逆向分析

黄哲 24214422

设计思想

该应用采用了模块化的设计思想，旨在通过清晰的功能划分和独立的模块，使得每个组件能够专注于自身职责，从而提升系统的可维护性和扩展性。模块化设计不仅确保了各功能模块的较好的高内聚和低耦合性，还为未来新增功能或修改现有功能提供了便利。同时，应用交互简单，注重易用性和用户体验。

设计原则

1. 单一职责原则：每个模块只负责一种功能，如 difficulty 模块只负责设置难度，cursor 模块只负责管理光标选择位置等。
2. 开放封闭原则：软件实体应该对扩展开放、对修改封闭。例如完善重做、撤消功能时应该添加新的模块而不是对原有逻辑脚本进行更改。
3. 依赖倒转原则：高层模块不应该依赖于低层模块，二者都应该依赖于抽象。如项目中的 UI 表现多通过状态管理与业务逻辑交互，不直接依赖于具体的实现细节。

设计模式

1. 观察者模式：一个对象的状态发生改变时，所有依赖于它的对象都得到通知并被自动更新。在该应用中，体现在通过 Svelte Store 表现层能够订阅 store 变化。
2. 单例模式：部分类只有一个实例，以确保全局状态唯一。例如全局只有一个实例管理光标 cursor，保证全局状态统一。

愿景

该数独应用旨在为用户提供多样化且富有挑战性的解题体验。用户可以选择随机生成的题目或自定义输入，体验不同难度的挑战。通过答案提示、撤回与重做、笔记功能和计时功能，用户能够在解题过程中获得支持，优化思路，并提升解题效率。

用例分析

用例	参与者	流程
开始游戏	用户	用户选择难度，生成随机棋盘；或用户输入序列，生成自定义棋盘。开始游戏同时计时。
暂停游戏	用户	点击暂停按钮，冻结棋盘与计时器，点击恢复按钮后复原
选择单元格	用户	用户选择单元格，对选中的单元格做高亮提示，同时提示所在行列与 box；如果选中的单元格中有数字则会同时提示相同数字所在位置。
填写答案	用户	用户选定空白格填写数字，同时触发验证逻辑（包括即时错误提示、游戏结束提示等）。
笔记记录	用户	用户选定空白格，点击笔记按键进入笔记模式，可以在空白格标注候选值并显示。再次点击笔记按键退出笔记模式。
获得提示	用户	用户选定需要提示的空白棋盘格，点击提示按钮，系统返回提示，将提示结果显示在选定的格子上

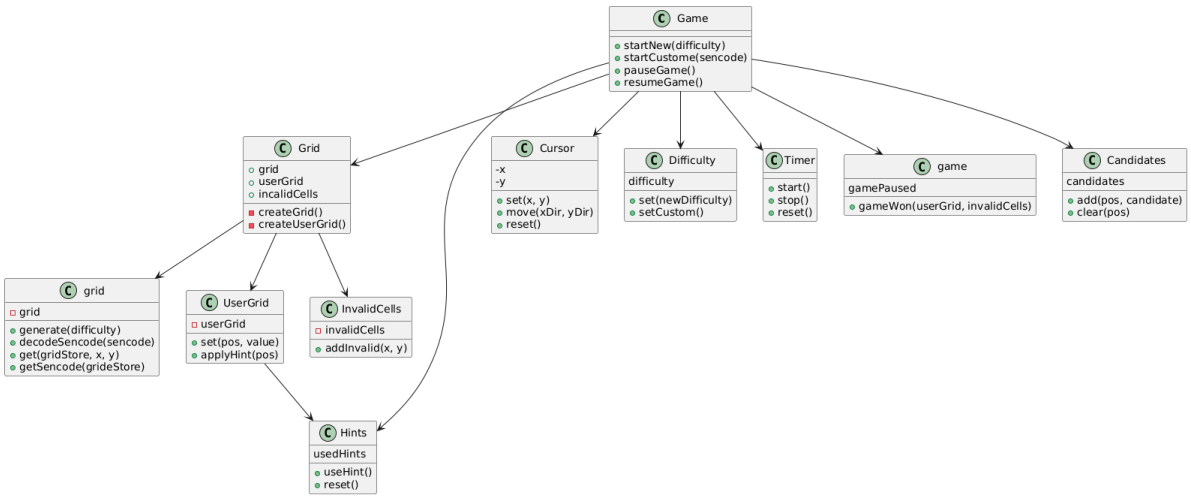
领域模型

1. Game：记录整局游戏状态，包括光标位置、难度、棋盘、计时等属性，负责开始、暂停游戏，管理游戏状态。
2. Grid：记录游戏棋盘，负责生成题目棋盘，生成用户棋盘，获取提示等操作。
3. Candidates：记录候选值，负责管理候选值，包括添加与删除。
4. Cursor：记录光标位置。
5. Difficulty：记录游戏难度。
6. Hints：记录和管理提示次数。
7. Keyboard：记录和管理键盘状态。
8. Modal：记录窗口状态，负责显示或隐藏窗口。
9. Note：记录和管理笔记模式的状态。
10. Setting：记录和管理用户设置。
11. Timer：记录计时器，负责计时器开始、暂停和重置。

技术架构

- 技术栈：
  前端框架：Svelte
  状态管理：Svelte Store
  逻辑脚本：JavaScript
- 层次架构：
  表现层：使用 Svelte 渲染前端界面和实现交互组件，处理用户交互。
  逻辑层：连接表现层和数据层，使用 JavaScript 实现数独生成、验证、提示等核心功能。
  数据层：存放棋盘内容、计时状态等数据，用 Svelte Store 管理全局状态。

对象模型



项目评价

优点：

1. 采用模块化的设计，，每个模块职责明确，如难度设置、光标管理、棋盘生成等。增加了代码的可维护性，也便于未来扩展功能。

2. 采用了分层架构，通过状态订阅实现 UI 层与数据层的交互与自动更新，使 UI 层不用关系方法实现细节。
3. 代码实现逻辑直观，易于理解。

缺点：

1. 状态管理分布在各个脚本中，带来复杂的依赖关系的同时不利于统一管理。
2. 当前的设计中并没有显著使用继承和多态等，各模块多是独立的类，缺乏通过继承扩展功能的可能性。
3. 部分类之间耦合度较高，例如 Grid 和 UserGrid 之间耦合性高，依赖性较强。

改进建议：

1. 将状态统一放入一个全局模块中，避免状态分散，便于统一管理和同步。
2. 引入抽象基类和继承来减少重复代码，同时增加功能扩展的灵活性。
3. 引入中介者模式，协调耦合度较高的模块之间的交互，减少直接依赖。