# Java 编程技术管理规范文档

## 一、命名规范（Naming Conventions）

### a. 强制（必须遵守）

1. 类名使用**大驼峰式命名**，如：UserServiceImpl。
2. 方法名和变量名使用**小驼峰式命名**，如：getUserName()、userId。
3. 常量命名必须全大写，单词间使用下划线分隔，如：MAX\_RETRY\_TIMES。
4. 枚举值命名全部大写，并用下划线分隔，如：SUCCESS\_CODE。
5. 包名一律小写，不得使用下划线，例如：com.example.project。
6. 严禁使用拼音、谐音或无语义的命名（如 a1, b2, test1）。
7. Boolean 变量必须以 is, has, can, should 等前缀命名，例如：isEnabled。

### b. 推荐（强烈建议遵守）

1. DAO 层接口建议以 Mapper 或 Dao 结尾，如 UserMapper。
2. Service 接口建议以 Service 结尾，如 OrderService。
3. Controller 层类建议以 Controller 结尾，如 UserController。
4. 避免使用重复含义的词，如 userInfoDataBean。
5. 变量命名不宜过长，但需表达清晰，例如：orderAmount 优于 oa。
6. 使用有意义的名字，表达对象用途，如 startTime, userList。

### c. 允许（在合理场景下可使用）

1. 临时变量可使用 i, j, k 等形式，但仅限于循环体。
2. 私有变量名前可使用 \_ 前缀，但需团队内部统一。

## 二、代码格式规范（Code Formatting）

### a. 强制

1. 使用 **4 个空格**进行缩进，不得使用 Tab。
2. 每行代码长度不得超过 **120 个字符**。
3. 类中只允许一个 public 类，类名与文件名一致。
4. 类、方法、属性之间需空一行进行分隔。
5. 左大括号 { 必须另起一行。
6. 方法参数过多需换行对齐，逗号结尾。
7. 类内成员变量顺序：静态常量 > 静态变量 > 成员变量 > 构造方法 > 方法。

### b. 推荐

1. 相似逻辑代码块之间添加空行，提高可读性。
2. 工具类使用 final 修饰，并使用私有构造函数阻止实例化。
3. 对于需要继承的类，不添加 final 关键字；否则应加以限制。

### c. 允许

1. 可以使用 IDE 自动格式化，但需保证统一规范（推荐使用 Prettier、Google Java Format 等）。

## 三、注释规范（Commenting Guidelines）

### a. 强制

1. 所有公共类、接口、方法必须使用 Javadoc 注释，包含方法用途、参数、返回值、异常等信息。
2. 所有复杂算法或业务逻辑代码必须进行块注释说明。
3. 严禁出现无意义注释（如 // TODO: 而无说明）。
4. 禁止使用中文拼音作为注释。

### b. 推荐

1. 注释内容应简明扼要，语言通顺，避免冗长。
2. 控制语句（如 if, for）块前推荐简要注释说明逻辑。

### c. 允许

1. 临时性注释（如调试信息）可使用 // DEBUG: 标记，但提交代码前必须清理。

## 四、面向对象设计规范（OOP Best Practices）

### a. 强制

1. 禁止类继承超过两层，推荐组合优于继承。
2. 所有成员变量必须 private，通过 getter/setter 方法访问。
3. 禁止在构造方法中调用可以被重写的方法。
4. 接口定义中不得包含实现代码（除非是 default 方法）。

### b. 推荐

1. 接口命名体现行为，例如：Runnable、Serializable。
2. 成员变量如非必须，应用 final 修饰以防止被修改。
3. 小对象传递时建议使用对象封装（DTO），而不是多个基本类型。

### c. 允许

1. 枚举中可包含方法，但应保持轻量。

## 五、异常与日志处理规范（Exception & Logging）

### a. 强制

1. 禁止使用 Exception 或 Throwable 进行 catch，必须具体明确。
2. catch 块中不得为空，需进行异常处理或日志记录。
3. 日志输出必须使用统一日志框架（如 SLF4J），禁止 System.out.println()。
4. 所有方法抛出的异常必须在方法签名中声明或使用自定义异常包装。

### b. 推荐

1. 日志应包含关键信息：操作类型、参数、上下文 ID、时间戳等。
2. 异常类命名统一以 Exception 结尾，如 UserNotFoundException。
3. 日志级别合理使用：ERROR（错误） > WARN（告警） > INFO（信息） > DEBUG（调试）。

### c. 允许

1. DEBUG 日志中可记录部分敏感数据，但需脱敏处理（如手机号加星号处理）。

## 六、其他规范（Miscellaneous）

### a. 强制

1. 所有提交的代码必须通过构建与单元测试。
2. 禁止提交 IDE 配置文件、临时调试文件到仓库。
3. 代码库必须配有 README.md 与项目结构说明文档。
4. 所有提交必须经过 Code Review，并附详细提交说明。

### b. 推荐

1. 分支命名规范：feature/xxx、bugfix/xxx、release/xxx。
2. 项目根目录下应有 .editorconfig 或 .prettierrc 等统一代码风格配置。

### c. 允许

1. 单测中可适当使用 mock 数据，但需使用 Mock 框架如 Mockito、JMockit。

## 附录

* 可选工具：
  1. IDE：IntelliJ IDEA（推荐配置 Google Java Code Style 插件）
  2. Lint 工具：Checkstyle、PMD、SpotBugs
  3. 格式化工具：Google Java Format、Prettier