Java技术管理规范

- 强制（必须遵守，否则代码不允许合入）

- 推荐（建议遵守，提升代码质量）

- 允许（允许的做法，视具体情况选择）

---

一、强制规则（Mandatory）

1. 代码必须遵守Google Java Style Guide的格式规范。

2. 类名采用大驼峰命名法（PascalCase），且必须具备描述性。

3. 方法名采用小驼峰命名法（camelCase），且动词开头。

4. 变量名采用小驼峰命名法，且具备语义。

5. 常量名必须全部大写，单词间用下划线分隔（UPPER\_SNAKE\_CASE）。

6. 每个类必须有类注释，说明类的功能和用途。

7. 公共方法必须有完整的Javadoc注释，包含参数、返回值和异常说明。

8. 禁止使用魔法数字，所有数字常量必须定义为常量。

9. 所有异常必须被捕获或声明抛出，禁止吞掉异常。

10. 禁止使用空的catch块，必须至少记录异常信息。

11. 代码中禁止出现System.out.println，必须使用日志框架（如SLF4J）。

12. 所有资源（如文件、数据库连接）必须在finally块或try-with-resources中关闭。

13. 禁止使用过时的API和方法。

14. 代码必须通过单元测试覆盖率不低于80%。

15. 所有提交的代码必须通过静态代码检查工具（如Checkstyle、PMD）检测。

16. Git提交信息必须规范，包含变更类型、简要描述和关联任务号。

17. 禁止在代码中硬编码配置，所有配置必须通过配置文件或环境变量注入。

18. 代码必须避免线程安全问题，必要时使用同步机制。

19. 禁止在业务逻辑中使用Thread.sleep()。

20. 所有公共接口必须设计合理的异常处理机制。

---

二、推荐规则（Recommended）

21. 建议使用Java 8及以上版本的新特性，如Lambda表达式和Stream API。

22. 建议使用Immutable对象减少副作用。

23. 建议方法长度不超过50行，保持单一职责。

24. 建议使用有意义的包结构，按功能模块划分。

25. 建议使用接口编程，依赖倒置原则。

26. 建议使用日志分级（DEBUG、INFO、WARN、ERROR）合理记录日志。

27. 建议使用单元测试框架（如JUnit、TestNG）进行测试。

28. 建议代码中避免深层嵌套，适当拆分方法。

29. 建议使用枚举类型替代常量字符串。

30. 建议使用Optional避免null值引发的空指针异常。

31. 建议使用Maven或Gradle进行项目构建和依赖管理。

32. 建议代码中注重性能优化，避免不必要的对象创建。

33. 建议使用设计模式提升代码复用性和可维护性。

34. 建议代码中注释清晰，避免冗余注释。

35. 建议定期进行代码重构，保持代码整洁。

---

三、允许规则（Allowed）

36. 允许使用第三方开源库，但必须经过安全和性能评估。

37. 允许在测试代码中使用System.out.println进行调试。

38. 允许使用多种日志框架，但需团队统一规范。

39. 允许在特殊情况下使用反射，但需注明原因。

40. 允许使用注解简化代码，但避免过度使用。

41. 允许在代码中使用TODO注释，但必须及时处理。

42. 允许使用多线程和并发工具，但需保证线程安全。

43. 允许在代码中使用断言（assert）进行调试。

44. 允许根据项目需求灵活调整代码风格，但需团队达成一致。

45. 允许在文档中使用中文或英文，但代码注释建议使用中文。

---