**java代码复查指南和检查表**

|  |  |  |  |
| --- | --- | --- | --- |
| 类型 | 检查项 | 合格 | 不合格 |
| 命名 |  |  |  |
|  | 命名规则是否与所采用的规范保持一致？ |  |  |
|  | 类名是否存在重名问题？ |  |  |
| 注释 |  |  |  |
|  | 注释是否较清晰且必要？ |  |  |
|  | 复杂的分支流程是否已经被注释？ |  |  |
|  | 距离较远的}是否已经被注释？ |  |  |
|  | 函数是否已经有文档注释？（功能、输入、返回及其他可选） |  |  |
|  | 特殊用法是否被注释？ |  |  |
| 声明、空白、缩进 |  |  |  |
|  | 每行是否只声明了一个变量？（特别是那些可能出错的类型） |  |  |
|  | 变量是否已经在定义的同时初始化？ |  |  |
|  | 类属性是否都执行了初始化？ |  |  |
|  | 代码段落是否被合适地以空行分隔？ |  |  |
|  | 是否合理地使用了空格使程序更清晰？ |  |  |
|  | 代码行长度是否在要求之内？ |  |  |
|  | 集合是否被定义为泛型类型？ |  |  |
| 语句/功能分布/规模 |  |  |  |
|  | 包含复合语句的{}是否成对出现并符合规范？ |  |  |
|  | 是否给单个的循环、条件语句也加了{}？ |  |  |
|  | 单个变量是否只做单个用途？ |  |  |
|  | 单行是否只有单个功能？（不要使用；进行多行合并） |  |  |
|  | 单个函数是否执行了单个功能并与其命名相符？ |  |  |
|  | 操作符＋＋和— —操作符的应用是否符合规范？ |  |  |
| 规模 |  |  |  |
|  | 单个函数不超过规定行数？ |  |  |
|  | 缩进层数是否不超过规定？ |  |  |
| 可靠性（总则/变量和语句） |  |  |  |
|  | 是否已经消除了所有警告？ |  |  |
|  | 常数变量是否声明为final？ |  |  |
|  | 对象使用前是否进行了检查？ |  |  |
|  | 成员变量，局部变量是否在使用前被赋值？ |  |  |
|  | 局部对象变量使用后是否被复位为NULL？ |  |  |
|  | 对数组的访问是否是安全的？（合法的index取值为[0, MAX\_SIZE-1]）。 |  |  |
|  | 是否确认没有同名变量局部重复定义问题? |  |  |
|  | 程序中是否只使用了简单的表达式？ |  |  |
|  | 是否已经用（）使操作符优先级明确化？ |  |  |
|  | 所有判断是否都使用了（常量==变量 或者 常量.equals(变量)）的形式？ |  |  |
|  | 是否每个if-else if-else语句都有最后一个else以确保处理了全集？ |  |  |
|  | 是否每个switch-case语句都有最后一个default以确保处理了全集？ |  |  |
|  | for循环是否都使用了包含下限不包含上限的形式？（k=0; k<MAX） |  |  |
|  | XML标记书写是否完整，字符串的拼写是否正确? |  |  |
|  | 对于流操作代码的异常捕获是否有finally操作以关闭流对象? |  |  |
|  | 退出代码段时是否对临时对象做了释放处理? |  |  |
|  | 对浮点数值的相等判断是否是恰当的？ |  |  |
|  | 是否对象比较都使用了equals？ |  |  |
|  | 使用equals进行比较时是否确保比较的两个对象类型一致？ |  |  |
|  | 是否在类型转换前确保了类型的兼容？ |  |  |
|  | 包装类做简单预算前是否保证非空？ 建议都使用包装类。 |  |  |
|  | 对象属性在使用前是否确保被准确赋值？ |  |  |
|  | 方法调用前是否有非空判断？ |  |  |
|  | 非线程安全的对象是否被正确保证线程安全？ |  |  |
|  | 相同用意的成员变量是否使用了相同的命名？ |  |  |
| 可靠性（函数） |  |  |  |
|  | 入口对象是否都被进行了判断不为空？ |  |  |
|  | 入口数据的合法范围是否都被进行了判断？ |  |  |
|  | 是否对有异常抛出的方法都执行了try...catch保护？ |  |  |
|  | 是否函数的所有分支都有返回值？ |  |  |
|  | int的返回值是否合理？（负值为失败，非负值成功） |  |  |
|  | 对于反复进行了int返回值判断是否定义了函数来处理？ |  |  |
|  | 关键代码是否做了捕获异常处理? |  |  |
|  | 字典表定义是否用枚举，或者有一个统一的定义？ |  |  |
|  | 是否对方法返回值对象做了null检查，该返回值定义时是否被初始化? |  |  |
|  | 是否对同步对象的遍历访问做了代码同步? |  |  |
|  | 是否确认在对Map对象使用迭代遍历过程中没有做增减元素操作？ |  |  |
|  | 线程处理函数循环内部是否有异常捕获处理，防止线程抛出异常而退出? |  |  |
|  | 原子操作代码异常中断，使用的相关外部变量是否恢复先前状态? |  |  |
|  | 函数对错误的处理是恰当的？ |  |  |
|  | 异常捕获后是否进行了日志记录或异常继续抛出？ |  |  |
|  | 是否构造方法中不调用当前对象的构造方法 |  |  |
| 可维护性 |  |  |  |
|  | 实现代码中是否消除了直接常数？（用于计数起点的简单常数例外） |  |  |
|  | 是否正确使用了日志记录? |  |  |
|  | 是否有冗余判断语句？（如：if (b) return true; else return false;） |  |  |
|  | 是否把方法中的重复代码抽象成私有函数? |  |  |
| 代码警告 |  |  |  |
|  | 是否清除了多余导入的包或类？ |  |  |
|  | 是否清除了只定义未使用的局部变量？ |  |  |
|  | 是否将魔鬼数字修改为常量使用？ |  |  |
| 可读性 |  |  |  |
|  | 是否用if else结构替换了三元运算符？ |  |  |
| 性能 |  |  |  |
|  | 日志记录的Log,Logger对象是否定义为常量？ |  |  |
| 日志 |  |  |  |
|  | 打印信息是否都用日志管理？ |  |  |