下半期作业

命名规范

**规范 01**

**级别**：a（强制）  
**名称**：类名使用 PascalCase  
**规范内容**：类名必须使用 PascalCase 命名法，首字母大写，多个单词每个首字母大写，不允许使用下划线或缩写。  
**示例代码**：

public class UserService {} // 正确

public class user\_service {} // 错误

public class USERSERVICE {} // 错误

**规范 02**

**级别**：a（强制）  
**名称**：变量名使用 camelCase  
**规范内容**：变量名应使用 camelCase，首字母小写，后续单词首字母大写。  
**示例代码**：

String userName = "Tom"; // 正确

String User\_Name = "Tom"; // 错误

**规范 03**

**级别**：a（强制）  
**名称**：常量命名使用全大写加下划线  
**规范内容**：常量命名必须使用全大写字母，单词间使用下划线分隔。  
**示例代码**：

public static final int MAX\_RETRY\_COUNT = 5; // 正确

public static final int maxRetryCount = 5; // 错误

**规范 04**

**级别**：a（强制）  
**名称**：包名使用全小写，不包含下划线  
**规范内容**：Java 包名统一使用小写字母，禁止使用大写字母或下划线。  
**示例代码**：

java

复制编辑

package com.example.service; // 正确

package com.Example\_Service; // 错误

**规范 05**

**级别**：b（推荐）  
**名称**：布尔变量以 is/has/can 开头  
**规范内容**：boolean 类型变量推荐以 is、has、can 等动词开头，表达更清晰。  
**示例代码**：

boolean isValid = true; // 正确

boolean valid = true; // 不推荐

**规范 06**

**级别**：a（强制）  
**名称**：统一缩进为4个空格  
**规范内容**：所有代码缩进必须为 4 个空格，禁止使用 Tab。  
**示例代码**：

public void hello() {

System.out.println("Hello"); // ✅ 正确

}

**规范 07**

**级别**：a（强制）  
**名称**：if / for / while 语句必须使用大括号  
**规范内容**：即使只有一行代码，也必须使用大括号 {} 包裹控制语句体，避免隐式错误。  
**示例代码**：

if (count > 0) {

doSomething();

} // 正确

if (count > 0)

doSomething(); // 错误

**规范 08**

**级别**：a（强制）  
**名称**：避免魔法值，使用常量或枚举  
**规范内容**：程序中不能直接使用数字、字符串等“魔法值”，必须使用有意义的常量或枚举代替。  
**示例代码**：

int max = Constants.MAX\_RETRY\_COUNT; // 正确

if (retryCount > 5) { // 错误

// magic number

}

**规范 09**

**级别**：a（强制）  
**名称**：资源必须关闭，使用 try-with-resources  
**规范内容**：对于实现 AutoCloseable 接口的资源（如流、连接），应使用 try-with-resources 自动关闭，防止内存泄漏。  
**示例代码**：

java

复制编辑

try (BufferedReader br = new BufferedReader(new FileReader("file.txt"))) {

br.readLine();

} // 正确

BufferedReader br = new BufferedReader(new FileReader("file.txt"));

br.readLine(); // 错误：未关闭

代码风格规范

**规范 10**

**级别**：a（强制）  
**名称**：统一使用 4 空格缩进  
**规范内容**：所有缩进必须使用 **4 个空格**，禁止使用 Tab。确保不同编辑器/IDE 下保持一致的代码展示。  
**示例代码**：

public void sayHello() {

System.out.println("Hello"); // 正确，4 个空格

}

**规范 11**

**级别**：a（强制）  
**名称**：大括号必须换行书写  
**规范内容**：{} 必须**换行**，并成对使用。控制语句（如 if、for）和方法体的 { 必须另起一行，避免省略。  
**示例代码**：

if (flag)

{

doSomething(); // 正确

}

if (flag) doSomething(); // 错误：一行内写了 if 和语句体

**规范 12**

**级别**：b（推荐）  
**名称**：单行长度不超过 120 字符  
**规范内容**：每行代码推荐不超过 120 个字符。超过请换行并进行缩进，以提高可读性。  
**示例代码**：

String message = "这是一段内容较长的文本，如果超过规定长度，请在合适的位置进行换行处理"; // 正确（不超过）

**规范 13**

**级别**：b（推荐）  
**名称**：方法之间空一行  
**规范内容**：不同方法之间必须空一行，逻辑块之间空行有助于结构清晰。  
**示例代码**：

public void start() {

// ...

}

public void stop() {

// ... //正确

}

**规范 14**

**级别**：b（推荐）  
**名称**：注释格式统一，使用 // 或 /\* \*/  
**规范内容**：单行注释使用 //，块注释使用 /\* \*/，不允许随意缩写、拼音、无意义注释。推荐用 JavaDoc 注释公共接口。  
**示例代码**：

// 初始化配置

config.load(); // 正确

/\* 保存用户数据 \*/

saveUser(user); // 正确

// haha 嘿嘿 //错误：无意义注释

**规范 16**

**级别**：a（强制）  
**名称**：文件编码统一为 UTF-8，结尾保留空行  
**规范内容**：所有 Java 文件必须使用 UTF-8 编码，并在文件末尾保留一个换行空行，避免系统兼容性问题。  
**示例代码**：

// 文件内容结束处应有一行空白行（不展示但必须存在）

编码实践规范

**规范 17**

**级别**：a（强制）  
**名称**：禁止对 null 调用 .equals()  
**规范内容**：避免空指针异常，不能用 obj.equals("str") 判断，应使用 "str".equals(obj) 反向调用。  
**示例代码**：

if ("admin".equals(userRole)) { // 正确

// do something

}

if (userRole.equals("admin")) { // 错误：userRole 可能为 null

// do something

}

**规范 18**

**级别**：a（强制）  
**名称**：资源必须关闭，使用 try-with-resources  
**规范内容**：对流、连接等必须及时关闭，推荐使用 try-with-resources 自动释放资源。  
**示例代码**：

try (BufferedReader reader = new BufferedReader(new FileReader("file.txt"))) {

String line = reader.readLine();

} //正确

**规范 19**

**级别**：a（强制）  
**名称**：禁止捕获 Exception / Throwable 总类异常  
**规范内容**：不能直接 catch (Exception) 或 catch (Throwable)，应捕获具体异常类。  
**示例代码**：

java

复制编辑

try {

int x = 1 / 0;

} catch (ArithmeticException e) { //正确

// handle

}

catch (Exception e) { //错误

}

**规范 20**

**级别**：b（推荐）  
**名称**：方法长度不超过 50 行  
**规范内容**：方法不宜过长，推荐拆分成小的职责明确的方法，提高可读性与可测试性。  
**示例代码**：

public void processOrder() {

validateOrder();

calculateTotal();

saveOrder(); // 正确，分块处理

}

**规范 21**

**级别**：a（强制）  
**名称**：禁止魔法值，统一使用常量  
**规范内容**：代码中不应出现硬编码数值/字符串，必须定义为常量或枚举。  
**示例代码**：

public static final int TIMEOUT = 3000;

if (connection.wait(TIMEOUT)) { // 正确

}

if (connection.wait(3000)) { // 错误：魔法值

}

**规范 22**

**级别**：b（推荐）  
**名称**：if-else 嵌套不超过三层  
**规范内容**：嵌套结构应尽量平坦，推荐使用早返回、策略模式等避免深层嵌套。  
**示例代码**：

if (!valid()) return;

if (ready()) {

if (hasPermission()) {

start(); //正确（最多三层）

}

}

**规范 23**

**级别**：a（强制）  
**名称**：公共方法必须添加注释  
**规范内容**：所有 public 方法必须包含 JavaDoc，说明功能、参数、返回值等（前文已展示）。  
**示例代码**：

/\*\*

\* 计算两个整数的和

\* @param a 第一个数

\* @param b 第二个数

\* @return 两数之和

\*/

public int add(int a, int b) {

return a + b;

} // 正确

**规范 24**

**级别**：b（推荐）  
**名称**：推荐使用 Optional 避免 null 返回  
**规范内容**：返回可能为 null 的值时，优先考虑使用 Optional<T>，提升可读性和安全性。  
**示例代码**：

public Optional<User> findUserById(Long id) {

return Optional.ofNullable(userMap.get(id)); // 正确

}

**规范 25**

**级别**：b（推荐）  
**名称**：代码中出现中文需注释说明  
**规范内容**：中文文本仅用于日志、UI 展示等场景，必须注明用途，避免误解。  
**示例代码**：

logger.info("库存不足，无法下单"); //正确，用于日志

**规范 26**

**级别**：a（强制）  
**名称**：禁止提交未使用的变量或 import  
**规范内容**：所有未使用的变量、方法、import 应在提交前清除，保持代码整洁。  
**示例代码**：

import java.util.List; //正确：使用了 List

import java.util.Date; //错误：未使用

面向对象设计规范

**规范 27**

**级别**：a（强制）  
**名称**：类的职责应单一（SRP）  
**规范内容**：一个类只负责一项职责，避免同时承担过多功能。违反单一职责将导致代码难以维护与扩展。  
**示例代码**：

//正确：职责分离

class UserValidator {

public boolean isValid(User user) { ... }

}

class UserService {

public void register(User user) { ... }

}

//错误：职责混合

class UserManager {

public boolean isValid(User user) { ... }

public void register(User user) { ... }

}

**规范 28**

**级别**：b（推荐）  
**名称**：使用接口进行面向接口编程  
**规范内容**：对外暴露接口，内部依赖接口而非实现类，方便单元测试与依赖注入（DI）。  
**示例代码**：

//正确

interface OrderRepository {

void save(Order order);

}

class OrderService {

private OrderRepository repository;

public OrderService(OrderRepository repository) {

this.repository = repository;

}

}

**规范 29**

**级别**：b（推荐）  
**名称**：禁止滥用继承，优先使用组合  
**规范内容**：继承需谨慎，优先使用组合（has-a）关系重用功能，避免类层次复杂和继承滥用。  
**示例代码**：

// 正确：组合优于继承

class Engine { ... }

class Car {

private Engine engine;

}

// 错误：不恰当继承

class Engine extends Car { ... }

**规范 30**

**级别**：a（强制）  
**名称**：构造方法中禁止执行复杂逻辑  
**规范内容**：构造方法只用于字段初始化，不应包含复杂逻辑（如网络、IO 操作），防止对象创建过程出错。  
**示例代码**：

java

复制编辑

public class User {

private String name;

public User(String name) {

this.name = name; // 正确：只初始化

}

}

// 错误：构造方法里请求网络

public class ConfigFetcher {

public ConfigFetcher() {

fetchFromServer(); // 不推荐

}

}

**规范 31**

**级别**：b（推荐）  
**名称**：优先使用不可变对象（Immutable）  
**规范内容**：对象一经创建不再改变，提高线程安全性，推荐将类设计为不可变类（如字段 final、无 setter）。  
**示例代码**：

java

复制编辑

public final class User {

private final String name;

public User(String name) {

this.name = name;

}

public String getName() {

return name;

}

}

异常与日志处理规范

**规范 32**

**级别**：a（强制）  
**名称**：禁止使用 catch 后空处理  
**规范内容**：异常 catch 后必须有处理逻辑，如记录日志、抛出新异常，禁止空 catch 块以免隐藏问题。  
**示例代码**：

try {

process();

} catch (IOException e) {

logger.error("文件处理失败", e); // 正确

}

// 错误：吞掉异常

try {

process();

} catch (IOException e) {

}

**规范 33**

**级别**：b（推荐）  
**名称**：统一异常处理机制  
**规范内容**：统一使用自定义异常类，避免抛出原始 Exception，便于排查与封装错误码。  
**示例代码**：

public class BusinessException extends RuntimeException {

public BusinessException(String msg) {

super(msg);

}

}

throw new BusinessException("参数非法"); // 推荐

**规范 34**

**级别**：a（强制）  
**名称**：抛出异常时务必带上原始异常信息  
**规范内容**：包装异常时，必须传入原始异常作为 cause，保留栈信息，方便定位。  
**示例代码**：

try {

process();

} catch (SQLException e) {

throw new BusinessException("查询失败", e); //正确

}

**规范 35**

**级别**：b（推荐）  
**名称**：日志使用占位符，不拼接字符串  
**规范内容**：日志记录中避免字符串拼接，使用日志框架的 {} 占位符功能以提升性能。  
**示例代码**：

logger.info("用户ID为：{}", userId); // 正确

logger.info("用户ID为：" + userId); // 错误

**规范 36**

**级别**：a（强制）  
**名称**：异常栈必须记录日志  
**规范内容**：在捕获并记录异常时，必须完整打印异常堆栈，不能只打印 message，否则排查困难。  
**示例代码**：

logger.error("调用失败", e); // 正确

logger.error("调用失败：" + e.getMessage()); // 错误

单元测试与注释规范

**规范 37**

**级别**：a（强制）  
**名称**：新增业务逻辑必须配套单元测试  
**规范内容**：所有新增的核心业务逻辑，必须编写单元测试，确保边界值、异常、正常流程均被覆盖。  
**示例代码**：

@Test

public void testAddUserSuccess() {

User user = new User("Tom");

userService.add(user);

assertTrue(userService.exists(user));

}

**规范 38**

**级别**：b（推荐）  
**名称**：测试类命名应以 Test 结尾  
**规范内容**：测试类应以被测类名开头，Test 结尾，方法以 testXxx 命名，便于识别与自动化测试运行。  
**示例代码**：

// 被测类：UserService

public class UserServiceTest {

@Test

public void testRegister() {

// ...

}

}

**规范 39**

**级别**：b（推荐）  
**名称**：注释须描述“为什么做”，不只是“做了什么”  
**规范内容**：注释内容应关注 “原因”与“意图”，不应仅仅描述代码行为。注释应具有解释性、背景性。  
**示例代码**：

// 使用缓存提升性能，避免频繁访问数据库（为什么）

// 避免重复发送验证码（业务背景）

if (cache.contains(phone)) {

return;

}

**规范 40**

**级别**：a（强制）  
**名称**：废弃方法必须添加 @Deprecated 注解并写明原因  
**规范内容**：所有废弃方法必须显式加 @Deprecated，并通过注释或 JavaDoc 说明替代方案与原因。  
**示例代码**：

/\*\*

\* @deprecated 请使用 loginV2(String username, String password)

\*/

@Deprecated

public void login(String username) {

// ...

}