# Java 技术管理规范

规范级别说明：  
- a. 强制（必须遵守，违反即 bug 或代码拒绝提交）  
- b. 推荐（遵循能提升质量和可维护性）  
- c. 允许（可灵活使用）

## 一、命名规范（10条）

* a. 强制

1. 类名、接口名使用 PascalCase（大驼峰），不得使用下划线。
2. 变量名、方法名使用 camelCase（小驼峰），不得使用下划线或大写字母开头。
3. 包名全小写，使用反域名结构，禁止出现大写字母和下划线。
4. 常量命名使用全大写 + 下划线分隔，例如 MAX\_COUNT。

* b. 推荐

1. 布尔变量以 is、has、can 开头，体现其真假特性。
2. 枚举值命名建议全大写，常量风格，如 ERROR\_CODE。
3. 测试类命名以 Test 结尾，与被测试类一致，如 UserServiceTest。

* c. 允许

1. 在局部代码中短生命周期的变量允许简写（如 i, j）。
2. 在 lambda 表达式中使用单字符变量（如 x -> x + 1）。
3. 生成代码工具自动生成的变量命名（如 MyBatis 逆向）可适当放宽。

## 二、代码格式与结构（10条）

* a. 强制

1. 禁止省略大括号 {}，即使代码块只有一行。
2. 每个语句必须独立一行，不得多个语句同一行。
3. 缩进为 4个空格，禁止使用 Tab。
4. if/else/for 等控制语句后必须换行开始代码块。

* b. 推荐

1. 超过120字符的代码行应换行，保持代码可读性。
2. 运算符左右各保留一个空格，增加代码清晰度。
3. 字段声明顺序为：静态常量 > 静态变量 > 实例变量。
4. 一个类只负责一个功能，超出请进行拆分。

* c. 允许

1. 构造器重载时可链式调用，但要避免构造器嵌套过深。
2. 在 POJO 类中自动生成的 getter/setter 可不写注释。

## 三、编码实践（20条）

* a. 强制

1. 禁止在代码中硬编码魔法值，须定义为常量或配置项。
2. 字符串比较必须使用 equals()，不可使用 ==。
3. 集合初始化必须指定容量，如 new HashMap<>(16)。
4. 禁止捕获顶层异常如 Throwable、Exception，必须精确捕获。
5. null 检查必须优先进行，防止空指针异常（NPE）。
6. 循环中禁止使用 System.out.println() 输出日志。
7. 必须使用日志框架（如 slf4j），严禁直接调用 System.out 或 e.printStackTrace()。
8. 禁止使用过时的 API，如 Vector、Hashtable、Date。

* b. 推荐

1. 使用 StringBuilder 替代 + 拼接字符串，尤其在循环中。
2. 业务异常和系统异常应区分处理，建议自定义异常类。
3. 优先使用 Java8+ 特性（如 Lambda、Stream、Optional）提升简洁性。
4. 构造器参数不超过5个，建议使用建造者模式（Builder）。
5. 禁止在 finally 中使用 return 或 throw，可能导致异常丢失。
6. 多线程并发建议使用线程池，禁止手动创建线程。
7. 使用枚举代替常量类中的一组常量，表达含义更清晰。

* c. 允许

1. 局部 try-catch 可用于处理容错逻辑，但应记录日志。
2. 适当使用 Java 注解简化模板代码，如 Lombok，但需在团队达成共识。
3. 工具类可定义为 final 类并私有化构造方法防止实例化。
4. 使用注解实现统一校验、切面日志，但注意执行顺序。

## 四、注释规范（5条）

* a. 强制

1. 类、方法必须添加 JavaDoc 注释，包括参数、返回值、异常说明。
2. 注释内容必须准确反映逻辑，不能出现误导性描述。

* b. 推荐

1. 复杂算法、关键流程使用单行或块注释解释逻辑。
2. 使用 // TODO、// FIXME 统一标记未完成内容，便于检索。

* c. 允许

1. 测试类中的注释可简略说明测试目的和断言点。

## 五、工程与安全规范（5条）

* a. 强制

1. 版本控制中禁止提交 .class、日志、临时配置等无关文件。
2. 线上环境严禁直接操作数据库，必须通过代码变更。
3. 接口层必须进行参数校验，避免非法入参导致系统崩溃。

* b. 推荐

1. 采用代码扫描工具（如 SonarQube）定期检查潜在问题。
2. 使用 @Slf4j 等注解统一日志风格，便于追踪分析。

* c. 允许

1. 在单元测试中使用 mock 框架如 Mockito 辅助测试，提高覆盖率。