下半期作业 郑鸿 2022141461086

选择JAVA

**1. 代码风格规范**

**a. 强制**

1. **文件编码**：必须使用 UTF-8 编码
2. **命名规范**：
   * 类名使用大驼峰式 (UpperCamelCase)
   * 方法名、变量名使用小驼峰式 (lowerCamelCase)
   * 常量名全部大写，单词间用下划线分隔 (UPPER\_CASE)
3. **大括号**：必须使用 K&R 风格 (Egyptian brackets)

java

复制

下载

*// 正确*

if (condition) {

*// ...*

}

*// 错误*

if (condition)

{

*// ...*

}

1. **缩进**：使用 4 个空格，禁止使用 Tab
2. **每行长度**：不超过 120 个字符

**b. 推荐**

1. **包命名**：使用公司域名的反写 (如 com.example.project)
2. **方法长度**：单个方法不超过 50 行
3. **类长度**：单个类不超过 500 行
4. **空行**：方法之间用空行分隔，逻辑相关的代码块之间用空行分隔

**c. 允许**

1. **行尾注释**：简单的单行注释可以放在代码行尾
2. **链式调用换行**：链式调用可以按照团队习惯选择是否换行

java

复制

下载

*// 两种方式都可以*

builder.setA(a).setB(b).build();

builder.setA(a)

.setB(b)

.build();

**2. 语言特性使用规范**

**a. 强制**

1. **异常处理**：
   * 禁止捕获异常后不做任何处理
   * 禁止捕获 **Throwable** 或 **Exception** 而不处理特定异常
2. **资源管理**：必须使用 try-with-resources 管理所有实现 **AutoCloseable** 的资源

java

复制

下载

*// 正确*

try (InputStream is = new FileInputStream("file")) {

*// ...*

}

1. **equals 和 hashCode**：重写 **equals()** 必须同时重写 **hashCode()**
2. **可变性**：工具类和常量类必须声明为 final

**b. 推荐**

1. **Optional**：方法返回可能为 null 的值时，推荐返回 **Optional<T>**
2. **Lambda**：简单的函数式操作推荐使用 Lambda 表达式
3. **不可变集合**：优先使用 **Collections.unmodifiableXXX** 返回不可变集合
4. **记录类**：Java 14+ 中，纯数据类推荐使用 record 类型

**c. 允许**

1. **Stream API**：复杂的流操作允许拆分为多步操作以提高可读性
2. **方法引用**：允许在适当场景使用方法引用替代 Lambda
3. **var**：局部变量类型推断 (Java 10+) 允许在明显类型场景下使用

**3. 工程结构规范**

**a. 强制**

1. **目录结构**：必须遵循 Maven/Gradle 标准目录结构

text

复制

下载

src/main/java

src/main/resources

src/test/java

src/test/resources

1. **包结构**：必须按照功能而非层级划分 (如 com.example.order, com.example.product)
2. **测试代码**：测试类必须与被测试类位于相同包下，命名以 Test 结尾

**b. 推荐**

1. **模块划分**：推荐按功能模块划分子工程
2. **API 分离**：推荐将 API 定义与实现分离
3. **DTO 分离**：推荐将数据传输对象与领域对象分离

**c. 允许**

1. **工具类位置**：允许将通用工具类放在公共模块中
2. **配置位置**：允许将配置类放在顶层包或专门的 config 包中

**4. 依赖管理规范**

**a. 强制**

1. **版本管理**：必须使用依赖管理工具 (Maven/Gradle) 统一管理版本
2. **冲突解决**：必须解决依赖冲突，禁止存在多个版本的同一依赖
3. **范围定义**：必须明确定义依赖范围 (compile, runtime, test 等)

**b. 推荐**

1. **依赖最少化**：推荐保持最小依赖原则
2. **版本升级**：推荐定期升级依赖版本
3. **依赖分析**：推荐使用工具分析依赖关系 (如 mvn dependency:tree)

**c. 允许**

1. **可选依赖**：允许在必要时使用 optional 依赖
2. **本地依赖**：允许在开发阶段使用本地依赖，但必须提供替代方案

**5. 测试规范**

**a. 强制**

1. **单元测试**：所有核心逻辑必须包含单元测试
2. **测试覆盖率**：核心模块行覆盖率必须 ≥80%
3. **测试独立性**：测试用例必须相互独立，不依赖执行顺序

**b. 推荐**

1. **测试命名**：推荐使用 Given-When-Then 模式命名测试方法

java

复制

下载

@Test

void givenInvalidInput\_whenProcess\_thenThrowException() {

*// ...*

}

1. **断言库**：推荐使用 AssertJ 等流式断言库
2. **Mock 使用**：推荐在适当场景使用 Mock 而非真实依赖

**c. 允许**

1. **集成测试**：允许编写少量集成测试验证系统整体行为
2. **测试数据**：允许使用测试数据生成工具

**6. 安全规范**

**a. 强制**

1. **敏感数据**：禁止在日志中输出敏感信息
2. **反序列化**：必须对反序列化操作进行安全控制
3. **SQL 注入**：必须使用预编译语句或 ORM 框架防止 SQL 注入

**b. 推荐**

1. **密码存储**：推荐使用 BCrypt 等安全算法存储密码
2. **HTTPS**：推荐所有外部调用使用 HTTPS
3. **依赖扫描**：推荐使用 OWASP Dependency-Check 扫描依赖漏洞

**c. 允许**

1. **安全框架**：允许使用 Spring Security 等成熟安全框架
2. **加密库**：允许使用 JCA/JCE 提供的高强度加密算法

**7. 性能规范**

**a. 强制**

1. **连接管理**：必须正确关闭数据库连接、文件流等资源
2. **集合初始化**：已知大小的集合必须指定初始容量

java

复制

下载

*// 正确*

List<String> list = new ArrayList<>(expectedSize);

**b. 推荐**

1. **字符串拼接**：推荐使用 StringBuilder 进行循环内字符串拼接
2. **缓存使用**：推荐对频繁访问的只读数据使用缓存
3. **日志级别**：推荐合理使用日志级别，避免生产环境输出 DEBUG 日志

**c. 允许**

1. **对象池**：允许对创建成本高的对象使用对象池
2. **并行流**：允许在适当场景使用并行流提高性能

**8. 文档与注释**

**a. 强制**

1. **公共 API**：所有公共类、方法必须包含 Javadoc 注释
2. **复杂逻辑**：非显而易见的复杂逻辑必须包含注释说明
3. **TODO/FIXME**：必须包含负责人和预计解决时间

**b. 推荐**

1. **接口文档**：推荐使用 Swagger 等工具生成 API 文档
2. **变更记录**：推荐在类注释中维护重要变更记录
3. **示例代码**：推荐为复杂用法提供示例代码

**c. 允许**

1. **实现注释**：允许在方法内部添加实现细节注释
2. **链接引用**：允许在注释中添加相关文档链接

**9. 版本与发布**

**a. 强制**

1. **版本控制**：必须使用语义化版本控制 (SemVer)
2. **兼容性**：必须明确标注破坏性变更
3. **发布检查**：必须通过所有自动化检查才能发布

**b. 推荐**

1. **变更日志**：推荐维护详细的变更日志 (CHANGELOG.md)
2. **发布计划**：推荐制定并遵循发布计划
3. **回滚方案**：推荐为每个发布准备回滚方案

**c. 允许**

1. **热修复**：允许在紧急情况下使用热修复流程
2. **特性开关**：允许使用特性开关控制新功能发布

**10. 工具与自动化**

**a. 强制**

1. **代码格式化**：必须使用统一配置的 IDE 格式化工具
2. **静态分析**：必须通过 Checkstyle/PMD/SonarQube 等静态分析工具检查
3. **CI/CD**：必须配置持续集成流水线

**b. 推荐**

1. **代码模板**：推荐使用统一的代码模板文件
2. **依赖更新**：推荐配置自动化依赖更新检查
3. **构建缓存**：推荐配置构建缓存加速构建过程

**c. 允许**

1. **本地钩子**：允许配置 Git 预提交钩子进行本地检查
2. **定制工具**：允许开发团队内部定制工具