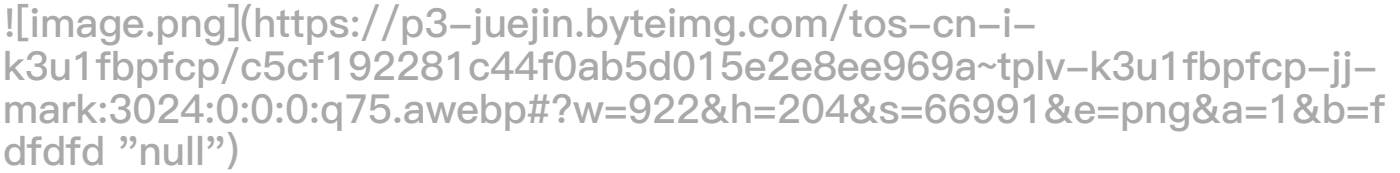
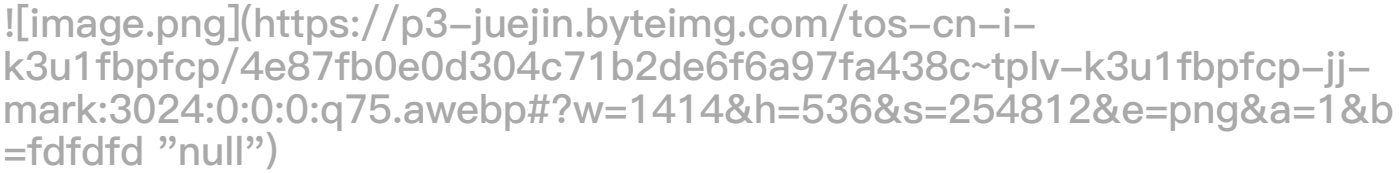
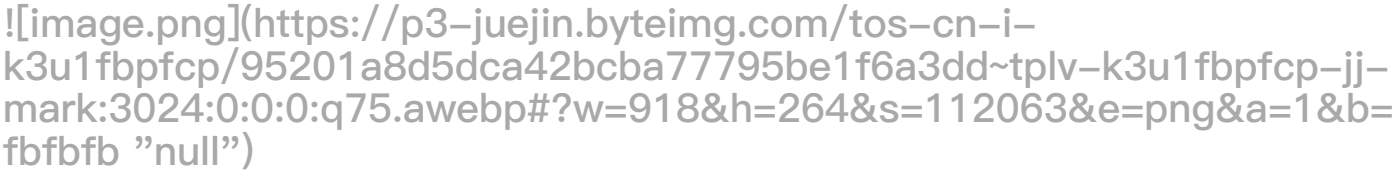
争论不休的一个话题：金额到底是用Long还是BigDecimal?

在网上一直流传着一个争论不休的话题：金额到底是用Long还是用BigDecimal？这个话题一出在哪都会引起异常无比激烈的讨论。。。。 比如说这个观点：算钱用BigDecimal是常识

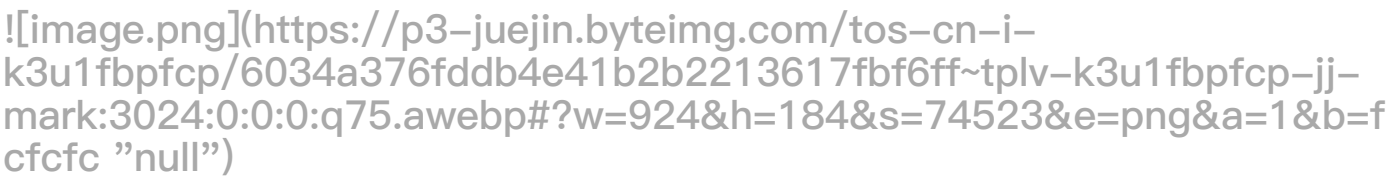




有支持用Long的，将金额的单位设计为分，然后乘以100，使用Long进行存储以及计算，这样不用担心小数点问题。



并且一些银行系统就会选择用Long



还有，最最最牛逼的万能大法：用String

cfcfc ”null”)

成年人不做选择题，Long跟BigDecimal都用。。。

![image.png](https://p3-juejin.byteimg.com/tos-cn-i-k3u1fbpfcp/dbf23f29f8ea43d38fbb35a6cdaecb9f~tplv-k3u1fbpfcp-jj-mark:3024:0:0:0:q75.awebp#?w=920&h=246&s=109540&e=png&a=1&b=fbfbfb ”null”)

image.png

还有一种就是封装一个金额的基类，对金额进行统一处理。

![image.png](https://p3-juejin.byteimg.com/tos-cn-i-k3u1fbpfcp/a81fb3f7055d43cb9a97e12b65859d0b~tplv-k3u1fbpfcp-jj-mark:3024:0:0:0:q75.awebp#?w=1456&h=358&s=170490&e=png&a=1&b=fdfdfd ”null”)

排除float和double

当然，对于金额，首先我们要排除的就是float和double。它们不适合用于精确的金融计算，因为`float`和`double`是基于IEEE 754标准的浮点数表示，它们无法精确地表示所有的十进制小数。这会导致在进行财务计算时出现舍入误差，这些误差可能会累积并导致不可预测的结果。

> 关于带精度的计算，我们不推荐使用float以及double，推荐使用BigDecimal，具体原因请参考：[聊一聊BigDecimal使用时的陷阱](http://cxyroad.com/ ”https://www.coderacademy.online/article/javabigdecimal.html”)

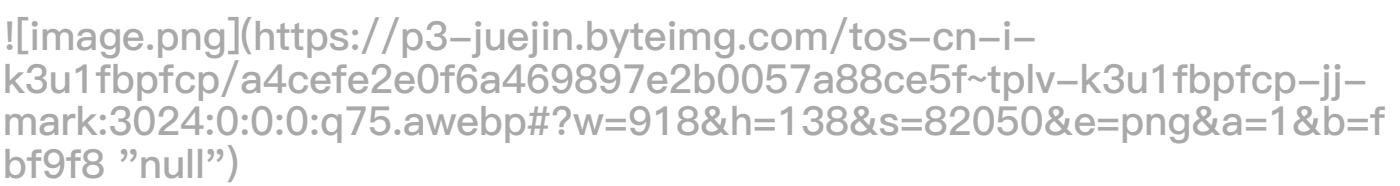
选择Long

`Long`类型在Java中用于存储64位整数。它的主要优点是速度快，因为整数运算在CPU层面是非常高效的。另外，`Long`类型也占用较少的内存，并且整数类型(`BIGINT`)在数据库中占用较少的存储空间。

但是`Long`类型在处理金额时有几个明显的缺点：

1. 1. **精度问题**：`Long`只能存储整数，无法直接表示小数。使用`Long`来表示以分为单位的金额（例如，100表示1元），此时就会失去小数的精度。即使使用某种方式来表示小数（例如，乘以100或10000），也会遇到舍入误差的问题。并且这种计算方式也会增加计算的复杂度。
2. 2. **浮点数问题**：虽然这不是直接使用`Long`的问题，但如果你尝试将`Long`与浮点数（如`double`或`float`）进行转换以进行计算（比如汇率计算等），还是会遇到浮点数精度问题，这可能导致在财务计算中出现不可接受的误差。

在阿里巴巴的开发手册中建议使用Long。



但是在一些金融系统当中，对小数位要求比较高的，比如精确到小数点后6位，那么我们使用Long进行存储，每次在计算时都要除以或者乘以1000000，那么计算的开销就很大了。

并且，如果在需求确认时，我们无法知道金额要求的小数位，那我们使用Long也是不行的，我们并不知道需要乘以或者除以多少个0。

选择BigDecimal

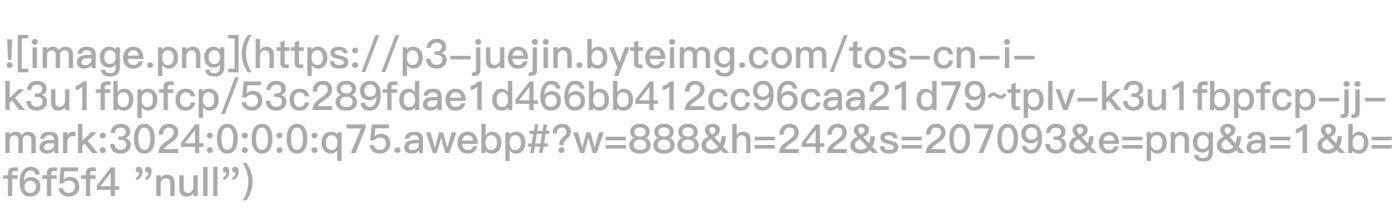
`BigDecimal`是Java提供的一个类，用于任意精度的算术运算。它的主要优点是提供了高精度的计算，这对于金融和货币计算来说是非常重要的。`BigDecimal`可以表示任意大小的正数、负数或零，并可以精确控制舍入行为。并且在数据库中存储时也有对应的类型进行匹配，比如MySQL的`DECIMAL`类型提供了精确的数值存储，可以匹配`BigDecimal`的精度。

但是`BigDecimal`也有一些缺点：

1. 1. **性能**：与`Long`相比，`BigDecimal`的性能较差。因为它的运算需要

更多的内存和CPU时间。
2. 2. **复杂性**：使用`BigDecimal`进行运算比使用`Long`或基本数据类型更复杂。你需要考虑舍入模式、精度等因素。
3. 3. 在数据库中需要更多的存储空间来存储小数部分。

而在Mysql的开发手册中，建议金额需要进行小数位计算时，存储要使用Decimal，否则我们要将金额乘以对应小数位的倍数变成BIGINT进行存储。



总结
--

基于上述对`Long`和`BigDecimal`的优缺点分析，我们可以得出以下结论：

在金额计算层面，即代码实现中，推荐使用`BigDecimal`进行所有与金额相关的计算。`BigDecimal`提供了高精度的数值运算，能够确保金额计算的精确性，避免了因浮点数精度问题导致的财务误差。使用`BigDecimal`可以简化代码逻辑，减少因处理精度问题而引入的复杂性。

而在数据库存储方面，我们需要根据具体需求进行权衡。如果业务需求已经明确金额只需精确到分（如某些国家/地区的货币最小单位为分），并且我们确信不会涉及到需要更高精度的小数计算，那么可以使用`Long`类型进行存储，将金额转换为最小货币单位（如分）进行存储。这样可以节省存储空间并提高查询性能。

但是如果业务需求中金额的小数位数不确定，或者可能涉及多位小数的计算（如国际货币交易等），那么最好使用`DECIMAL`或`NUMERIC`类型进行存储。这些类型提供了精确的数值存储，可以确保数据库中的数据与应用程序中的`BigDecimal`对象保持一致，避免数据转换过程中可能引入的精度损失。

本文已收录于我的个人博客：[码农Academy的博客，专注分享Java技术干货，包括Java基础、Spring Boot、Spring Cloud、Mysql、Redis、Elasticsearch、中间件、架构设计、面试题、程序员攻略等](http://cxyroad.com/ ”https://www.coderacademy.online/”)

原文链接: https://juejin.cn/post/7358670107902984229