Java

一、命名

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。

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。

抽象类命名使用 Abstract 或 Base 开头； 异常类命名使用 Exception 结尾； 测试类命名以它要测试的类的名称开始，以 Test 结尾。

中括号是数组类型的一部分，数组定义如下： String[] args;

POJO 类中布尔类型的变量，都不要加 is，否则部分框架解析会引起序列化错误。

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。

Service/DAO 层方法命名规约

1） 获取单个对象的方法用 get 做前缀。

2） 获取多个对象的方法用 list 做前缀。

3） 获取统计值的方法用 count 做前缀。

4） 插入的方法用 save（推荐） 或 insert 做前缀。

5） 删除的方法用 remove（推荐） 或 delete 做前缀。

6） 修改的方法用 update 做前缀。

二、常量定义

不允许任何未经定义的常量直接出现在代码中。

long 或者 Long 初始赋值时，必须使用大写的 L，不能是小写的 l，小写容易跟数字1 混淆，造成误解。

不要使用一个常量类维护所有常量，应该按常量功能进行归类，分开维护。如：缓存相关的常量放在类： CacheConsts 下； 系统配置相关的常量放在类： ConfigConsts 下。

三、OOP规约

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。

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。

不能使用过时的类或方法。

定义 DO/DTO/VO 等 POJO 类时，不要设定任何属性默认值(在数据提取时并没有置入具体值，在更新其它字段时又附带更新了此字段，导致创建时间被修改成当前时间。)

序列化类新增属性时，请不要修改 serialVersionUID 字段，避免反序列失败； 如果完全不兼容升级，避免反序列化混乱，那么请修改 serialVersionUID 值。注意 serialVersionUID 不一致会抛出序列化运行时异常。

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。

类内方法定义顺序依次是：公有方法或保护方法 > 私有方法 > getter/setter方法。

循环体内，字符串的连接方式，使用 StringBuilder 的 append 方法进行扩展。

慎用 Object 的 clone 方法来拷贝对象。对象的 clone 方法默认是浅拷贝，若想实现深拷贝需要重写 clone 方法实现属性对象的拷贝。

三、集合处理

在 subList 场景中， 高度注意对原集合元素个数的修改，会导致子列表的遍历、增加、删除均产生 ConcurrentModificationException 异常。

使用集合转数组的方法，必须使用集合的 toArray(T[] array)，传入的是类型完全一样的数组，大小就是 list.size()。

使用工具类 Arrays.asList()把数组转换成集合时，不能使用其修改集合相关的方法，它的 add/remove/clear 方法会抛出 UnsupportedOperationException 异常。

泛型通配符<? extends T>来接收返回的数据，此写法的泛型集合不能使用 add 方法， 而<? super T>不能使用 get 方法，做为接口调用赋值时易出错。

不要在 foreach 循环里进行元素的 remove/add 操作。 remove 元素请使用 Iterator方式，如果并发操作，需要对 Iterator 对象加锁。

集合初始化时， 指定集合初始值大小。

使用 entrySet 遍历 Map 类集合 KV，而不是 keySet 方式进行遍历。

四、控制语句

在 if/else/for/while/do 语句中必须使用大括号。 即使只有一行代码，避免使用单行的形式： if (condition) statements;

表达异常的分支时， 少用 if-else 方式。如果非得使用 if()...else if()...else...方式表达逻辑，避免后续代码维护困难， 请勿超过 3 层。

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。

五、异常处理

Java 类库中定义的一类 RuntimeException 可以通过预先检查进行规避，而不应该通过 catch 来处理，比如： IndexOutOfBoundsException， NullPointerException 等等。

异常不要用来做流程控制，条件控制，因为异常的处理效率比条件分支低。

有 try 块放到了事务代码中， catch 异常后，如果需要回滚事务，一定要注意手动回滚事务。

防止空指针异常，是程序员的基本修养。

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

六、日志规约

应用中不可直接使用日志系统（Log4j、 Logback） 中的 API，而应依赖使用日志框架SLF4J 中的 API，使用门面模式的日志框架，有利于维护和各个类的日志处理方式统一。

日志文件推荐至少保存 15 天，因为有些异常具备以“周”为频次发生的特点。

对 trace/debug/info 级别的日志输出，必须使用条件输出形式或者使用占位符的方式。

谨慎地记录日志。

七、其他

在使用正则表达式时，利用好其预编译功能，可以有效加快正则匹配速度。

后台输送给页面的变量必须加$!{var}——中间的感叹号。

不要在视图模板中加入任何复杂的逻辑。

任何数据结构的构造或初始化，都应指定大小。

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。