游戏开始:

writeLog("Game starts!");

writeLog("Players: " + Joiner.on(", ").join(playerBinaryPathsMap.keySet()));

writeLog("Starting table");

writeLog(JsonFormat.printer()

.print(

Table.newBuilder()

.addAllCards(cardsOnTable)

.addAllGems(gems.values())

.addAllNobles(nobles)

.build()));

每一轮开始:

writeLog("Round " + round);

writeLog("Highest score: " + highestScore);

每一次给对应玩家发request:

writeLog("Player " + playerName + "'s turn");

调用binary文件获取response:

二进制文件执行错误: writeLog("Player binary execution is interrupted.");

运行超时 writeLog("Player binary execution timeout.");

运行时错误 writeLog("Player binary execution error.");

Json格式错误 writeLog("Player action response JSON format is invalid.");

出现以上任何错误最后附带

writeLog("Player missed this turn.");

取不同颜色宝石操作:

writeLog(playerName + " is getting different color gems.");

宝石颜色不合法: writeLog("Invalid action. All colors are invalid.");

多于三个颜色:writeLog("Invalid Action. More than 3 colors.");

场上宝石数量不够:writeLog("Invalid Action. More than 3 colors.");

手中宝石操作后多于10:writeLog("Invalid action. Can't hold more than 10 gems.");

成功获取宝石: writeLog("Successfully got " + Joiner.on(", ").join(colors) + " gems.");

取同色宝石操作:

writeLog(playerName + " is getting two same color gems.");

颜色不合法:writeLog("Invalid action. Given color is invalid.");

宝石个数小于4:writeLog("Invalid Action. Can't get gems from the color has less than 4 gems.");

手中宝石操作后多于10:writeLog("Invalid action. Can't hold more than 10 gems.");

成功获得宝石:writeLog("Successfully got two " + color + " gems.");

保留卡牌操作:

writeLog(playerName + " is reserving the card.");

保留卡已达三张:writeLog("Invalid Action. Can't reserve more than 3 cards.");

保留场上卡牌:

writeLog(converToJson(card));

卡牌不合法:writeLog("Invalid action. Can't find the card from table.");

保留某个等级:

writeLog(playerName + " is reserving a card from level " + level);

等级不合法: writeLog("Invalid action. Level " + level + " is invalid.");

该等级卡牌卡池为空:writeLog("Invalid action. No more card for level " + level + ".");

保留成功:writeLog("Successfully reserved the card.");

购买桌上卡牌操作:

writeLog(playerName + " is purchasing the card from table.");

writeLog(converToJson(card));

卡牌不合法:writeLog("Invalid action. Can't find the card from table.");

宝石数量不够:writeLog("Invalid Action. Doesn't have enough gems to purchase.");

购买成功: writeLog("Successfully purchased the card.");

购买保留卡操作:

writeLog(playerName + " is purchasing the reserved card.");

writeLog(converToJson(card));

保留卡不合法:writeLog("Invalid action. Can't find the reserved card.");

宝石数量不够:writeLog("Invalid Action. Doesn't have enough gems to purchase.");

购买成功:writeLog("Successfully purchased the reserved card.");

response中含有贵族操作:

writeLog(playerName + " is being visited by the noble.");

writeLog(converToJson(noble));

贵族不在桌面上存在: writeLog("Invalid action. Can't find the noble from table.");

条件不满足: writeLog("Invalid Action. Doesn't meet the requirement to be visited

条件满足:writeLog("Successfully be visited by the noble.");

从牌库中补充新卡到桌面上:

writeLog("A new Card on the table.");

writeLog(converToJson(card));

游戏结束:

writeLog("Final status of players.");

finalPlayers.forEach(player -> writeLog(converToJson(player)));

writeLog("Final results.");

按照名次输出:

writeLog(“PlayerName got %d score and purchased %d cards")