### **Java 技术管理规范文档（精简版）**

以下是按照强制、推荐、允许三个级别整理的 ****40 条核心 Java 规范****，参考大厂标准并确保实用性：

#### ****一、代码风格规范****

****a、强制（1-15）****

****命名规范****

1.类名使用大驼峰（如UserService），方法 / 变量使用小驼峰（如getUserName）。

2.常量全部大写，单词间用下划线分隔（如MAX\_PAGE\_SIZE）。

3.包名统一使用小写字母，多级包用点分隔（如com.example.utils）。

****代码格式****

1.使用 4 个空格缩进，禁止使用 Tab。

2.左大括号不换行，右大括号单独占一行（K&R 风格）。

3.每行代码不超过 120 个字符，超长时需合理换行。

****注释规范****

1.类和公共方法必须使用 Javadoc 注释，说明功能、参数和返回值。

2.复杂逻辑代码块必须添加行注释（如算法实现、边界条件处理）。

****空行与分隔****

1.方法之间用空行分隔，逻辑相关的代码块之间可不空行。

2.类的成员变量集中声明在类顶部，不穿插方法定义。

****异常处理****

1.禁止捕获通用异常（如Exception），必须指定具体异常类型。

2.受检异常（Checked Exception）必须显式处理或声明抛出。

****访问控制****

1.类成员变量默认使用private修饰，通过公共方法访问。

2.工具类应声明为final，构造方法私有化。

****泛型使用****

1.集合类必须指定泛型类型（如List<String>），禁止使用原生态类型。

****字符串处理****

1.禁止在循环中使用+拼接字符串，必须使用StringBuilder。

****常量定义****

1.常量必须使用static final修饰，禁止魔法值（如if (status == 1)）。

****方法设计****

1.方法参数不超过 5 个，超过时使用对象封装。

2.禁止方法有过多职责（单一职责原则）。

****枚举使用****

1.枚举类名使用Enum后缀（如StatusEnum），成员名称全部大写。

****日期处理****

1.禁止使用Date和SimpleDateFormat，必须使用java.time包下的类（如LocalDateTime）。

****集合操作****

1.遍历集合时优先使用增强for循环或 Stream API。

2.集合初始化时尽量指定初始容量（如new ArrayList<>(10)）。

****日志规范****

1.使用 SLF4J 接口记录日志，禁止直接使用System.out.println。

****序列化****

1.实现Serializable接口的类必须定义serialVersionUID。

****b、推荐（16-30）****  
16. ****Lambda 表达式****

* 优先使用 Lambda 替代匿名内部类（如list.forEach(item -> System.out.println(item))）。

****Stream API****

* + 复杂集合操作优先使用 Stream（如过滤、映射、聚合）。

****Optional 类****

* + 方法返回值可能为null时，推荐使用Optional包装。

****单元测试****

* + 新增方法覆盖率不低于 80%，核心业务逻辑覆盖率不低于 90%。
  + 使用 JUnit 5 和 AssertJ 编写测试用例。

****依赖注入****

* + 使用 Spring 框架时，优先通过构造函数注入依赖。

****事务管理****

* + 事务方法应足够短小，避免大事务导致数据库锁超时。

****异步处理****

* + 耗时操作推荐使用CompletableFuture或消息队列异步处理。

****性能优化****

* + 避免在循环中创建新对象，可在循环外初始化。
  + 集合频繁删除插入操作时使用LinkedList，随机访问优先用ArrayList。

****配置管理****

* + 外部配置（如数据库连接信息）必须通过配置文件或环境变量注入。

****代码复用****

* + 提取公共逻辑为工具类或通用组件，避免代码重复。

****接口设计****

* + 接口参数使用对象封装，避免大量参数平铺。

****线程安全****

* + 多线程环境下优先使用线程安全的集合类（如ConcurrentHashMap）。

****SQL 操作****

* + 禁止 SQL 拼接，必须使用参数化查询（如PreparedStatement）。

****缓存使用****

* + 高频读场景推荐使用缓存（如 Redis），并设置合理过期时间。

****代码审查****

* + 所有代码变更必须经过至少 1 人代码审查。

****c、允许（31-40）****  
31. ****单例模式****

* 允许使用静态内部类或枚举实现单例，不强制使用双重检查锁。

****反射使用****

* + 允许在框架开发或工具类中使用反射，但需做好性能优化。

****原生类型与包装类****

* + 允许根据场景选择原生类型（如int）或包装类（如Integer）。

****内部类****

* + 允许使用内部类封装高度内聚的逻辑，但需控制类的复杂度。

****自定义注解****

* + 允许在需要 AOP 增强的场景下自定义注解（如权限校验）。

****序列化方式****

* + 允许使用 JSON（如 Jackson）替代 Java 原生序列化。

****日志级别****

* + 允许根据环境调整日志级别（如生产环境关闭 DEBUG 日志）。

****异常捕获范围****

* + 允许在工具类中捕获Exception并记录日志，但需谨慎处理。

****方法重载****

* + 允许合理使用方法重载，但需避免参数类型相似导致的调用歧义。

****第三方库****

* + 允许引入经过安全审计的第三方库，但需控制依赖数量。

### ****二、开发流程规范****

****a、强制****

* 分支管理使用 Git Flow 或 GitHub Flow，禁止直接提交到主分支。
* 提交代码前必须通过静态代码检查（如 SonarQube）。

****b、推荐****

* 采用 TDD（测试驱动开发）模式编写核心业务代码。
* 使用自动化部署工具（如 Jenkins、GitLab CI）。

****c、允许****

* 允许在紧急修复时简化代码审查流程，但事后需补充审查。

### ****三、版本与依赖管理****

****a、强制****

* 使用语义化版本号（如1.0.0）。
* 禁止引入 SNAPSHOT 版本的依赖。

****b、推荐****

* 统一管理依赖版本（如 Maven 的dependencyManagement）。

****c、允许****

* 允许在测试环境使用快照版本加速开发。

### ****四、安全规范****

****a、强制****

* 禁止明文存储密码，必须使用加密算法（如 BCrypt）。
* SQL 查询必须防止 SQL 注入（如使用 ORM 框架）。

****b、推荐****

* 敏感信息（如身份证号）传输时使用 HTTPS。

****c、允许****

* 允许在开发环境使用 HTTP 以简化配置。

### ****五、性能规范****

****a、强制****

* 禁止在循环中执行数据库查询，必须批量操作。

****b、推荐****

* 大文件读写使用 NIO 替代传统 IO。

****c、允许****

* 允许在非核心业务中使用简单实现，后续再优化。

以上规范可根据团队规模和项目特点进行调整，建议每季度组织一次规范评审会，确保规范的有效性和可执行性。