Java 技术管理规范文档

一、 命名规范

1. **A. 强制**: **类名**使用大驼峰命名法 (UpperCamelCase)。
    - `class RallyCar;`
2. **A. 强制**: **方法名、参数名、成员变量、局部变量**均使用小驼峰命名法 (lowerCamelCase)。
    - `void sendEmail(String toAddress);`
3. **A. 强制**: **常量 (static final)** 命名使用大写下划线风格 (CONSTANT_CASE)。
    - `static final int MAX_RETRIES = 3;`
4. **A. 强制**: **包名 (Package)** 全部小写，使用点分隔，且必须是唯一的。
    - `package com.mycompany.project.module;`
5. **A. 强制**: **泛型类型参数**使用单个大写字母。
    - `class List<T> { ... }`
6. **B. 推荐**: **测试类**的命名以 `Test` 结尾。
    - `class UserServiceTest { ... }`
7. **B. 推荐**: **布尔类型**的变量或方法名，使用 `is`、`has`、`can` 等作为前缀。
    - `boolean isReady();`
8. **C. 允许**: **JNI/Native 方法**可以为了匹配本地函数名而使用下划线。

二、 格式化

9. **A. 强制**: **使用大括号 `{}`**。`if`，`else`，`for`，`do`，`while` 语句，即使只有一行代码，也必须使用大括号。
10. **A. 强制**: **缩进**使用 4 个空格，禁止使用 Tab 字符。
11. **A. 强制**: **行长度**每行不超过 120 个字符。
12. **A. 强制**: **非空块**的左大括号 `{` 不换行，在其之前有一个空格。
    - `void myMethod() {`
13. **A. 强制**: **文件编码**统一使用 UTF-8。
14. **B. 推荐**: **方法间**用一个空行隔开，以增加可读性。
15. **B. 推荐**: **成员变量定义**和 **构造函数**之间用一个空行隔开。

16. **C. 允许**: 为了对齐，**允许**在变量声明时有多于一个的空格。
    - `int a = 1;`
    - `int bbb = 2;`

三、 注释

17. **A. 强制**: **公开的 (public) 类和方法**必须有 Javadoc 注释，描述其功能、参数、返回值和可能抛出的异常。
18. **A. 强制**: 禁止使用 **C 风格的块注释** (`/* ... */`)，除非是用于 Javadoc 或在单行内临时注释掉代码。
19. **B. 推荐**: 对**复杂或非直观的业务逻辑**添加必要的行内注释 (`// ...`)。
20. **B. 推荐**: 注释应解释“**为什么**”这么做，而不是“**做什么**”。代码本身应该能清晰地展示“做什么”。
21. **C. 允许**: 在开发阶段，**允许**使用 `// TODO:` 来标记未完成的功能或待办事项。

四、 编程实践

22. **A. 强制**: **禁止使用魔法值 (Magic Number)**。应将数字或字符串定义为常量。
23. **A. 强制**: **覆写 (`@Override`)**。只要是覆写父类或接口的方法，必须使用 `@Override` 注解。
24. **A. 强制**: **处理异常**。绝不能捕获异常后不做任何处理 (空 `catch` 块)。至少要记录日志。
25. **A. 强制**: **`equals()` 方法**。当覆写 `equals()` 方法时，必须同时覆写 `hashCode()` 方法。
26. **A. 强制**: **资源关闭**。所有实现了 `Closeable` 或 `AutoCloseable` 的资源（如流、连接等）必须在 `try-with-resources` 语句或 `finally` 块中确保关闭。
27. **A. 强制**: **禁止使用 `System.out` 或 `System.err`** 进行日志输出，应使用统一的日志框架（如 SLF4J）。
28. **B. 推荐**: **使用接口引用**。声明变量时应尽量使用接口类型，而不是具体的实现类。
    - `List<String> names = new ArrayList<>();` (推荐)
    - `ArrayList<String> names = new ArrayList<>();` (不推荐)

29. **B. 推荐**: **使用 `Optional`** 来避免返回 `null`。这使得 API 的使用者能明确地处理可能不存在的值。
30. **B. 推荐**: **优先使用 for-each 循环**，而不是传统的 for 循环，因为它更简洁且不易出错。
31. **B. 推荐**: **工具类 (`Utility Class`)** 的构造函数应声明为 `private`，以防止被实例化。
32. **B. 推荐**: **字符串拼接**优先使用 `StringBuilder` 或 `String.format()`，而不是 `+` 操作符，尤其是在循环中。
33. **B. 推荐**: **代码分层**。Controller/Resource 层、Service 层、Repository/DAO 层职责要清晰，禁止跨层调用。
34. **C. 允许**: 在性能要求极高的场景下，**允许**对代码进行深度优化，即使这会牺牲一定的可读性，但必须附带详尽的注释说明原因和优化效果。
35. **C. 允许**: **允许**使用反射，但仅限于框架或扩展点开发等必要场景，业务代码中应避免使用。

五、 依赖与并发

36. **A. 强制**: **依赖管理**。所有项目的外部依赖必须通过 Maven 或 Gradle 进行统一管理，禁止在项目中直接引入 JAR 包。
37. **A. 强制**: **禁止循环依赖**。模块或类之间不允许出现循环依赖。
38. **B. 推荐**: **明确依赖版本**。避免使用 `LATEST` 或 `SNAPSHOT` 作为生产环境的依赖版本。
39. **B. 推荐**: **创建线程**时，优先使用线程池 (`ExecutorService`)，而不是直接 `new Thread()`。
40. **B. 推荐**: **并发控制**。对于共享的可变数据，必须使用 `synchronized`、`Lock` 或 `java.util.concurrent` 包中的原子类来保证线程安全。
41. **C. 允许**: **允许**在简单的、一次性的脚本或工具中使用 `new Thread()`，但服务端应用中应避免。