

基于B/S架构的在线数独对战平台

代

码

走

查

报

告

课 程： 软件工程

题 目： OurSudoku

成 员： SE-2020-G02

|  |  |  |
| --- | --- | --- |
| 文件状态：  [ ] 草稿  [√] 正式发布  [ ] 正在修改 | 文件标识： | SE2020-G02-软件代码走查报告 |
| 当前版本： | 1.0.0 |
| 作者： | 刘羽佳、张鑫、潘言 |
| 完成日期： | 2021.01.20 |

软件代码走查报告

说明：

《软件代码走查说明》 描述了软件在编程开发过程中根据代码规范以及功能检测对所涉及的代码格式以及性能等一系列检查。该文档可能还要用《软件测试报告》和《软件代码清单说明》以补充。

**版本记录**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 版本 | 作者 | 参与者 | 审核人 | 起始日期 | 备注 |
| 0.0.1 | 潘言 | 潘言 | 刘羽佳 | 2020.12.20 |  |
| 0.1.0 | 潘言 | 潘言 | 刘羽佳 | 2021.01.19 |  |
| 1.0.0 | 潘言 | 刘羽佳 | 刘羽佳 | 2021.01.20 |  |

目录

[软件代码走查报告 2](#_Toc8371)

[1引言 4](#_Toc12823)

[1.1标识 4](#_Toc18402)

[1.2系统概述 4](#_Toc25610)

[1.3文档概述 4](#_Toc1195)

[2引用文件 4](#_Toc23150)

[3代码走查规范报告 4](#_Toc11621)

[附录 14](#_Toc1579)

# 1引言

## 1.1标识

标题：《软件代码走查报告》

版本号：1.0.0

说明：本文档支持Office、WPS等word文档阅读软件。

## 1.2系统概述

本小组开发的数独在线对战平台是一款能够让数独爱好者之间互相PK的平台，除了最基本的房间对战功能，我们还开放了天梯排位模块、用户出题模块、悬赏模块以及自主练习模块。

## 1.3文档概述

本文档的内容包括前端代码和后端代码等并对其进行详细分析。

本文档仅对SEG02小组成员开放，在征得所有成员的同意后才可对外展示。

# 2引用文件

[1] 百度文库《代码走查规范表》

# 3代码走查规范报告

**参考评审表：**

|  |  |  |
| --- | --- | --- |
| 评审对象 | 后端代码 | 评审意见  注释可以多加一点 |
| 评审人 | 潘言 |
| 被评人 | 张鑫 |
| 序号 | 检查项 |
| **走查前准备** | | |
| 1 | 得到一份解释代码的最新的设计文档 | 通过 |
| 2 | 代码都已提交 ，版本统一 | 通过 |
| **程序结构组织** | | |
| 1 | 所有代码的结构清晰，具有良好的结构外观和整齐 | 通过 |
| 2 | 所有的模块（函数和外部接口）定义清晰，模块分解清楚 | 通过 |
| 3 | 所有的功能需求都明显的覆盖 | 通过 |
| 4 | 整个代码体系结构组合合理 | 通过 |
| 5 | 所有主要的数据构造描述清楚，合理 | 通过 |
| 6 | 模块中所有的数据结构都定义为局部的，并且通过定义好的函数进行访问 | 通过 |
| 7 | 所有的接口模块化，修改时不影响其他代码模块 | 通过 |
| 8 | 代码体系构架对空间和速度都已经进行考虑 | 通过 |
| 9 | 提供了处理数据的策略 | 通过 |
| 10 | 数据库操作、IO操作等是否正确关闭资源。并且必须在try -catch-finally 的finally中关闭。 | 通过 |
| 11 | 多处使用相同代码时，应定义唯一方法或变量以供使用。 | 通过 |
| 12 | 导入类时，如果仅使用包中的几个类，应导入具体类，而不是导入整个包。 | 通过 |
| 13 | 数组声明的时候使用 int[] index ，而不要使用 int index[]。 | 通过 |
| 14 | 检查代码和详细设计是否一致。 | 基本通过 |
| 15 | 检查类中是否有无效的代码或者是无用的代码。 | 通过 |
| 16 | 将错误写进日志 | 通过 |
| 17 | 所有的文件名符合文件命名规范，见词知意 | 通过 |
| 18 | 文件和模块分组清晰 | 通过 |
| 19 | 所有的代码行在80字符以内 | 基本通过 |
| 20 | 整个程序的文件小于2000行 | 基本通过 |
| **代码组织** | | |
| 1 | 数据库查询语句不要出现select \* | 基本通过，特定场合需要读取改表的所有信息 |
| 2 | 所有的变量名都小于32字符 | 通过 |
| 3 | 每行代码都小于80字符 | 通过 |
| 4 | 类名、方法名、变量名的命名是否规范。 | 通过 |
| 5 | 有返回值的方法是否正确返回。Return语句应定义在方法结尾处。 | 通过 |
| 6 | 代码排版是否规范 | 通过 |
| 7 | 所有的行每行最多只有一句代码或一个表达式 | 通过 |
| 8 | 复杂的表达式具备可读性 | 通过 |
| 9 | 续行缩进 | 通过 |
| 10 | 括号在合适的位置 | 通过 |
| 11 | 每个顺序的小块用空行隔开 | 通过 |
| 12 | 注释和代码对齐或接续在代码之后 | 通过 |
| 13 | 显示的文本无拼写和语法错误 | 通过 |
| 14 | 代码与操作符的优先级无关 | 通过 |
| 15 | 所有的表达式使用了正确的操作符 | 通过 |
| **函数组织** | | |
| 1 | 所有的函数名都小于64个字符 | 通过 |
| 2 | 代码中有相关注解 | 通过 |
| 3 | 函数的名字清晰的定义了它的目标以及函数所做的事情 | 通过 |
| 4 | 函数中所有的部分都合理的组成函数，相关独立的语句组组成函数 | 通过 |
| 5 | 函数高内聚 只做一件事情，并做好 | 通过 |
| 6 | 函数和其他代码弱耦合 | 通过 |
| 7 | 参数遵循一个明显的顺序 | 通过 |
| 8 | 所有的参数都被使用 | 通过 |
| 9 | 函数的参数接口关系清晰 | 通过 |
| 10 | 如果一个函数有返回值，在所有的出口都有返回值 | 通过 |
| 11 | 函数使用了最少数目的return语句 | 通过 |
| 12 | 函数的参数个数小于7个 | 通过 |
| 13 | 使用的算法有清楚的说明 | 通过 |
| 14 | 函数检查了输入数据的合法性 | 通过 |
| 15 | 函数异常处理清楚 | 通过 |
| 16 | 函数设计已经考虑了将来的变化 | 基本通过 |
| 17 | 调试信息存在于代码中并容易激活 | 通过 |
| 18 | 代码检查调用函数的返回值，参数和调用匹配 | 通过 |
| 19 | 函数确保了没有影响函数外代码 | 通过 |
| 20 | 对接口的调用必须要有判断传入的参数不能为空和传入的参数正确。 | 通过 |
| **数据类型与变量** | | |
| 1 | 数据类型存在数据类型解释 | 通过 |
| 2 | 代码避免了重新定义预先定义的数据类型 | 通过 |
| 3 | 静态变量明确区分 | 通过 |
| 4 | 命名和现实生活中的事务接近而不仅仅是一个程序类型 | 通过 |
| 5 | 所有的变量都有最小的活动范围 | 通过 |
| 6 | 所有的全局变量都描述清楚 | 通过 |
| 7 | 所有的变量都用到了 | 通过 |
| **条件判断** | | |
| 1 | 条件检查和结果在代码中清晰 | 通过 |
| 2 | If/else 使用正确 | 通过 |
| 3 | 判断的次数降到最小 | 通过 |
| 4 | boolen表达式表示清楚 | 通过 |
| 5 | 最常用的情况最先判断 ，所有的情况都考虑 | 通过 |
| 6 | 判断体足够短，以使得一次可以看清楚 ，不要有臃肿的逻辑判断。 | 通过 |
| 7 | 嵌套层次小于3次 | 通过 |
| 8 | 判断条件使用引用时，应先判断其是否为null。 | 通过 |
| **循环** | | |
| 1 | 不要出现三层以上的for循环。如有必要必须写清楚注释 | 通过 |
| 2 | 循环体不为空，循环体能够一次看清楚 | 通过 |
| 3 | 循环之前做好初始化代码 | 通过 |
| 4 | 当有明确的多次循环操作，使用For循环 | 通过 |
| 5 | 当有不明确的多次循环操作，while循环被使用 | 通过 |
| 6 | 代码中不存在无穷次循环 | 通过 |
| 7 | 循环的头部进行循环控制 | 通过 |
| 8 | 循环体内的循环变量起到指示作用。 | 通过 |
| 9 | 所有的循环边界是否正确； | 通过 |
| 10 | 循环终止的条件清晰 | 通过 |
| 11 | 循环条件、判断条件的边界值，数组、集合下标是否越界。 | 通过 |
| **总括** | | |
| 1 | 代码直观 | 通过 |
| 2 | 无用的代码、注解已经删除 | 通过 |
| 3 | 除非必要，不要再循环体内定义对象。 | 通过 |
| 4 | 代码是否与详细设计一致。 | 基本通过 |
| 5 | 代码与操作系统无关，不需要任何假设条件 | 通过 |
| **特殊** | | |
| 1 | 所有的数组访问在它们的边界内 | 通过 |
| 2 | 所有常量定义和使用替代代码中的数字 | 通过 |
| 3 | 类型转换明确指明 | 通过 |

|  |  |  |
| --- | --- | --- |
| 评审对象 | 前端代码 | 评审意见  有部分内容简单的标签可以放到同一行中 |
| 评审人 | 张鑫 |
| 被评人 | 刘羽佳 |
| 序号 | 检查项 |
| **走查前准备** | | |
| 1 | 得到一份解释代码的最新的设计文档 | 通过 |
| 2 | 代码都已提交 ，版本统一 | 通过 |
| **程序结构组织** | | |
| 1 | 所有代码的结构清晰，具有良好的结构外观和整齐 | 通过 |
| 2 | 所有的模块（函数和外部接口）定义清晰，模块分解清楚 | 通过 |
| 3 | 所有的功能需求都明显的覆盖 | 通过 |
| 4 | 整个代码体系结构组合合理 | 通过 |
| 5 | 所有主要的数据构造描述清楚，合理 | 通过 |
| 6 | 模块中所有的数据结构都定义为局部的，并且通过定义好的函数进行访问 | 通过 |
| 7 | 所有的接口模块化，修改时不影响其他代码模块 | 通过 |
| 8 | 代码体系构架对空间和速度都已经进行考虑 | 通过 |
| 9 | 提供了处理数据的策略 | 通过 |
| 10 | 数据库操作、IO操作等是否正确关闭资源。并且必须在try -catch-finally 的finally中关闭。 | 通过 |
| 11 | 多处使用相同代码时，应定义唯一方法或变量以供使用。 | 通过 |
| 12 | 导入类时，如果仅使用包中的几个类，应导入具体类，而不是导入整个包。 | 通过 |
| 13 | 数组声明的时候使用 int[] index ，而不要使用 int index[]。 | 通过 |
| 14 | 检查代码和详细设计是否一致。 | 基本通过 |
| 15 | 检查类中是否有无效的代码或者是无用的代码。 | 通过 |
| 16 | 将错误写进日志 | 通过 |
| 17 | 所有的文件名符合文件命名规范，见词知意 | 通过 |
| 18 | 文件和模块分组清晰 | 通过 |
| 19 | 所有的代码行在80字符以内 | 不通过，前端因组件较多，无法在80行之内完成 |
| 20 | 整个程序的文件小于2000行 | 未通过，前端组件较多 |
| **代码组织** | | |
| 1 | 数据库查询语句不要出现select \* | 通过 |
| 2 | 所有的变量名都小于32字符 | 通过 |
| 3 | 每行代码都小于80字符 | 通过 |
| 4 | 类名、方法名、变量名的命名是否规范。 | 通过 |
| 5 | 有返回值的方法是否正确返回。Return语句应定义在方法结尾处。 | 通过 |
| 6 | 代码排版是否规范 | 通过 |
| 7 | 所有的行每行最多只有一句代码或一个表达式 | 通过 |
| 8 | 复杂的表达式具备可读性 | 通过 |
| 9 | 续行缩进 | 通过 |
| 10 | 括号在合适的位置 | 通过 |
| 11 | 每个顺序的小块用空行隔开 | 通过 |
| 12 | 注释和代码对齐或接续在代码之后 | 通过 |
| 13 | 显示的文本无拼写和语法错误 | 通过 |
| 14 | 代码与操作符的优先级无关 | 通过 |
| 15 | 所有的表达式使用了正确的操作符 | 通过 |
| **函数组织** | | |
| 1 | 所有的函数名都小于64个字符 | 通过 |
| 2 | 代码中有相关注解 | 通过 |
| 3 | 函数的名字清晰的定义了它的目标以及函数所做的事情 | 通过 |
| 4 | 函数中所有的部分都合理的组成函数，相关独立的语句组组成函数 | 通过 |
| 5 | 函数高内聚 只做一件事情，并做好 | 通过 |
| 6 | 函数和其他代码弱耦合 | 通过 |
| 7 | 参数遵循一个明显的顺序 | 通过 |
| 8 | 所有的参数都被使用 | 通过 |
| 9 | 函数的参数接口关系清晰 | 通过 |
| 10 | 如果一个函数有返回值，在所有的出口都有返回值 | 通过 |
| 11 | 函数使用了最少数目的return语句 | 通过 |
| 12 | 函数的参数个数小于7个 | 通过 |
| 13 | 使用的算法有清楚的说明 | 通过 |
| 14 | 函数检查了输入数据的合法性 | 通过 |
| 15 | 函数异常处理清楚 | 通过 |
| 16 | 函数设计已经考虑了将来的变化 | 通过 |
| 17 | 调试信息存在于代码中并容易激活 | 通过 |
| 18 | 代码检查调用函数的返回值，参数和调用匹配 | 通过 |
| 19 | 函数确保了没有影响函数外代码 | 通过 |
| 20 | 对接口的调用必须要有判断传入的参数不能为空和传入的参数正确。 | 通过 |
| **数据类型与变量** | | |
| 1 | 数据类型存在数据类型解释 | 通过 |
| 2 | 代码避免了重新定义预先定义的数据类型 | 通过 |
| 3 | 静态变量明确区分 | 通过 |
| 4 | 命名和现实生活中的事务接近而不仅仅是一个程序类型 | 通过 |
| 5 | 所有的变量都有最小的活动范围 | 通过 |
| 6 | 所有的全局变量都描述清楚 | 通过 |
| 7 | 所有的变量都用到了 | 通过 |
| **条件判断** | | |
| 1 | 条件检查和结果在代码中清晰 | 通过 |
| 2 | If/else 使用正确 | 通过 |
| 3 | 判断的次数降到最小 | 通过 |
| 4 | boolen表达式表示清楚 | 通过 |
| 5 | 最常用的情况最先判断 ，所有的情况都考虑 | 通过 |
| 6 | 判断体足够短，以使得一次可以看清楚 ，不要有臃肿的逻辑判断。 | 通过 |
| 7 | 嵌套层次小于3次 | 通过 |
| 8 | 判断条件使用引用时，应先判断其是否为null。 | 通过 |
| **循环** | | |
| 1 | 不要出现三层以上的for循环。如有必要必须写清楚注释 | 通过 |
| 2 | 循环体不为空，循环体能够一次看清楚 | 通过 |
| 3 | 循环之前做好初始化代码 | 通过 |
| 4 | 当有明确的多次循环操作，使用For循环 | 通过 |
| 5 | 当有不明确的多次循环操作，while循环被使用 | 通过 |
| 6 | 代码中不存在无穷次循环 | 通过 |
| 7 | 循环的头部进行循环控制 | 通过 |
| 8 | 循环体内的循环变量起到指示作用。 | 通过 |
| 9 | 所有的循环边界是否正确； | 通过 |
| 10 | 循环终止的条件清晰 | 通过 |
| 11 | 循环条件、判断条件的边界值，数组、集合下标是否越界。 | 通过 |
| **总括** | | |
| 1 | 代码直观 | 通过 |
| 2 | 无用的代码、注解已经删除 | 通过 |
| 3 | 除非必要，不要再循环体内定义对象。 | 通过 |
| 4 | 代码是否与详细设计一致。 | 通过 |
| 5 | 代码与操作系统无关，不需要任何假设条件 | 通过 |
| **特殊** | | |
| 1 | 所有的数组访问在它们的边界内 | 通过 |
| 2 | 所有常量定义和使用替代代码中的数字 | 通过 |
| 3 | 类型转换明确指明 | 通过 |

|  |  |  |
| --- | --- | --- |
| 评审对象 | 前端代码 | 评审意见  可以把部分style放到<style>中，用class的形式调用 |
| 评审人 | 刘羽佳 |
| 被评人 | 潘言 |
| 序号 | 检查项 |
| **走查前准备** | | |
| 1 | 得到一份解释代码的最新的设计文档 | 通过 |
| 2 | 代码都已提交 ，版本统一 | 通过 |
| **程序结构组织** | | |
| 1 | 所有代码的结构清晰，具有良好的结构外观和整齐 | 通过 |
| 2 | 所有的模块（函数和外部接口）定义清晰，模块分解清楚 | 通过 |
| 3 | 所有的功能需求都明显的覆盖 | 通过 |
| 4 | 整个代码体系结构组合合理 | 通过 |
| 5 | 所有主要的数据构造描述清楚，合理 | 通过 |
| 6 | 模块中所有的数据结构都定义为局部的，并且通过定义好的函数进行访问 | 通过 |
| 7 | 所有的接口模块化，修改时不影响其他代码模块 | 通过 |
| 8 | 代码体系构架对空间和速度都已经进行考虑 | 通过 |
| 9 | 提供了处理数据的策略 | 通过 |
| 10 | 数据库操作、IO操作等是否正确关闭资源。并且必须在try -catch-finally 的finally中关闭。 | 通过 |
| 11 | 多处使用相同代码时，应定义唯一方法或变量以供使用。 | 通过 |
| 12 | 导入类时，如果仅使用包中的几个类，应导入具体类，而不是导入整个包。 | 通过 |
| 13 | 数组声明的时候使用 int[] index ，而不要使用 int index[]。 | 通过 |
| 14 | 检查代码和详细设计是否一致。 | 基本通过 |
| 15 | 检查类中是否有无效的代码或者是无用的代码。 | 通过 |
| 16 | 将错误写进日志 | 通过 |
| 17 | 所有的文件名符合文件命名规范，见词知意 | 通过 |
| 18 | 文件和模块分组清晰 | 通过 |
| 19 | 所有的代码行在80字符以内 | 基本通过 |
| 20 | 整个程序的文件小于2000行 | 未通过，前端组件较多 |
| **代码组织** | | |
| 1 | 数据库查询语句不要出现select \* | 通过 |
| 2 | 所有的变量名都小于32字符 | 通过 |
| 3 | 每行代码都小于80字符 | 通过 |
| 4 | 类名、方法名、变量名的命名是否规范。 | 通过 |
| 5 | 有返回值的方法是否正确返回。Return语句应定义在方法结尾处。 | 通过 |
| 6 | 代码排版是否规范 | 通过 |
| 7 | 所有的行每行最多只有一句代码或一个表达式 | 通过 |
| 8 | 复杂的表达式具备可读性 | 通过 |
| 9 | 续行缩进 | 通过 |
| 10 | 括号在合适的位置 | 通过 |
| 11 | 每个顺序的小块用空行隔开 | 通过 |
| 12 | 注释和代码对齐或接续在代码之后 | 通过 |
| 13 | 显示的文本无拼写和语法错误 | 通过 |
| 14 | 代码与操作符的优先级无关 | 通过 |
| 15 | 所有的表达式使用了正确的操作符 | 通过 |
| **函数组织** | | |
| 1 | 所有的函数名都小于64个字符 | 通过 |
| 2 | 代码中有相关注解 | 通过 |
| 3 | 函数的名字清晰的定义了它的目标以及函数所做的事情 | 通过 |
| 4 | 函数中所有的部分都合理的组成函数，相关独立的语句组组成函数 | 通过 |
| 5 | 函数高内聚 只做一件事情，并做好 | 通过 |
| 6 | 函数和其他代码弱耦合 | 通过 |
| 7 | 参数遵循一个明显的顺序 | 通过 |
| 8 | 所有的参数都被使用 | 通过 |
| 9 | 函数的参数接口关系清晰 | 通过 |
| 10 | 如果一个函数有返回值，在所有的出口都有返回值 | 通过 |
| 11 | 函数使用了最少数目的return语句 | 通过 |
| 12 | 函数的参数个数小于7个 | 通过 |
| 13 | 使用的算法有清楚的说明 | 通过 |
| 14 | 函数检查了输入数据的合法性 | 通过 |
| 15 | 函数异常处理清楚 | 通过 |
| 16 | 函数设计已经考虑了将来的变化 | 通过 |
| 17 | 调试信息存在于代码中并容易激活 | 通过 |
| 18 | 代码检查调用函数的返回值，参数和调用匹配 | 通过 |
| 19 | 函数确保了没有影响函数外代码 | 通过 |
| 20 | 对接口的调用必须要有判断传入的参数不能为空和传入的参数正确。 | 通过 |
| **数据类型与变量** | | |
| 1 | 数据类型存在数据类型解释 | 通过 |
| 2 | 代码避免了重新定义预先定义的数据类型 | 通过 |
| 3 | 静态变量明确区分 | 通过 |
| 4 | 命名和现实生活中的事务接近而不仅仅是一个程序类型 | 通过 |
| 5 | 所有的变量都有最小的活动范围 | 通过 |
| 6 | 所有的全局变量都描述清楚 | 通过 |
| 7 | 所有的变量都用到了 | 通过 |
| **条件判断** | | |
| 1 | 条件检查和结果在代码中清晰 | 通过 |
| 2 | If/else 使用正确 | 通过 |
| 3 | 判断的次数降到最小 | 通过 |
| 4 | boolen表达式表示清楚 | 通过 |
| 5 | 最常用的情况最先判断 ，所有的情况都考虑 | 通过 |
| 6 | 判断体足够短，以使得一次可以看清楚 ，不要有臃肿的逻辑判断。 | 通过 |
| 7 | 嵌套层次小于3次 | 通过 |
| 8 | 判断条件使用引用时，应先判断其是否为null。 | 通过 |
| **循环** | | |
| 1 | 不要出现三层以上的for循环。如有必要必须写清楚注释 | 通过 |
| 2 | 循环体不为空，循环体能够一次看清楚 | 通过 |
| 3 | 循环之前做好初始化代码 | 通过 |
| 4 | 当有明确的多次循环操作，使用For循环 | 通过 |
| 5 | 当有不明确的多次循环操作，while循环被使用 | 通过 |
| 6 | 代码中不存在无穷次循环 | 通过 |
| 7 | 循环的头部进行循环控制 | 通过 |
| 8 | 循环体内的循环变量起到指示作用。 | 通过 |
| 9 | 所有的循环边界是否正确； | 通过 |
| 10 | 循环终止的条件清晰 | 通过 |
| 11 | 循环条件、判断条件的边界值，数组、集合下标是否越界。 | 通过 |
| **总括** | | |
| 1 | 代码直观 | 通过 |
| 2 | 无用的代码、注解已经删除 | 通过 |
| 3 | 除非必要，不要再循环体内定义对象。 | 通过 |
| 4 | 代码是否与详细设计一致。 | 基本通过 |
| 5 | 代码与操作系统无关，不需要任何假设条件 | 通过 |
| **特殊** | | |
| 1 | 所有的数组访问在它们的边界内 | 通过 |
| 2 | 所有常量定义和使用替代代码中的数字 | 通过 |
| 3 | 类型转换明确指明 | 通过 |

# 附录