生活记录仪项目编程规范

第七组

一、命名规范

1. 变量、函数、类名使用驼峰命名法，首字母根据范围大小写：

- 变量、函数：首字母小写，如 “userName”

- 类名：首字母大写，如 “UserManager”

- 特例: 后端有特殊写法如“getRobot\_id”，其中下划线前为函数功能，下划线后为实体类中变量名。

- 对局部变量和特定函数参数，可采用下划线命名法，如：“user\_id”

2. 常量使用全大写，单词间以下划线分隔，如 `MAX\_VALUE`

3. 用途明确的缩写可以使用，如 `ID`、`URL` 等。

4. 名称应具有描述性，避免使用模糊的字母变量名。若采用缩写，仅采用常用的、公认的缩写方式命名，如“count”应缩写为“cnt”，而非“cou”。

二、格式规范

1. 与编译器自动缩进方式一致，缩进使用制表符（Tab）或数个空格。

2. 每行代码长度不超过80个字符，如需换行，尽量在操作符后换行。

3. 大括号（`{}`）风格：

- 代码很长的：大括号左边空一格，新的一行开始

- 仅包括一行内容的：大括号左边空一格，同一行开始

4. 同一行中的逗号、运算号和分号后应有一个空格。

5. 控制语句、函数声明等关键字后应留一个空格。

三、注释规范

1. 采用单行注释（`//`）和多行注释（`/\* \*/`）。

2. 类、函数、模块等应有描述性的注释。

3. 注释应简洁明了，避免冗余信息。

4. 对于复杂的逻辑和算法，添加注释以解释原理。

四、控制语句规范

1. 使用空格将控制语句关键字与括号隔开。

2. 优先使用 for-of、map、filter 等高阶函数代替传统 for 循环。

3. 对于多个条件判断，优先使用 switch-case 结构。

五、异常处理

1. 处理可能出现异常的代码，考虑到所有可能出错的情况。

2. 若使用try-catch的方式处理异常，对捕获的异常进行适当处理，避免直接忽略。

3. 将特定异常与通用异常分开处理。

六、单元测试

1. 为每个模块编写相应的单元测试，使用xUnit单元测试框架，编写单元测试代码，自动生成测试报告。

2. 测试用例应覆盖正常情况、边界情况和异常情况。

3. 使用断言库（如 Jest）验证测试结果。

4. 定期运行单元测试以确保代码质量。

5. 整体上单元测试应该覆盖90%的代码范围。

遵循以上编程规范，有助于提高代码可读性、可维护性和稳定性。团队成员应共同遵守规范，以确保项目的顺利进行。