BigDecimal 主要用来操作（大）浮点数，BigInteger 主要用来操作大整数（超过 long 类型）。
BigDecimal 的实现利用到了 BigInteger，不同的是 BigDecimal 加入了小数位的概念。

BigDecimal 的用处

《阿里巴巴Java开发手册》中提到：浮点数之间的等值判断，基本数据类型不能用==来比较，包装数据类型不能用 equals 来判断。 具体原理和浮点数的编码方式有关，这里就不多提了，我们下面直接上实例：

```
float a = 1.0f - 0.9f;
float b = 0.9f - 0.8f;
System.out.println(a);// 0.100000024
System.out.println(b);// 0.099999964
System.out.println(a == b);// false
```

具有基本数学知识的我们很清楚的知道输出并不是我们想要的结果（精度丢失），我们如何解决这个问题呢？一种很常用的方法是：使用使用 BigDecimal 来定义浮点数的值，再进行浮点数的运算操作。

```
BigDecimal a = new BigDecimal("1.0");
BigDecimal b = new BigDecimal("0.9");
BigDecimal c = new BigDecimal("0.8");
BigDecimal x = a.subtract(b);// 0.1
BigDecimal y = b.subtract(c);// 0.1
System.out.println(x.equals(y));// true
```

1.3.2. BigDecimal 的大小比较

[redacted] ：返回 -1 表示小于，0 表示 等于， 1表示 大于。

```
BigDecimal a = new BigDecimal("1.0");
BigDecimal b = new BigDecimal("0.9");
System.out.println(a.compareTo(b));// 1
```

1.3.3. BigDecimal 保留几位小数

通过 [redacted]方法设置保留几位小数以及保留规则。保留规则有挺多种，不需要记，IDEA会提示。

```
BigDecimal m = new BigDecimal("1.255433");
BigDecimal n = m.setScale(3,BigDecimal.ROUND_HALF_DOWN);
System.out.println(n);// 1.255
```

1.3.4. BigDecimal 的使用注意事项

注意：我们在使用BigDecimal时，为了防止精度丢失，推荐使用它的 BigDecimal(String) 构造方法来创建对象。《阿里巴巴Java开发手册》对

这部分内容也有提到如下图所示。

10.【强制】为了防止精度损失，禁止使用构造方法 BigDecimal(double)的方式把 double 值转化为 BigDecimal 对象。

说明：BigDecimal(double)存在精度损失风险，在精确计算或值比较的场景中可能会导致业务逻辑异常。如：BigDecimal g = new BigDecimal(0.1f); 实际的存储值为：0.10000000149

正例：优先推荐入参为 String 的构造方法，或使用 BigDecimal 的 valueOf 方法，此方法内部其实执行了 Double 的 toString，而 Double 的 toString 按 double 的实际能表达的精度对尾数进行了截断。


```
BigDecimal recommend1 = new BigDecimal("0.1");
BigDecimal recommend2 = BigDecimal.valueOf(0.1);
```

1.3.5. 总结

BigDecimal 主要用来操作（大）浮点数，BigInteger 主要用来操作大整数（超过 long 类型）。

BigDecimal 的实现利用到了 BigInteger，所不同的是 BigDecimal 加入了小数位的概念