Java技术管理文档

一、命名规范

a. 强制

1. 类名：使用UpperCamelCase风格，如JavaServerlessPlatform。

2. 方法名、参数名、成员变量、局部变量：统一驼峰命名，如localValue、getHttpMessage()。

3. 常量名：全部大写，单词间用下划线隔开，如MAX\_STOCK\_COUNT。

4. 抽象类命名：使用Abstract或Base开头，如AbstractTranslator。

5. 异常类命名：使用Exception结尾，如BusinessException。

6. 测试类命名：以要测试的类的名称开始，以Test结尾，如UserServiceTest。

7. 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，如com.alibaba.ai.util。

8. 避免在子父类的成员变量之间、或者不同代码块的局部变量之间采用完全相同的命名，以降低可读性。

9. 杜绝完全不规范的缩写，避免望文不知义，如AbsClass应改为AbstractClass。

10. 枚举类名：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，如ProcessStatusEnum的成员SUCCESS、UNKNOWN\_REASON。

11. 接口命名：如某Service接口，其实现类用Impl后缀；若是形容能力的带形容词-able的接口，实现类取该形容词的名词形式。

12. Controller层方法命名：

• 获取单个对象的方法用get做前缀。

• 获取对象列表的方法用list做前缀。

• 获取统计值的方法用count做前缀。

• 插入的方法用save做前缀。

• 删除的方法用delete做前缀。

• 修改的方法用update做前缀。

b. 推荐

13. 为了达到代码自解释的目标，任何自定义编程元素在命名时，使用尽量完整的单词组合来表达其意。

c. 允许

14. 类名复数形式：如果类名有复数含义，可以使用复数形式，但需保持一致性。

二、代码格式

a. 强制

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

16. 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，则左大括号前不换行，左大括号后换行，右大括号前换行，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。

17. 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。

18. 任何二目、三目运算符的左右两边都需要加一个空格。

19. 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。

20. 数组定义：类型与中括号紧挨相连，如int[] arrayDemo。

21. POJO类中布尔类型变量不要加is前缀，否则部分框架解析会引起序列化错误。

22. 避免在子父类的成员变量之间、或者不同代码块的局部变量之间采用完全相同的命名，以降低可读性。

23. 禁止使用可变参数进行功能开发。

24. 使用equals方法时，应使用常量或确定有值的对象来调用，避免抛空指针异常。

25. 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。

26. 所有的POJO类属性必须使用包装数据类型。

27. RPC方法的返回值和参数必须使用包装数据类型。

28. 所有的局部变量使用基本数据类型。

29. 定义DO/DTO/VO等POJO类时，不要设定任何属性默认值。

30. 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。

31. 不允许在程序任何地方中使用：java.sql.Date、java.sql.Time、java.sql.Timestamp。

32. 不要在程序中写死一年为365天，避免在公历闰年时出现日期转换错误或程序逻辑错误。

三、注释规范

a. 强制

33. 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。

b. 推荐

34. 注释应该增加代码的清晰度，保持注释的简洁，在写代码之前或同时写注释。注释出为什么做了一些事，而不仅仅是做了什么。

c. 允许

35. 对于简单的逻辑或显而易见的代码，可以省略注释，但需保持代码的自解释性。

四、文档管理

a. 强制

36. 定义清晰的文档结构：包括不同类型文档的存放位置、命名规则以及分类方式。例如，技术文档、会议记录、需求变更记录应分别存放，且每类文档应有统一的命名规则。

37. 制定统一的文档标准：包括文档的格式、模板、字体和大小等。

38. 利用版本控制工具（如Git、SVN）管理文档的版本，确保团队成员随时都能访问到最新的文档，同时也能查看文档的历史修改记录。

39. 设置合理的文档权限：不同的文档对应不同的访问权限，比如设计文档可能只对项目内部成员开放，而项目的需求文档可能需要与客户共享。

40. 建立文档的定期更新机制：比如每周或每月定期审查文档的更新情况，确保文档与项目实际相符。同时，应鼓励团队成员养成良好的文档维护习惯，任何项目进展和变更都应及时反映在相关文档中。

b. 推荐

41. 使用专业的文档管理工具（如Confluence、JIRA、GitLab等）来协作和版本控制文档，使团队成员能够方便地创建、编辑、审阅和共享文档。

c. 允许

42. 对于简单的项目或文档，可以使用文本编辑器或简单的文件共享系统进行管理，但需确保文档的版本控制和权限管理。