Object x) {
    //常规查看类型
    if (!(x instanceof BigDecimal))
        return false;
    //将Object强转
    BigDecimal xDec = (BigDecimal) x;
    //对比内存地址
    if (x == this)
        return true;
    //对比小数位数，注意：为什么我们采用字符串构建和double构建，虽然小数位数看着一样，但是false的原因，double的scale有40多位，但是如果是字符串则有几位是几位
    if (scale != xDec.scale)
        return false;
    //移位运算计算可用于计算的紧凑值
    long s = this.intCompact;
    long xs = xDec.intCompact;

    if (s != INFLATED) {//-2的63次方
        if (xs == INFLATED)
            xs = compactValFor(xDec.intVal);//压缩数据值
        return xs == s;
    } else if (xs != INFLATED)
        return xs == compactValFor(this.intVal);

    return this.inflated().equals(xDec.inflated());//开始值比较
}
```

与Double比较

采用Double很严重一个问题就是精度问题

为什么在很多的项目中会采用BigDecimal而不采用double呢，或许你有时候也会遇到前端页面显示格会有8/9位小数，这就是因为精度的原因。举个例子当你计算钱或者货物。eg：有个老板有很多的金。现在金价1g算350，他本来有0.03吨（30kg），这个时候卖了 0.02吨（20kg），如果是Doubl的话他现在还剩多少？还剩下结果是0.009999999999998；哦豁，本来只是2位小数，现在好家伙几位。