**1. 命名规范**

**a. 强制**

1. 类名使用大驼峰法，如：UserService。
2. 变量、方法名使用小驼峰法，如：userName、getUser()。
3. 常量名使用全大写字母加下划线，如：MAX\_VALUE。
4. 包名全部小写，避免使用下划线，如：com.example.service。
5. 枚举类型名称使用大驼峰，枚举常量使用全大写，如：enum Status { SUCCESS, ERROR }。
6. 不允许使用拼音或缩写作为命名，例如：int sj = 0 禁止。
7. boolean变量使用 is 开头命名，如：isAvailable。

**b. 推荐**

1. 接口名建议以 I 开头或以 able、ible 结尾，如：IUserService 或 Serializable。
2. 异常类命名以 Exception 结尾，如：UserNotFoundException。
3. DAO 层类名建议以 Mapper 或 Dao 结尾，如：UserMapper。
4. 测试类名以被测试类名 + Test 命名，如：UserServiceTest。

**c. 允许**

1. 在极特殊情况（如与第三方对接）可适度使用缩写，但必须注明含义。

**2. 代码风格**

**a. 强制**

1. 使用 4 个空格缩进，禁止使用 Tab。
2. 每个语句占一行，控制语句必须加大括号 {}。
3. if、for、while 后必须有空格：if (condition) {}。
4. import语句按包路径排序，不允许使用 \* 通配符。
5. 每个类文件只能有一个 public 类，类名与文件名必须相同。
6. 所有类必须有类注释，说明类的作用和作者信息。

**b. 推荐**

1. 空行的使用应恰到好处，逻辑段落之间空一行。
2. 每个方法不超过 80 行，类不超过 500 行。
3. 方法内部代码应尽量避免深层嵌套（不超过三层）。

**c. 允许**

1. 可在工具类中使用静态导入，但应避免滥用。

**3. 面向对象设计**

**a. 强制**

1. 不允许出现 God 类（职责过多的类），每个类应单一职责。
2. 成员变量使用 private 修饰，提供 getter/setter 方法。
3. 构造函数中不应包含业务逻辑。
4. 禁止循环依赖。
5. 严禁滥用继承，应优先使用组合。

**b. 推荐**

1. 抽象类中尽量只定义模板方法，不实现具体逻辑。
2. 接口设计应小而精，一个接口不超过 5 个方法。

**c. 允许**

1. 内部类可用于只在一个类中使用的结构，但不推荐深度嵌套。

**4. 异常与日志**

**a. 强制**

1. 不允许捕获 Exception 或 Throwable，应捕获具体异常。
2. 所有异常必须记录日志或向上抛出处理，禁止空 catch。
3. 日志使用统一工具（如 SLF4J），禁止使用 System.out.println。
4. 日志内容中不得包含敏感信息，如密码、身份证号。

**b. 推荐**

1. 自定义异常应继承自 RuntimeException。
2. 日志记录格式统一：[模块] 操作 - 描述：{关键数据}。

**c. 允许**

1. 可在调试阶段临时输出日志，但需在提交前删除或转为 debug 级别。

**5. 并发编程与性能**

**a. 强制**

1. 禁止使用过期的线程类（如 Thread.stop()），统一使用线程池。
2. 多线程共享变量必须使用 volatile 或同步机制保证可见性。

**b. 推荐**

1. 并发场景优先使用 Concurrent 包下的类，如 ConcurrentHashMap。
2. 不推荐使用 synchronized 修饰静态方法，避免锁竞争。

**c. 允许**

1. 性能优化可在特定关键模块中使用 unsafe 操作，但需评审通过。

**6. 其他开发约定**

**a. 强制**

1. 所有提交代码必须通过编译检查与静态分析。
2. 重要变更必须通过 Code Review。
3. 所有方法和类必须有必要的 JavaDoc。

**b. 推荐**

1. 单元测试覆盖率不低于 80%，公共方法必须有单测。
2. 对外接口文档使用标准工具生成（如 Swagger）。

**c. 允许**

1. 对于临时或实验代码，可注释 // TODO，需说明处理计划和责任人。