1.1 百度Java编码规范V1.3

前言

为统一公司Java项目编码规范，提高可读性，特制定本规范。

本规范同样适用于Android项目。

本规范基于Google Java Style和阿里巴巴Java开发规范，主要的区别如下：

- 保留了百度原有规范的4空格缩进和120字符列宽限制。
- 放松了对import语句的组织方式要求。只要有一定逻辑性即可，不做严格限制。但要求未使用的import语句必须删除。
- 对Android的成员变量命名允许了例外。
    - 非public非static的变量可以使用m开头
    - 非常量的static变量可以使用s开头
- switch语句禁止了case贯穿，同时要求default必须被终止。
- 允许特殊场景下的单字符参数命名，例如使用x、y表示坐标。
- 禁用了枚举类的单行风格。

```
private enum Suit { CLUBS, HEARTS, SPADES, DIAMONDS }
```

- 禁用了无参数注解的不换行风格

```
@Override public int hashCode() { ... }
@Partial @Mock DataLoader loader;
```

- 增加了对使用拼音命名的说明：不建议但允许少量例外。
- 禁止使用连续的空行进行分隔。
- 禁用了一种数组初始化换行的风格。

```
new int[]
{0, 1, 2, 3}
```

- 禁止使用连续的空格分隔。不再允许使用增加空格来进行对齐的风格。
- 允许包名中包含数字，例如org.apache.logging.log4j。
- 增加了换行符强制使用Unix格式的要求。
- 删除了非ASCII字符的要求。
- 删除了类成员顺序的要求。
- 简化了列宽和换行要求描述。
- 删除了枚举类的要求。
- 补充特定类型命名规范，增加对抽象类、异常、单测的类名要求，增加POJO布尔类型变量获取方法、内部实现方法、魔法值等方面的严格要求。
- 补充编程实践方面要求，完善OOP、集合处理、并发处理、异常处理等方面要求。
- 删除了Javadoc注释的要求。
- 增加了Java安全编码规范要求。

相对于原先的百度Java编码规范，本规范做出了大量更加严格的限制，并加强了相关编程实践方面内容。

术语

除非特别约定，我们使用以下术语：

- **类**用于指代所有类（class）、枚举（enum class）、接口（interface）以及注解（annotation）。
- **注释**仅用于指代行间注释，**Javadoc**用于指代文档注释。

文档说明

本文中所使用的示例代码不具备唯一权威性。示例代码只代表一种符合规范的写法，也可能存在其他符合规范的不同写法。

目录

1. 源文件规范

1.1. 文件名

[强制] [JAVA001] 源文件名必须和它包含的顶层类名保持一致，包括大小写，并以.java作为后缀名。

1.2. 文件编码

[强制] [JAVA002] 所有源文件编码必须是**UTF-8**

1.3. 特殊字符

[强制] [JAVA003] 除了换行符之外，ASCII空格（0x20）是唯一合法的空格字符。这意味着：

- 所有在源代码中（包括字符、字符串以及注释中）出现的其他空格字符需要转义,例如Tab用\t表示。
- 缩进必须使用空格而不是Tab

[强制] [JAVA004] 对于有特殊转义表示的字符（\b,\t,\n,\f,\r,\",\',\\），禁止使用其它等价转义方式。例如\012或者\u00a表示。

[强制] [JAVA005]文件的换行符使用 Unix 格式，不要使用 Windows 格式。

2. 源文件组织结构

[强制] [JAVA006] 源文件必须按顺序由以下部分组成：

- 许可证（License）或版权声明（Copyright）
- package语句
- import语句
- 唯一的顶层类

每两部分之间用一个空行分隔，不允许多个空行。

2.1. 许可证（License）或版权声明（Copyright）

如果文件有许可证（License）或版权声明（Copyright），放在最开头。如果没有的话，此部分可以忽略。

2.2. package语句

[强制] [JAVA007] package语句占据单独一行不换行，允许超出120字符列宽限制。

2.3. import语句

[强制] [JAVA008] 非static imports，禁止使用通配符import。

[强制] [JAVA009] 每条import语句占据单独一行不换行，允许超出120字符列宽限制。

[强制] [JAVA010] 所有未使用的import语句应该被删除。

[推荐] [JAVA011] import语句需按照一定的逻辑顺序组织。

可参考以下顺序进行分组，每两组之间用一个空行分隔：

- 所有的static import语句
- 每个顶层包独立一组，按ASCII顺序排列，按照第三方包、java、javax、baidu内公共库分组。

组内不包含空行，按照所import的包名的ASCII码顺序排列。

2.4. 类声明

[强制] [JAVA012] 每个源文件只允许包含唯一一个顶层类。

[强制] [JAVA013] 重载的方法必须放在一起，即同名的构造函数或方法之间禁止插入其他成员。

3. 代码格式

3.1. 花括号

[强制] [JAVA014] 在非空代码块中使用花括号时要遵循K&R风格（Kernighan and Ritchie Style）：

- 左花括号（{）前不能换行，在其后换行。
- 在右花括号（}）前要有换行。
- 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。

```
return new MyClass() { //

    @Override
    public void method() {
        if (condition()) {
            try {
                do {
                    something();
                } while (!stop()); // do-while
            } catch (ProblemException e) { // try-catch
                recover();
            } // try-catch
        } else { // if-else
            doSomethingElese();
        } // if-else
    }
}; //
```

[强制] [JAVA015] 如果一个代码块是空的，可以直接使用{}。

3.2. 块缩进：4个空格

[强制] [JAVA016] 每次开始书写一个新的代码块时，使用4个空格进行缩进，在代码块结束时，恢复之前的缩进级别。

3.3. 每行只写一条语句

[强制] [JAVA017] 每条语句之后都要换行。

3.4. 列宽和换行

[强制] [JAVA018] 单行字符数限制不超过120个，超出需要换行，换行时遵循如下原则：

- 第二行相对第一行缩进4个空格，从第三行开始，不再继续缩进，参考示例。
- 运算符与下文一起换行。
- 方法调用的点符号与下文一起换行。
- 方法调用中的多个参数需要换行时，在逗号后进行。
- 在括号前不要换行，见反例。

```
//
StringBuffer buffer = new StringBuffer();
// 1204
buffer.append("abc").append("bcd")...
    .append("ert")...
    .append("dfg")...
    .append("fds");


//
StringBuffer buffer = new StringBuffer();
// 120
buffer.append("abc").append("bcd")...append
    ("abc");


// 120
method(args1, args2, args3, ...
    , argsX);
```

3.5.空白

[强制] [JAVA019] 在以下情况下增加空行：

- 在类的不同的成员间增加空行，包括：成员变量、构造函数、方法、内部类、静态初始化块、实例初始化块等
- 两个成员变量声明之间可以不加空行。空行通常用于对成员变量进行逻辑分组。
- 方法体内，按需增加空行，以便从逻辑上对语句进行分组

> 禁止使用连续的空行。

[强制] [JAVA020] 除了语法要求，字符串内的空格，以及JavaDoc里的空格，需要在下列情况里使用空格

- if/for/while/switch/do/else/catch 等保留字与括号之间都必须加空格。
- 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；
- 方法参数在定义和传入时，多个参数逗号后边必须加空格。
- 在任意左花括号（{）之前要有一个空格。@SomeAnnotation({a,b})和String[][]x={{"foo"}};这两种情况除外。
- 在二目和三目运算符两边各要有一个空格。该规则同样适用于<TextendsFoo&Bar>中的&，catch(FooException|BarExceptione)中的|，以及foreach中的:
- 在逗号（,）、冒号（:）和类型转换用的右圆括号（)）后要有一个空格
- 在用于行末注释的//前后各要有一个空格
- 在声明语句的类型和名称之间要有一个空格，比如List<String>list
- 初始化数组的形式只可使用如下方式：

```
new int[] {5, 6}
```

- 除行首缩进、注释和字符串内的空格以外，禁止使用连续的空格

```
public class Example {
    public List<Element> getAllValidElements(String[] nodes) throws
PermissionDeniedException {
        List<Element> result;
        for (int i = 0; i < nodes.length; ++i) {
            if (checkPermission(nodes[i])) {
                for (int id : getAllElementIdList(nodes[i])) {
                    Element e = (Element) getElement(id);
                    if (isValidElement(e)) {
                        result.add(e);
                    }
                }
            } else {
                throw new PermissionDeniedException("Can not access node "
+ nodes[i]);
            }
        }
        return result;
    }
}
```

3.6. 表达式圆括号

由于不能保证所有人都清楚Java操作符优先级，因此推荐在表达式中增加圆括号用来明确其中运算的优先级，不做强制要求。

3.7. 其他说明

[强制] [JAVA021] 一行只声明一个变量，在需要时声明变量，声明后尽快初始化。

[强制] [JAVA022] 类型与中括号紧挨相连来 表示 数组。用String[]args的方式来声明数组，而非Stringargs[]。

[强制] [JAVA023] 添加在类、方法、构造函数、成员属性上的注解（Annotation）直接写在注释块之后，每个注解独占一行。

[强制] [JAVA024] 当同时使用多个修饰符时，按照下列顺序：

```
public protected private abstract static final transient volatile
synchronized native strictfp
```

[强制] [JAVA025] 长整型数字必须使用大写字母L结尾，不能使用小写字母l，以便和数字1进行区分。例如使用3000000000L而不是3000000000l

4. 命名

4.1. 通用命名规范

所有的标识符只允许使用ASCII字符和数字合法的，标识符命名必须能够匹配正则表达式：\w ([0−9a−zA−Z_])。

禁止使用一些特定的前缀和后缀，比如：name_、mName、m_name。

不建议使用中文拼音来命名。例外：

- 得到广泛认可的中文产品名，像贴吧、凤巢等
- 以个人命名的算法或数据结构的实现

4.2. 特定类型命名规范

[强制] [JAVA026] 代码中的下划线或美元符号结束。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。所有的标识符只允许使用ASCII字符和数字。合法的标识符命名必须能够匹配正则表达式：\w ([0-9a-zA-Z_])

[强制] [JAVA027] 类名使用 UpperCamelCase 风格，但以下情形例外：DO / BO / DTO / VO / AO / PO / UID 等。

[强制] [JAVA028] 方法名、参数名、成员变量、局部变量都统一使用 lowerCamelCase 风格，必须遵从 驼峰形式。

[强制] [JAVA029] 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，不要嫌名字长。

[强制] [JAVA030] 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。

[强制] [JAVA031] POJO 类中布尔类型的变量，都不要加 is 前缀，否则部分框架解析会引起序列化错误。

[强制] [JAVA032] 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。

[强制] [JAVA033] 杜绝完全不规范的缩写，避免望文不知义。

[强制] [JAVA034] 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。

[强制] [JAVA035] 不允许任何魔法值（即未经预先定义的常量）直接出现在代码中。

[强制] [JAVA036] 泛型类型变量名必须遵循以下两种方式之一：

- 单独一个大写字母，有时后面再跟一个数字。（例如，E、T2）。
- 类命名的最后接一个大写字母。（例如，RequestT）。

4.3. 驼峰命名方式定义

通常有多种方式将短语组织成驼峰方式，像一些缩写词：IPv6、iOS等。为了统一，必须遵循以下几点规则。

- 将字符全部转换为ASCII字符，并且去掉’等符号。例如，M ü ller'salgorithm被转换为Muellersalgorithm
- 在空格和标点符号处对上一步的结果进行切分，组成一个词组。

> 推荐：一些已经是驼峰命名的词语，也应该在这个时候被拆分。（例如 AdWords 被拆分为 ad words）。但是例如iOS之类的词语，它其实不是一个驼峰形式的词语，而是人们惯例使用的一个词语，因此不用做拆分。

- 经过上面两步后，先将所有的字母转换为小写，再把每个词语的第一个字母转换为大写。
- 最后，将所有词语连在一起，形成一个标识符。

注意：词语原来的大小写规则，应该被完全忽略。以下是一些例子：

原始语句	正确写法	非法写法
"XML HTTP request"	XmlHttpRequest	XMLHTTPRequest
"new customer ID"	newCustomerId	newCustomerID
"inner stopwatch"	innerStopwatch	innerStopWatch
"supports IPv6 on iOS?"	supportsIpv6OnIos	supportsIPv6OnIOS
"YouTube importer"	YouTubeImporter or YoutubeImporter[1]	

[1]号表示可以接受，但是不建议使用。

> 注意：有些词语在英文中，可以用[-]连接使用，也可以不使用[-]直接使用。例如“nonempty”和“non-empty”都可以。因此方法名字为checkNonempty或者checkNonEmpty都是合法的写法。

5. 编程实践

5.1. OOP规约

关于基本数据类型与包装数据类型的使用标准如下：

1）[强制] [JAVA037] 所有的 POJO 类属性必须使用包装数据类型。

2）[强制] [JAVA038] RPC 方法的返回值和参数必须使用包装数据类型。

3）[推荐] [JAVA039] 所有的局部变量使用基本数据类型。

> 说明：POJO 类属性没有初值是提醒使用者在需要使用时，必须自己显式地进行赋值，任何 NPE 问题，或者入库检查，都由使用者来保证。

[强制] [JAVA040] 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。

[强制] [JAVA041] 所有的覆写方法，必须加@Override 注解。

[强制] [JAVA042] 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。

[强制] [JAVA043] 不能使用过时的类或方法。

[强制] [JAVA044] 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。

```
//
"test".equals(object);
//
object.equals("test");
//  java.util.Objects#equalsJDK7
```

[强制] [JAVA045] 所有的相同类型的包装类对象之间值的比较，全部使用 equals 方法比较。

[强制] [JAVA046] 定义 DO/DTO/VO 等 POJO 类时，不要设定任何属性默认值。

[强制] [JAVA047] 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。

[强制] [JAVA048] 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。

[强制] [JAVA049] POJO 类必须写 toString 方法。

[强制] [JAVA050] 禁止在类中，同时存在对应属性 xxx 的 isXxx() 和 getXxx() 方法。

> 说明：框架在调用属性 xxx 的提取方法时，并不能确定哪个方法一定是被优先调用到。

[强制] [JAVA051] 禁止覆盖（Override）Object.finalize方法。

5.2. 集合处理

[强制] [JAVA052] 关于 hashCode 和 equals 的处理，只要重写 equals，就必须重写 hashCode。

[强制] [JAVA053] ArrayList 的 subList 结果不可强转成 ArrayList，否则会抛出 ClassCastException 异常，即 java.util.RandomAccessSubList cannot be cast to java.util.ArrayList。在 subList 场景中，高度注意对原集合元素的增加或删除，均会导致子列表的遍历、 增加、删除产生 ConcurrentModificationException 异常。

> 说明：subList 返回的是 ArrayList 的内部类 SubList，并不是 ArrayList 而是 ArrayList 的一个视图，对于 SubList 子列表的所有操作最终会反映到原列表上。

[强制] [JAVA054] 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()。

说明：使用 toArray 带参方法，入参分配的数组空间不够大时，toArray 方法内部将重新分配 内存空间，并返回新数组地址；如果数组元素个数大于实际所需，下标为[list.size()] 的数组元素将被置为 null，其它数组元素保持原值，因此最好将方法入参数组大小定义与集 合元素个数一致。

```
//
List list = new ArrayList(2);
list.add("guan");
list.add("bao");
String[] array = new String[list.size()];
array = list.toArray(array);


//
//  toArray  Object[] ClassCastException
```

[强制] [JAVA055] 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。

说明：asList 的返回对象是一个 Arrays 内部类，并没有实现集合的修改方法。Arrays.asList 体现的是适配器模式，只是转换接口，后台的数据仍是数组。

```
String[] str = new String[] { "you", "wu" };
List list = Arrays.asList(str);
// list.add("yangguanbao");
// str[0] = "gujin";  list.get(0)
```

[强制] [JAVA056] 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，作为接口调用赋值时易出错。

说明：扩展说一下 PECS(Producer Extends Consumer Super)原则：第一、频繁往外读取内 容的，适合用<? extends T>。第二、经常往里插入的，适合用<? super T>。

[强制] [JAVA057] 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。

```
//
List list = new ArrayList<>();
list.add("1");
list.add("2");
Iterator iterator = list.iterator();
while (iterator.hasNext()) {
    String item = iterator.next();
        if () {
            iterator.remove();
        }
}
//
for (String item : list) {
    if ("1".equals(item)) {
        list.remove(item);
    }
}
```

5.3. 并发处理

[强制] [JAVA058] 获取单例对象需要保证线程安全，其中的方法也要保证线程安全 (Immutable对象除外)。

[强制] [JAVA059] 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，并设置为daemon状态。

[强制] [JAVA060] 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。

[强制] [JAVA061] 线程池不允许使用 Executors 去创建，而是通过 ThreadPoolExecutor 的方式，这样的处理方式让写的同学更加明确线程池的运行规则，规避资源耗尽的风险。

> 说明：Executors 返回的线程池对象的弊端如下：
>
> - FixedThreadPool 和 SingleThreadPool: 允许的请求队列长度为 Integer.MAX_VALUE，可能会堆积大量的请求，从而导致 OOM。
> - CachedThreadPool 和 ScheduledThreadPool: 允许的创建线程数量为 Integer.MAX_VALUE，可能会创建大量的线程，从而导致 OOM。

[强制] [JAVA062] 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。

> 说明：如果是 JDK8 的应用，可以使用 Instant 代替 Date，LocalDateTime 代替 Calendar， DateTimeFormatter 代替 SimpleDateFormat，官方给出的解释：simple beautiful strong immutable thread-safe。

[强制] [JAVA063] 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。

[强制] [JAVA064] 对多个资源、数据库表、对象同时加锁时，需要保持一致的加锁顺序，否则可能会造成死锁。

[强制] [JAVA065] 并发修改同一记录时，避免更新丢失，需要加锁。要么在应用层加锁，要么在缓存加 锁，要么在数据库层使用乐观锁，使用 version 作为更新依据。

[强制] [JAVA066] 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。

5.4. 控制语句

[强制] [JAVA067] 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。

[强制] [JAVA068] 在 if/else/for/while/do 语句中必须使用花括号。即使只有一行代码，避免采用 单行的编码方式：if (condition) statements;

[强制] [JAVA069] 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。

5.5. 注释规约

[强制] [JAVA070] 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。

[强制] [JAVA071] 所有的抽象方法（包括接口中的方法）必须要用 Javadoc 注释、除了返回值、参数、 异常说明外，还必须指出该方法做什么事情，实现什么功能。

[强制] [JAVA072] 方法内部单行注释，在被注释语句上方另起一行，使用//注释。方法内部多行注释 使用/* */注释，注意与代码对齐。

[强制] [JAVA073] 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。

[推荐] [JAVA074] 所有的类都必须添加创建者和创建日期。

5.6. 异常处理

[强制] [JAVA075] 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。

> 说明：无法通过预检查的异常除外，比如，在解析字符串形式的数字时，不得不通过 catch NumberFormatException 来实现。

[强制] [JAVA076] 异常不要用来做流程控制，条件控制。

[强制] [JAVA077] catch 时请分清稳定代码和非稳定代码，稳定代码指的是无论如何不会出错的代码。 对于非稳定代码的 catch 尽可能进行区分异常类型，再做对应的异常处理。

[强制] [JAVA078] 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。

[强制] [JAVA079] 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。

[强制] [JAVA080] finally 块必须对资源对象、流对象进行关闭，有异常也要做 try-catch。 说明：如果 JDK7 及以上，可以使用 try-with-resources 方式。

[强制] [JAVA081] 不要在 finally 块中使用 return 。

[强制] [JAVA082] 捕获异常与抛异常，必须是完全匹配，或者捕获异常是抛异常的父类。

6.Java安全编码规范

详见：Java CMC安全类编码规范