JAVA语言编码规范

姓名： 李依芳

班级： 信1604-1

学号： 20163755

日期： 2019年6月14日

目录

[1 编程规约 3](#_Toc11443173)

[1.1命名风格 3](#_Toc11443174)

[1.2常量的定义 4](#_Toc11443175)

[1.3 代码格式 4](#_Toc11443176)

[1.4 OOP规约 5](#_Toc11443177)

[1.5 并发处理 5](#_Toc11443178)

[1.6控制语句 6](#_Toc11443179)

[1.7注释 7](#_Toc11443180)

[2 异常日志 7](#_Toc11443181)

[2.1异常处理 7](#_Toc11443182)

[2.2 日志规约 8](#_Toc11443183)

[3 单元测试 8](#_Toc11443184)

[4 MySQL 数据库 8](#_Toc11443185)

[4.1建表规约 8](#_Toc11443186)

[4.2索引规约 9](#_Toc11443187)

[4.3 SQL语句 9](#_Toc11443188)

[5设计规约 9](#_Toc11443189)

# 1 编程规约

## 1.1命名风格

（1）代码中的命名均不能以下划线或美元符号开始，也不能以下划线或美元符号结束。

如：\_\_name / $name / name\_ / name$，这是不允许的。

（2）类名大小写的使用要有意义不能混乱使用，毫无意义。

如：macroPolo / UserDo / XMLService / TCPUDPDeal，这是不允许的。

（3）常量命名全部大写，单词间用下划线隔开，力求语义表达完整清楚。

如：MAXRESULT,该类名称是不建议的，最好写成MAX\_RESULT。

（4）代码中的命名严禁使用拼音与英文混合的方式，更不允许直接使用中文的方式。 说明：正确的英文拼写和语法可以让阅读者易于理解，避免歧义。注意，即使纯拼音命名方式 也要避免采用。

（5）包名统一使用小写，点分隔符之间有且仅有一个自然语义的英语单词。包名统一使用 单数形式，但是类名如果有复数含义，类名可以使用复数形式。

如：应用工具类包名为 com.alibaba.ai.util、类名为 MessageUtils，这是允许并建议的。

（6）杜绝完全不规范的缩写，避免望文不知义

如：AbstractClass“缩写”命名成 AbsClass；condition“缩写”命名成 condi，此类随 意缩写严重降低了代码的可阅读性。这是不允许的。

（7）接口类中的方法和属性不要加任何修饰符号（public 也不要加），保持代码的简洁 性，并加上有效的 Javadoc 注释。尽量不要在接口里定义变量，如果一定要定义变量，肯定是 与接口方法相关，并且是整个应用的基础常量。

如：接口方法定义 public abstract void f();这是不允许的。

（8）接口和实现类的命名有两套规则

对于 Service 和 DAO 类，基于 SOA 的理念，暴露出来的服务一定是接口，内部 的实现类用 Impl 的后缀与接口区别。 正例：CacheServiceImpl 实现 CacheService 接口。

如果是形容能力的接口名称，取对应的形容词为接口名（通常是–able 的形式）。 正例：AbstractTranslator 实现 Translatable 接口。

（9）枚举类名建议带上 Enum 后缀，枚举成员名称需要全大写，单词间用下划线隔开。 说明：枚举其实就是特殊的类，域成员均为常量，且构造方法被默认强制是私有.

## 1.2常量的定义

（1）不允许任何未经预先定义的常量）直接出现在代码中。

如：*String key = "Id#taobao\_" + tradeId;*

*cache.put(key, value);* 这是不允许的。

（2）在 long 或者 Long 赋值时，数值后使用大写的 L，不能是小写的 l，小写容易跟数字 1 混淆，造成误解。

（3）如果变量值仅在一个固定范围内变化用 enum 类型来定义。 说明：如果存在名称之外的延伸属性应使用 enum 类型，下面正例中的数字就是延伸信息，表 示一年中的第几个季节。

如：

*public enum SeasonEnum {*

*SPRING(1), SUMMER(2), AUTUMN(3), WINTER(4);*

*private int seq;*

*SeasonEnum(int seq){*

*this.seq = seq;*

*}*

*} ，*这是允许并推荐的。

## 1.3 代码格式

（1）大括号的使用约定。如果是大括号内为空，则简洁地写成{}即可，不需要换行；如果 是非空代码块则：

①左大括号前不换行。

②左大括号后换行。

③右大括号前换行。

④右大括号后还有 else 等代码则不换行；表示终止的右大括号后必须换行。

（2）左小括号和字符之间不出现空格；同样，右小括号和字符之间也不出现空格；而左大括号前需要空格。

（3）if/for/while/switch/do 等保留字与括号之间都必须加空格。

（4）任何二目、三目运算符的左右两边都需要加一个空格。

（5）采用 4 个空格缩进，禁止使用 tab 字符。 说明：如果使用 tab 缩进，必须设置 1个tab 为4个空格。IDEA 设置 tab 为 4 个空格时，请勿勾选 Use tab character；而在 eclipse中，必须勾选 insert spaces for tabs。

（6）注释的双斜线与注释内容之间有且仅有一个空格

（7）单行字符数限制不超过 120 个，超出需要换行

（8）IDE 的 text file encoding 设置为 UTF-8; IDE 中文件的换行符使用 Unix 格式， 不要使用 Windows 格式。

（9）不同逻辑、不同语义、不同业务的代码之间插入一个空行分隔开来以提升可读性。 说明：任何情形，没有必要插入多个空行进行隔开。

## 1.4 OOP规约

（1）避免通过一个类的对象引用访问此类的静态变量或静态方法，无谓增加编译器解析成 本，直接用类名来访问即可

（2）所有的覆写方法，必须加@Override 注解

（3）相同参数类型，相同业务含义，才可以使用 Java 的可变参数，避免使用 Object。

（4）外部正在调用或者二方库依赖的接口，不允许修改方法签名，避免对接口调用方产生 影响。接口过时必须加@Deprecated 注解，并清晰地说明采用的新接口或者新服务是什么。

（5）Object 的 equals 方法容易抛空指针异常，应使用常量或确定有值的对象来调用 equals

（6）构造方法里面禁止加入任何业务逻辑，如果有初始化逻辑，请放在 init 方法中。

（7）当一个类有多个构造方法，或者多个同名方法，这些方法应该按顺序放置在一起， 便于阅读，此条规则优先于第 8 条规则。

（8）循环体内，字符串的连接方式，使用 StringBuilder 的 append 方法进行扩展。 说明：下例中，反编译出的字节码文件显示每次循环都会 new 出一个 StringBuilder 对象， 然后进行 append 操作，最后通过 toString 方法返回 String 对象，造成内存资源浪费。

## 1.5 并发处理

（1）创建线程或线程池时请指定有意义的线程名称，方便出错时回溯。

（2）线程资源必须通过线程池提供，不允许在应用中自行显式创建线程。 说明：使用线程池的好处是减少在创建和销毁线程上所消耗的时间以及系统资源的开销，解决资源不足的问题。如果不使用线程池，有可能造成系统创建大量同类线程而导致消耗完内存或 者“过度切换”的问题。

（3）SimpleDateFormat 是线程不安全的类，一般不要定义为 static 变量，如果定义为 static，必须加锁，或者使用 DateUtils 工具类。

（4）高并发时，同步调用应该去考量锁的性能损耗。能用无锁数据结构，就不要用锁；能锁区块，就不要锁整个方法体；能用对象锁，就不要用类锁。

（5）多线程并行处理定时任务时，Timer 运行多个 TimeTask 时，只要其中之一没有捕获 抛出的异常，其它任务便会自动终止运行，使用 ScheduledExecutorService 则没有这个问题。

（6）在并发场景下，通过双重检查锁（double-checked locking）实现延迟初始化的优 化问题隐患(可参考 The "Double-Checked Locking is Broken" Declaration)，推荐解 决方案中较为简单一种（适用于 JDK5 及以上版本），将目标属性声明为 volatile 型。

（7）ThreadLocal 无法解决共享对象的更新问题，ThreadLocal 对象建议使用 static 修饰。这个变量是针对一个线程内所有操作共享的，所以设置为静态变量，所有此类实例共享 此静态变量 ，也就是说在类第一次被使用时装载，只分配一块存储空间，所有此类的对象(只 要是这个线程内定义的)都可以操控这个变量。

## 1.6控制语句

（1）在一个 switch 块内，每个 case 要么通过 break/return 等来终止，要么注释说明程 序将继续执行到哪一个 case 为止；在一个 switch 块内，都必须包含一个 default 语句并且 放在最后，即使空代码。

（2）在 if/else/for/while/do 语句中必须使用大括号。即使只有一行代码，避免采用 单行的编码方式：if (condition) statements;

（3）在高并发场景中，避免使用”等于”判断作为中断或退出的条件。 说明：如果并发控制没有处理好，容易产生等值判断被“击穿”的情况，使用大于或小于的区间 判断条件来代替

（4）除常用方法（如 getXxx/isXxx）等外，不要在条件判断中执行其它复杂的语句，将 复杂逻辑判断的结果赋值给一个有意义的布尔变量名，以提高可读性。

（5）下列情形，不需要进行参数校验：

①极有可能被循环调用的方法。但在方法说明里必须注明外部参数检查要求。

②底层调用频度比较高的方法。毕竟是像纯净水过滤的最后一道，参数错误不太可能到底 层才会暴露问题。一般 DAO 层与 Service 层都在同一个应用中，部署在同一台服务器中，所 以 DAO 的参数校验，可以省略。

③被声明成 private 只会被自己代码所调用的方法，如果能够确定调用方法的代码传入参 数已经做过检查或者肯定不会有问题，此时可以不校验参数。

## 1.7注释

（1）类、类属性、类方法的注释必须使用 Javadoc 规范，使用/\*\*内容\*/格式，不得使用 // xxx 方式。

（2）所有的类都必须添加创建者和创建日期

（3）所有的枚举类型字段必须要有注释，说明每个数据项的用途

（4）对于注释的要求：第一、能够准确反应设计思想和代码逻辑；第二、能够描述业务含 义，使别的程序员能够迅速了解到代码背后的信息。完全没有注释的大段代码对于阅读者形同天书，注释是给自己看的，即使隔很长时间，也能清晰理解当时的思路；注释也是给继任者看 的，使其能够快速接替自己的工作。

# 2 异常日志

## 2.1异常处理

(1)Java 类库中定义的可以通过预检查方式规避的 RuntimeException 异常不应该通过 catch 的方式来处理，比如：NullPointerException，IndexOutOfBoundsException 等等。

(2) 捕获异常是为了处理它，不要捕获了却什么都不处理而抛弃之，如果不想处理它,请将该异常抛给它的调用者。最外层的业务使用者，必须处理异常,将其转化为用户可以理解的内容。

(3) 定义时区分 unchecked / checked异常,避免直接抛出 new RuntimeException()，更不允许抛Exception或者Throwable,应使用有业务含义的自定义异常。推荐业界已定义过的自定义异常，如：DAOException / ServiceException 等.

(4) 避免出现重复的代码（Don’t Repeat Yourself），即 DRY 原则.

## 2.2 日志规约

（1）避免重复打印日志，浪费磁盘空间，务必在 log4j.xml 中设置 additivity=false

（2）谨慎地记录日志。生产环境禁止输出 debug 日志；有选择地输出 info 日志；如果使 用 warn 来记录刚上线时的业务行为信息，一定要注意日志输出量的问题，避免把服务器磁盘 撑爆，并记得及时删除这些观察日志。

（3）尽量用英文来描述日志错误信息，如果日志中的错误信息用英文描述不清楚的话使用 中文描述即可，否则容易产生歧义。国际化团队或海外部署的服务器由于字符集问题。

# 单元测试

（1）好的单元测试必须遵守 AIR 原则。

（2）单元测试应该是全自动执行的，并且非交互式的。测试用例通常是被定期执行的，执 行过程必须完全自动化才有意义。输出结果需要人工检查的测试不是一个好的单元测试。单元 测试中不准使用 System.out 来进行人肉验证，必须使用 assert 来验证。

（3）核心业务、核心应用、核心模块的增量代码确保单元测试通过。

（4）单元测试的基本目标：语句覆盖率达到 70%；核心模块的语句覆盖率和分支覆盖率都 要达到 100%

# 4 MySQL 数据库

## 4.1建表规约

（1）表达是与否概念的字段，必须使用 is\_xxx 的方式命名，数据类型是 unsigned tinyint （1 表示是，0 表示否）。

（2）表名、字段名必须使用小写字母或数字，禁止出现数字开头，禁止两个下划线中间只 出现数字。数据库字段名的修改代价很大，因为无法进行预发布，所以字段名称需要慎重考虑。

（3）如果存储的字符串长度几乎相等，使用 char 定长字符串类型。

（4）varchar 是可变长字符串，不预先分配存储空间，长度不要超过 5000，如果存储长 度大于此值，定义字段类型为 text，独立出来一张表，用主键来对应，避免影响其它字段索 引效率。

（5）合适的字符存储长度，不但节约数据库表空间、节约索引存储，更重要的是提升检索速度。

## 4.2索引规约

（1）业务上具有唯一特性的字段，即使是多个字段的组合，也必须建成唯一索引。

（2）超过三个表禁止 join。需要 join 的字段，数据类型必须绝对一致；多表关联查询时， 保证被关联的字段需要有索引。

（3）页面搜索严禁左模糊或者全模糊，如果需要请走搜索引擎来解决。

（4）建组合索引的时候，区分度最高的在最左边。

## 4.3 SQL语句

（1）不要使用 count(列名)或 count(常量)来替代 count(\*)，count(\*)是 SQL92 定义的 标准统计行数的语法，跟数据库无关，跟 NULL 和非 NULL 无关。

（2）不得使用外键与级联，一切外键概念必须在应用层解决。

（3）数据订正（特别是删除、修改记录操作）时，要先 select，避免出现误删除，确认 无误才能执行更新语句。

（4）如果有国际化需要，所有的字符存储与表示，均以 utf-8 编码，注意字符统计函数 的区别。

# 5设计规约

（1）在需求分析阶段，如果与系统交互的 User 超过一类并且相关的 User Case 超过 5 个， 使用用例图来表达更加清晰的结构化需求。

（2）如果某个业务对象的状态超过 3 个，使用状态图来表达并且明确状态变化的各个触发条件。

（3）如果系统中某个功能的调用链路上的涉及对象超过 3 个，使用时序图来表达并且明确 各调用环节的输入与输出。

（4）如果系统中超过 2 个对象之间存在协作关系，并且需要表示复杂的处理流程，使用活

动图来表示。

（5）类在设计与实现时要符合单一原则。

（6）谨慎使用继承的方式来进行扩展，优先使用聚合/组合的方式来实现。

（7）避免如下误解：敏捷开发 = 讲故事 + 编码 + 发布。

（8）系统设计主要目的是明确需求、理顺逻辑、后期维护、次要目的用于指导编码。 说明：避免为了设计而设计，系统设计文档有助于后期的系统维护，所以设计结果需要进行分

类归档保存。