目录

[一、 Must 2](#_Toc201339851)

[1.1命名规范 2](#_Toc201339852)

[1.2代码风格 2](#_Toc201339853)

[1.3文档注释 2](#_Toc201339854)

[1.4集合使用 2](#_Toc201339855)

[1.5并发 2](#_Toc201339856)

[1.6注解 3](#_Toc201339857)

[1.7序列化 3](#_Toc201339858)

[1.8性能 3](#_Toc201339859)

[1.9安全 3](#_Toc201339860)

[1.10依赖管理 3](#_Toc201339861)

[1.11中文编码方式 3](#_Toc201339862)

[1.12依赖管理 3](#_Toc201339863)

[1.13泛型 3](#_Toc201339864)

[1.14泛型 3](#_Toc201339865)

[二、 Should 3](#_Toc201339866)

[2.1包划分 3](#_Toc201339867)

[2.2新特性 4](#_Toc201339868)

[2.3静态分析 4](#_Toc201339869)

[2.4日志管理 4](#_Toc201339870)

[2.5容器化部署 4](#_Toc201339871)

[三、 May 4](#_Toc201339872)

[3.1元编程 4](#_Toc201339873)

[3.2编译期检查 4](#_Toc201339874)

[3.3文档自动化 4](#_Toc201339875)

[3.4 AI 辅助 4](#_Toc201339876)

[3.5 开源 4](#_Toc201339877)

[3.6 AI 辅助跨语言互操作 4](#_Toc201339878)

1. Must

## 1.1命名规范

包名：全小写，使用反向域名（如 com.myname.project）。

类/接口名：PascalCase（大驼峰）。

方法/变量名：camelCase（小驼峰）。

常量：大写+下划线（如 MAX\_SIZE）。

## 1.2代码风格

统一缩进（4 空格）和行宽（不超过 120 字符）。

使用 Google Java Format 或 Spotless 自动格式化

## 1.3文档注释

公共 API 必写完整 Javadoc，包含说明、参数、返回值、异常。

核心算法适当注释，避免注释显而易见的逻辑。

CI 中检查 Javadoc 完整性。

## 1.4集合使用

根据场景选合适实现（ArrayList vs LinkedList），并预设初始容量。

并发场景用线程安全集合或外部同步。

## 1.5并发

优先用 java.util.concurrent 高级 API（ExecutorService、CompletableFuture 等）。

避免手写 while(true)+Thread.sleep()。

共享状态用最小锁或原子变量。

## 1.6注解

合理用框架注解（如 Spring、验证注解）

## 1.7序列化

避免 Java 原生序列化；优先 JSON/Protobuf。

## 1.8性能

避免频繁对象创建，注意装箱。

## 1.9安全

参数/输入校验、防注入。

## 1.10依赖管理

Maven/Gradle 指定版本并锁定。

## 1.11中文编码方式

统一使用UTF-8

## 1.12依赖管理

避免硬编码环境配置，使用参数化或 profiles。

## 1.13泛型

避免原始类型，使用带边界的泛型。

尽量减少类型转换。

## 1.14泛型

前后端交互统一用json数据封装

1. Should

## 2.1包划分

按功能或领域分层（如 com.myname.project.user、com.myname.project.order）。

## 2.2新特性

在评估过后使用 Java 的记录类（Record）、流式 API（Stream）等。

## 2.3静态分析

引入 Checkstyle、SpotBugs；在 IDE 或 CI 中实时反馈。

## 2.4日志管理

生产环境 INFO/WARN/ERROR 分级，避免 DEBUG 泛滥。

## 2.5容器化部署

多阶段构建瘦镜像，JVM 参数调优。

1. May

## 3.1元编程

使用注解处理器或代码生成工具（如 Lombok、MapStruct）减少样板代码。

## 3.2编译期检查

引入 Checker Framework、Error Prone 等进行更严格的编译期校验。

## 3.3文档自动化

使用 PlantUML、Swagger UI 自动生成类图、序列图和 API 文档。

## 3.4 AI 辅助

尝试 AI 驱动的代码补全和审查工具，提高开发效率。

## 3.5 开源

参与社区、维护项目模板，提升影响力。

## 3.6 AI 辅助跨语言互操作

通过 JNI/JNA、gRPC/REST 等方式与非 JVM 语言系统交互。