**Java 技术管理规范文档（v1.0）**

**一、命名规范**

1. （a）类名必须使用大驼峰命名法（PascalCase），如 UserService。
2. （a）变量名和方法名必须使用小驼峰命名法（camelCase），如 userName、getUserInfo()。
3. （a）常量名必须全大写，单词之间用下划线连接，如 MAX\_RETRY\_COUNT。
4. （b）包名统一使用小写字母，禁止使用下划线，推荐使用公司域名倒序 + 模块名，如 com.example.user。
5. （b）接口命名建议以I开头或以Service结尾，如 IUserService、UserService。
6. （c）枚举名使用大驼峰，枚举值全大写加下划线，如 SUCCESS, NOT\_FOUND。

**二、代码格式**

1. （a）每行代码长度不得超过 120 个字符，超出需换行。
2. （a）使用 4 个空格缩进，不允许使用制表符（Tab）。
3. （a）花括号必须另起一行，并与控制语句对齐，如：
4. （b）类、方法之间空一行，增强代码可读性。
5. （b）方法参数超过三个时，建议换行对齐编写。
6. （c）允许使用 IDE 自动格式化工具，如 IDEA 的 Code Style。

**三、注释规范**

1. （a）所有公共类和方法必须有 Javadoc 注释。
2. （a）禁止无意义注释，如 // 这是一个构造方法。
3. （b）复杂逻辑代码块必须写清楚注释，描述“为什么这样做”。
4. （b）TODO 注释必须带上处理人和时间，如 // TODO: 优化查询逻辑 @张三 2025-06-21。
5. （c）允许使用 @param、@return、@throws 说明方法参数和返回值。

**四、异常与日志**

1. （a）不允许捕获 Exception 或 Throwable 做通用处理，必须精准捕获。
2. （a）严禁空 catch 块，必须记录日志或采取措施。
3. （b）使用自定义业务异常，如 BusinessException。
4. （b）日志使用 slf4j，变量使用占位符方式：log.info("User id: {}", userId);。
5. （b）异常信息应包含上下文信息，便于定位。
6. （c）允许在调试期间临时使用 e.printStackTrace()，上线前必须移除。

**五、面向对象设计**

1. （a）禁止神类（God Class），单个类代码行数不得超过 800 行。
2. （a）禁止循环依赖，禁止多重继承（Java 不允许，但可通过组合方式扩展）。
3. （b）推荐使用接口 + 实现类组合方式。
4. （b）建议使用 Lombok，但必须控制在少量实体类中使用（避免代码不清晰）。
5. （c）允许使用设计模式，如单例、工厂、策略等，但需结合实际业务合理使用。

**六、集合与控制结构**

1. （a）优先使用接口作为类型声明，如 List<String> list = new ArrayList<>();。
2. （a）禁止使用 == 比较字符串，必须使用 equals() 或 Objects.equals()。
3. （a）禁止出现魔法值，必须定义为常量。
4. （b）使用增强 for 循环或 Stream API 替代传统 for 循环。
5. （b）集合初始化时应预估大小，避免扩容。
6. （c）允许使用 switch，但推荐使用 enum 代替 magic number。

**七、线程与并发**

1. （a）禁止直接使用 Thread 创建线程，应使用线程池（如 Executors、ThreadPoolExecutor）。
2. （a）共享变量需使用 volatile 或 synchronized 保证线程安全。
3. （b）推荐使用 CompletableFuture、CountDownLatch 等高阶并发工具。
4. （b）线程池应统一管理，禁止随意创建。
5. （c）允许使用异步注解（如 @Async）处理非核心流程。

**八、数据库与持久层**

1. （a）DAO 层禁止写业务逻辑，只能进行数据库操作。

**九、接口与服务设计**

1. （a）公共接口必须使用 @RequestMapping、@GetMapping 等明确声明请求方式。
2. （a）接口返回值必须统一封装，如使用 Result<T> 或 ResponseEntity<T>。
3. （b）推荐使用 RESTful 风格，避免冗余前缀如 /getUserInfo.do。
4. （b）建议接口设计采用幂等性原则（GET 查询不改动数据、POST 提交可重复）。
5. （c）允许将 GET 用于分页、筛选操作，但不应修改数据状态。

**十、配置与环境隔离**

1. （a）禁止将敏感信息写死在代码中，如数据库密码、密钥等。
2. （a）必须区分开发、测试、生产环境配置，使用 application-{env}.yml 管理。
3. （b）建议使用配置中心（如 Nacos、Apollo）管理配置。
4. （b）日志级别应在不同环境合理设置，生产环境禁止 DEBUG。
5. （c）允许使用 .env 文件临时配置，但不应提交到 Git。

**十一、版本与依赖管理**

1. （a）所有模块使用统一的 JDK 版本（如 Java 17），写入 pom.xml 管理。
2. （a）依赖必须通过 Maven/Gradle 管理，禁止手动引入 jar 包。
3. （b）建议使用依赖管理插件（如 spring-boot-dependencies）统一版本。
4. （b）定期排查依赖冲突和版本更新，避免 CVE 风险。
5. （c）允许使用本地安装的私服或缓存依赖（如 Nexus、阿里云镜像）加快构建速度。

**十二、安全规范**

1. （a）用户密码等敏感信息必须加密存储，禁止明文入库。
2. （a）所有对外接口必须验证身份（JWT、Session、Token、OAuth2 等）。
3. （b）建议对重要操作添加权限控制注解（如 @PreAuthorize）。
4. （b）输入数据必须做参数校验（如使用 @Validated、@NotBlank 等）。
5. （c）允许使用 XSS、SQL 注入安全检测工具定期扫描项目。