# Java 技术管理规范

## 1. 代码风格

A. 强制（必须遵守）

缩进：使用 4个空格，禁止使用Tab。

行宽：单行不超过 100字符，超长需换行并对齐。

大括号：即使单行语句也需使用 {}，如 if (condition) { doSomething(); }。

命名：

类名：PascalCase（如 UserService）。

方法/变量：camelCase（如 getUserName()）。

常量：UPPER\_SNAKE\_CASE（如 MAX\_COUNT）。

包名：全小写，禁止使用下划线（如 com.example.project）。

注释：公共API必须用Javadoc（如 /\*\* Retrieves user data. \*/）。

B. 推荐

空行：方法之间用1行空行，逻辑块之间可加空行提升可读性。

链式调用：若超出行宽，每行一个方法，并缩进对齐。

枚举：枚举值全大写，单词间用下划线（如 HTTP\_STATUS.OK）。

C. 允许（视情况选择）

单行注释：可用 // 或 /\* \*/，但需保持一致风格。

Lambda表达式：简单Lambda可单行，复杂逻辑需换行。

## 2. 编码实践

A. 强制

异常处理：禁止捕获异常后空处理（catch (Exception e) {}），至少打印日志。

资源关闭：使用 try-with-resources 管理 Closeable 资源。

不可变性：final 修饰的集合仍可修改内容，需用 Collections.unmodifiableList() 防御性拷贝。

Null检查：公共方法必须显式检查 null 或使用 @NonNull 注解。

字符串拼接：循环内用 StringBuilder，禁止直接 +。

B. 推荐

Optional：返回值可能为 null 时，返回 Optional<T>。

静态工厂方法：代替构造器，提供更清晰的创建逻辑（如 User.fromName()）。

防御性编程：对输入参数进行校验（如使用Guava的 Preconditions）。

C. 允许

方法长度：尽量不超过50行，但算法复杂时可适当放宽。

## 3. 面向对象设计

A. 强制

单一职责：一个类/方法只做一件事。

访问控制：成员变量默认 private，通过方法暴露必要访问。

继承慎用：优先组合而非继承，禁止为复用代码而继承。

B. 推荐

接口隔离：避免臃肿接口（如 UserService 拆分为 UserReader 和 UserWriter）。

依赖注入：使用Spring或手动DI，避免 new 直接创建依赖。

C. 允许

工具类：可包含静态方法，但需用 private 构造器防止实例化。

## 4. 并发与性能

A. 强制

线程安全：共享变量必须用 volatile、synchronized 或并发容器（如 ConcurrentHashMap）。

线程池：禁止直接 new Thread()，需使用线程池（如 ThreadPoolExecutor）。

B. 推荐

不可变对象：多线程环境下优先使用不可变类（如 String）。

异步编程：使用 CompletableFuture 代替回调地狱。

C. 允许

锁粒度：根据场景选择粗粒度锁（性能差但安全）或细粒度锁（如 ReadWriteLock）。

## 5. 测试与维护

A. 强制

单元测试：核心逻辑必须覆盖，命名规范为 被测类名 + Test（如 UserServiceTest）。

日志：使用SLF4J+Logback，禁止 System.out.println()。

B. 推荐

测试数据：使用Mock框架（如Mockito）隔离依赖。

代码覆盖率：关键模块行覆盖率≥80%。

C. 允许

集成测试：Spring项目可用 @SpringBootTest，但需控制执行时间。

## 6. 工具与工程化

A. 强制

版本控制：提交信息需清晰（如 feat: add user login），遵循Conventional Commits。

代码格式化：提交前用Spotless或Checkstyle自动格式化。

B. 推荐

静态分析：集成SonarQube检测代码坏味道。

依赖管理：用 dependencyManagement 统一版本（如Spring BOM）。

C. 允许

IDE：可用IntelliJ或Eclipse，但需统一代码模板。