军规一：【避免在程序中使用魔鬼数字，必须用有意义的常量来标识。】

说明：是否是魔鬼数字要基于容易阅读和便于全局替换的原则。0、1作为某种专业领域物理量枚举数值时必须定义常量，严禁出现类似NUMBER\_ZERO的“魔鬼常量”。

军规二：【明确方法的功能，一个方法仅完成一个功能。】

说明：方法功能太多，会增加方法的复杂度和依赖关系，不利于程序阅读和将来的持续维护，无论是方法还是类设计都应符合单一职责原则。

军规三：【方法参数不能超过5个】

说明：参数太多影响代码阅读和使用，为减少参数，首先要考虑这些参数的合理性，保持方法功能单一、优化方法设计，如果参数确实无法减少，可以将多个参数封装成一个类（对象），同时考虑在新的类（对象）中增加相应的行为，以期更符合OOP。

军规四：【方法调用尽量不要返回null，取而代之以抛出异常，或是返回特例对象（SPECIAL CASE object，SPECIAL CASE PATTERN）；对于以集合或数组类型作为返回值的方法，取而代之以空集合或0长度数组。】

说明：返回null会增加不必要的空指针判断，遗漏判断也会导致严重的NullPointerException错误。

军规五：【在进行数据库操作或IO操作时，必须确保资源在使用完毕后得到释放，并且必须确保释放操作在finally中进行。】

说明：数据库操作、IO操作等需要关闭对象必须在try -catch-finally 的finally中close()，如果有多个IO对象需要关闭，需要分别对每个对象的close()方法进行try-catch,防止一个IO对象关闭失败其他IO对象都未关闭。推荐做法如下：

Connection jdbcConnection = null;

Statement stmt = null;

try

{

........

}

catch (SQLException e)

{

........

}

finally

{

if (stmt != null)

{

try

{

stmt.close();

}

catch (SQLException e)

{

logger.log(Level.WARNING, "异常说明", e);

}

}

if (jdbcConnection != null)

{

try

{

jdbcConnection.close();

}

catch (SQLException e)

{

logger.log(Level.WARNING, "异常说明", e);

}

}

}

军规六：【异常捕获不要直接 catch(Exception ex) ，应该把异常细分处理。】

说明：catch (Exception ex)的结果会把RuntimeException异常捕获，RuntimeException是运行期异常，是程序本身考虑不周而抛出的异常，是程序的BUG，如无效参数、数组越界、被零除等，程序必须确保不能抛出RuntimeException异常，不允许显示捕获RuntimeException异常就是为了方便测试中容易发现程序问题。

军规七：【对于if „ else if „(后续可能有多个elseif …)这种类型的条件判断，最后必须包含一个else分支，避免出现分支遗漏造成错误；每个switch-case语句都必须保证有default，避免出现分支遗漏，造成错误。】

军规八：【覆写对象的equals()方法时必须同时覆写hashCode()方法。】

说明：equals和hashCode方法是对象在hash容器内高效工作的基础，正确的覆写这两个方法才能保证在hash容器内查找对象的正确性，同时一个好的hashCode方法能大幅提升hash容器效率。

军规九：【禁止循环中创建新线程，尽量使用线程池。】

军规十：【在进行精确计算时(例如:货币计算)避免使用float和double，浮点数计算都是不精确的，必须使用BigDecimal或将浮点数运算转换为整型运算。】

说明：浮点运算在一个范围很广的值域上提供了很好的近似，但是它不能产生精确的结果。二进制浮点对于精度计算是非常不适合的，因为它不可能将0.1——或者10的其它任何次负幂精确表示为一个长度有限的二进制小数。