大作业完整设计文档

刘星翰 2023214259

设计思路：

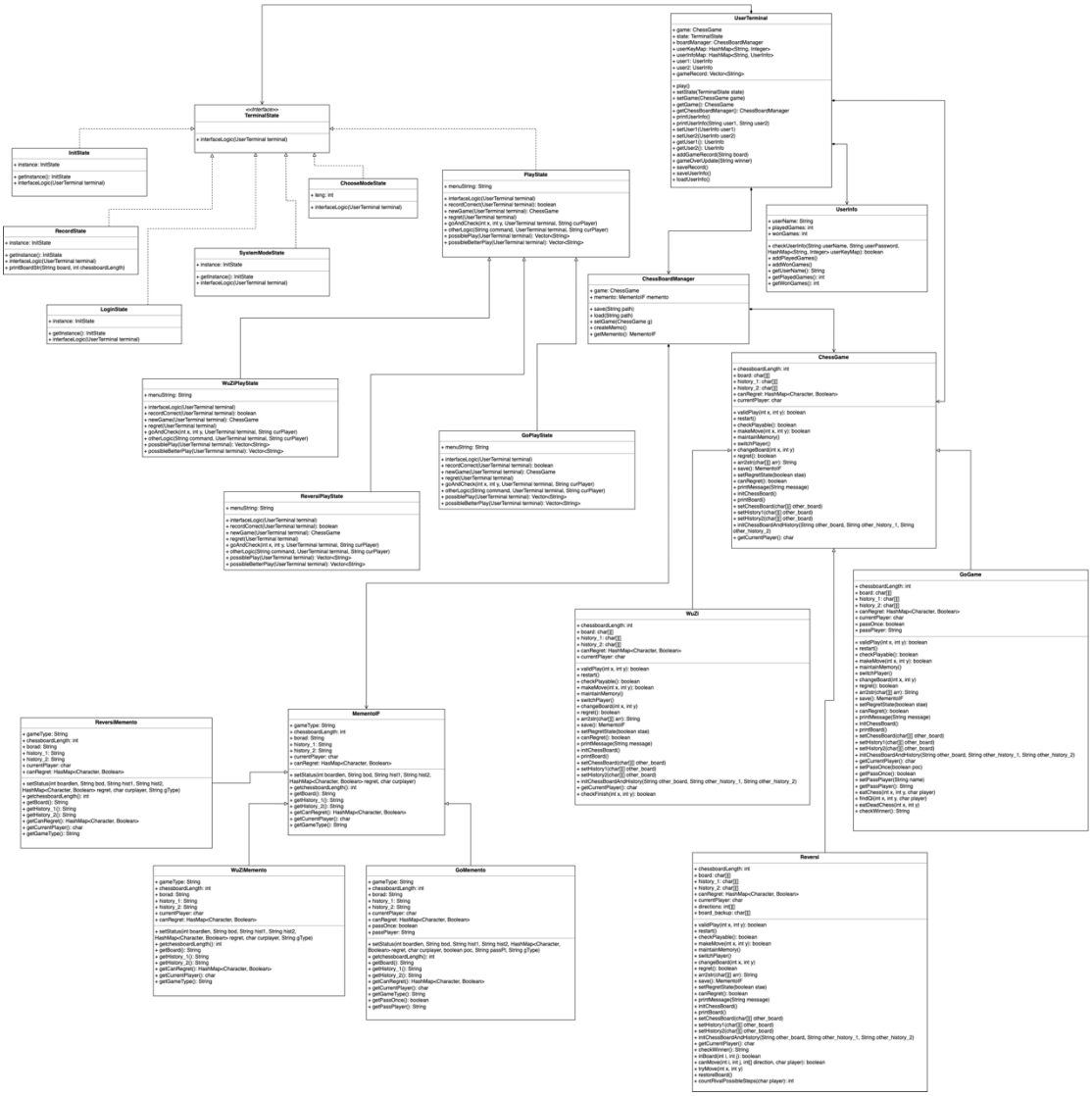
在第一阶段中，我用到的设计模式、设计思路，以及一些对游戏规则的实现介绍如下：（该部分已写在第一阶段文档中）

1. 客户端类为 UserTerminal，负责前端与用户的交互逻辑，考虑到交互逻辑中的状态转移比较复杂，不适合直接用条件判断语句处理，故我采用了状态模式（State Pattern）来设计这一部分。具体来说，以 TerminalState 接口为代表，通过继承和实现方式衍生出多个子类，用于处理不同的逻辑：包括初始化棋盘（选择尺寸、游戏类型）、进行对局中的各项操作（重新开始、落子、悔棋、提子、认输、胜负判断、虚着）等。
2. 由于 InitState 类的对象是全局唯一的，且不会产生变化，故我将其实现为单例（Singleton Pattern），避免重复创建和释放。
3. 考虑到存档、读档的需求，故设计并实现了备忘录模式（Memento Pattern），包括 MementoIF 抽象类以及衍生出的具体备忘录类，另外还实现了 ChessBoardManager 类作为备忘录模式中的 Caretaker 角色，负责以窄接口的形式管理数据的保存和读取。
4. 具体实现细节方面：五子棋白棋先手，围棋黑棋先手。对于悔棋，我规定了每个玩家每局只有一次机会。对于五子棋，没有特别实现禁手的规定。对于围棋，当两边各虚着一次后，认为双方同意进入判断胜负阶段。目前判断胜负的手段是比较双方在棋盘上的棋子数量，相同时判后手的白棋获胜。
5. 围棋的可下点判断算法方面，我实现了通过宽度优先遍历与落子点 4-连通的同色棋子块，并考察其是否有“气”，如果遍历结束时仍没有找到“气”，则判断为不可落子。对于提子的逻辑，则是在判断某棋子块没有“气”之后，再走一遍遍历顺序，并依次提去对应位置的棋子。

在第二阶段中，我保留了所有之前设计的类型和结构，并尽可能不做修改，以遵循“开闭原则”。具体包含以下新内容：

1. 增加 ReversiMemento 类，同样为 Memento 的子类，按备忘录模式负责读写黑白棋相关的游戏记录，这部分没有太多特别的设计。
2. 增加 Reversi 类，继承已有 ChessGame 类，作为黑白棋游戏的功能类，实现相关的操作和规则判定。黑白棋的棋盘大小固定为 8*8，一方有棋可下时不允许跳过，无棋可下时强制跳过，双方均无棋可下时游戏结束进入结算，最终棋子更多的那方胜利。
3. 在状态模式中，新增了单例类主菜单状态 SystemModeState：将录像回放功能整合到主菜单里，其他状态都算作游戏内容的一部分，在这一级不做区分；单例类录像状态 RecordState：整合了所有和录像回放相关的交互逻辑，包括支持上一步/下一步/退出回放；单例类用户身份状态 LoginState：整合了玩家身份选择（包括 AI/游客/用户）、用户注册/登陆相关的逻辑。新增黑白棋交互状态 ReversiPlayState，并且在 ChooseModeState 中，添加了黑白棋相关的逻辑，不改动其余部分；在 PlayState 中，增加了两级 AI 的逻辑。
4. 客户端类 UserTerminal 中，增加了支持维护用户信息的逻辑，以及维护全局录像、保存回放的逻辑，其余部分没有改动。
5. 关于 AI 的设计，我将一级 AI 设计为在可落子点中随机选取。二级 AI 只支持黑白棋，基于黑白棋中的一些基本棋理，例如类似围棋的“金角银边草肚皮”，以及尽可能降低对手的可落子点等，整合出的基于规则的 AI 逻辑如下：可以下角直接下角；不能下角时，优先考虑使对方可落子点最少的方案，如果存在多个，则优先选择距离中心较远的点落子，但星位（X 位）和 C 位是例外，这几个点优先级最低。经过上述筛选过后，如果仍存在多个可落子点，则随机选择。
6. 关于用户信息管理，采用 UserInfo 记录用户名和对局数、胜场数，用 userInfoMap 存储所有用户的信息，用户身份验证采用 userKeyMap 结构，value 是输入密码后将用户名和密码拼接之后的哈希值，以此实现不严格且不必明文存储密码信息，大部分情况下可靠的身份认证。两个 map 都保存到本地文件。

UML 图：（大作业 2.png）



代码仓库：

https://git.tsinghua.edu.cn/xh-liu23/oop_design_final

演示视频：

https://cloud.tsinghua.edu.cn/d/97153ff107e2449d9977/