

FOOP Final Report

組員及分工

| 系級 | 學號 | 姓名 | 分工 |
|-----|-----|-----------|----------|
| 資工三 | 謝心默 | B06505017 | Frontend |
| 資工四 | 許博翔 | B06902069 | Frontend |
| 資工四 | 魏任擇 | B06705059 | Backend |
| 資工四 | 黃奕鈞 | B06902033 | Backend |

遊戲簡介

風聲是一款卡牌型桌遊，玩家人數3~9人。每個玩家會有角色、陣營以及手牌，角色有名稱、技能以及秘密任務；陣營分為軍情局、潛伏戰線、打醬油的；手牌可用作功能牌，也可作為情報。遊戲中依序每回合一位玩家傳遞情報，直到有人達成勝利條件。

角色

每位玩家都會有一位角色，角色有名稱、性別、技能及機密任務（由於技能過度複雜，需要為每個角色特別設計而使軟體設計難度大幅提高，因此暫時還未實作）

- 性別：分為男、女、可男可女，有些角色的技能或秘密任務與性別有關
- 秘密任務：當陣營為打醬油的時，可藉由完成秘密任務而勝利。
- 潛伏：部分角色會在遊戲開始時處於潛伏狀態，其他玩家無法得知其角色資訊，直到發動技能。

陣營

- 軍情局：當屬於此陣營的任何一位玩家獲得3張藍色情報，則全員勝利。
- 潛伏戰線：當屬於此陣營的任何一位玩家獲得3張紅色情報，則全員勝利。
- 打醬油的：當完成秘密任務，則個人勝利。

手牌



作為功能牌：僅需考慮卡片上的標題以及敘述。共有九種功能牌，每種皆有不同效果，可使用時機也不同。

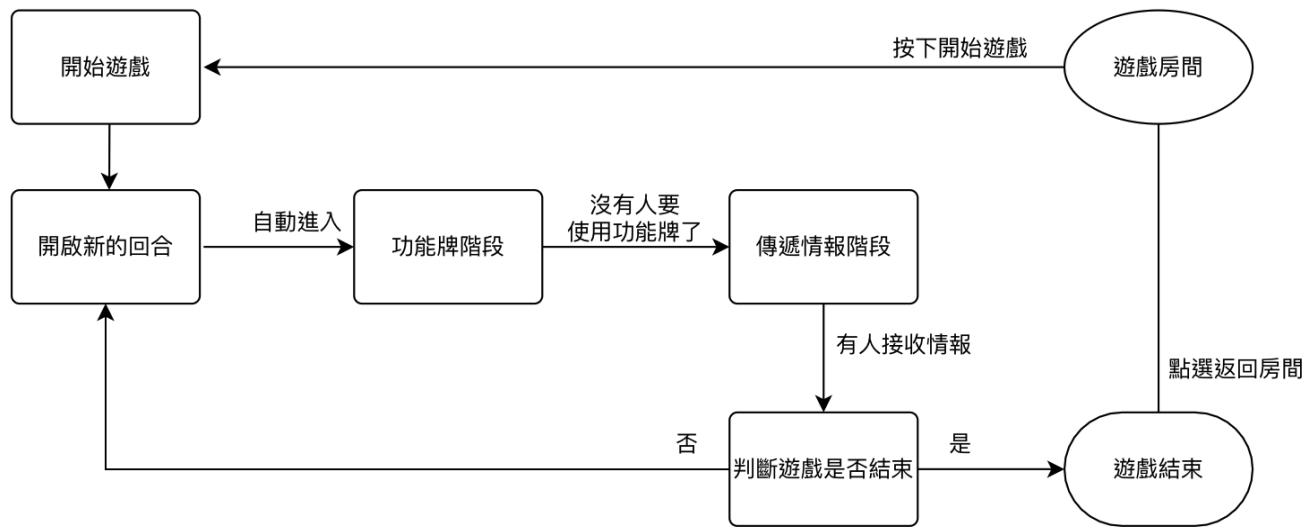
- **識破**: 當有功能牌被打出之後可以使用此牌。使該被打出的牌(包括其他識破)無效化，並放入棄牌堆。
- **試探**: 只能在自己回合使用。面朝下遞給一位玩家。被試探的玩家必須根據自己的身份作出行動，不可隱瞞。結束後將這張卡移除這場遊戲。
- **破譯**: 當情報傳遞到面前的時候可以使用，可檢視該情報。
- **鎖定**: 在自己回合中對(自己以外)任意一位玩家使用。本回合傳出的情報若傳遞到該玩家，則該玩家必須接收。
- **調虎離山**: 當情報開始傳遞後使用。指定一位自己以外的玩家不參與本次的情報傳遞。
- **真偽莫辨**: 只能在自己回合使用。從牌庫頂將等同於現存玩家數的卡牌從自己開始，逆時針分配給每人一張作為情報。
- **退回**: 當情報傳遞到面前時使用。該情報以反方向傳遞(若已被鎖定則此牌無法發揮作用)
- **截獲**: 在他人回合的情報傳遞中使用。獲得一張傳遞中的情報(不受**鎖定**和**調虎離山**的影響，也不受傳遞方式的限制)
- **燒毀**: 在功能牌階段使用。燒毀任意一位玩家的一份黑色情報。

作為情報：僅需考慮卡片顏色和右上角傳遞方式。卡片顏色包括紅、藍、黑三色，收到三張黑色情報的玩家會死亡。傳遞方式分為密電、文本、直達，密電會逆時針覆蓋傳遞，文本逆時針正面傳遞，直達則是覆蓋住直接傳遞至選擇的一位玩家面前，當情報無人接收而回到發出者時，發出者必須接收。

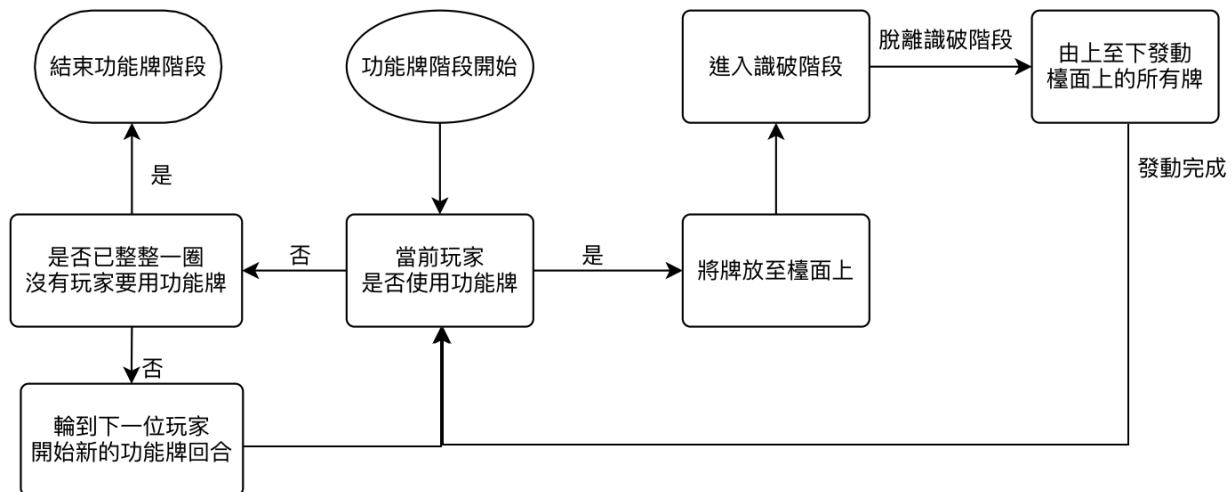
遊戲流程

玩家輪流傳遞情報，每個回合會有一位玩家要傳遞情報，他會先從牌堆頂端抽兩張手牌，情報傳遞前、傳遞中至接收前，皆可使用符合時機的功能牌，由於網路通訊難以實踐實體遊玩時多位玩家同時使用功能牌時可以討論讓誰用的情況，因此我們改採輪流決定是否出牌的方式，如下列流程圖所示。遊戲持續至有任意玩家獲勝，若只剩下一位玩家存活，則該位玩家及其同陣營玩家獲勝（打醬油的玩家各自為不同陣營）；若有玩家在遊戲中沒有手牌可以傳情報，則直接死亡並輸掉遊戲，即使隊友獲勝也無法取得勝利。

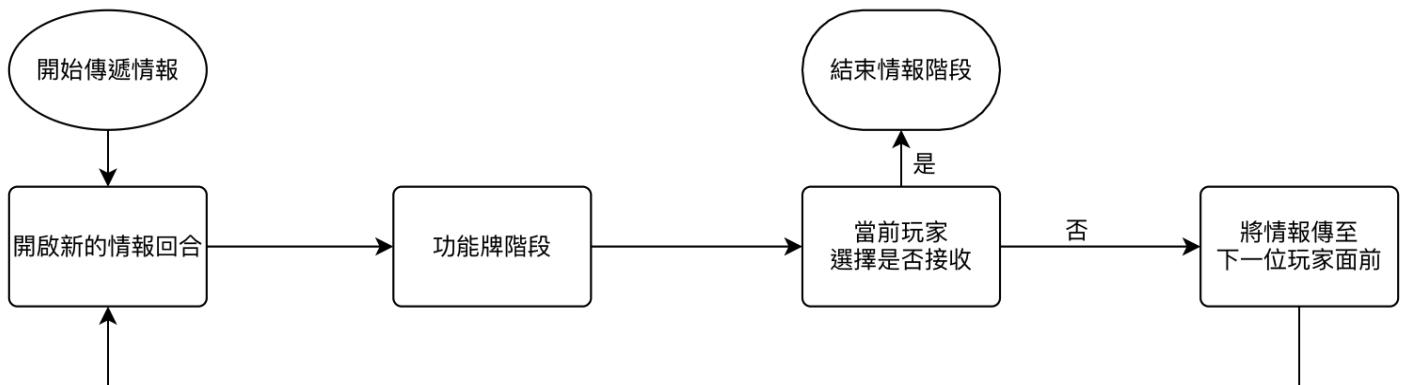
主要遊戲回合 (MainRound)



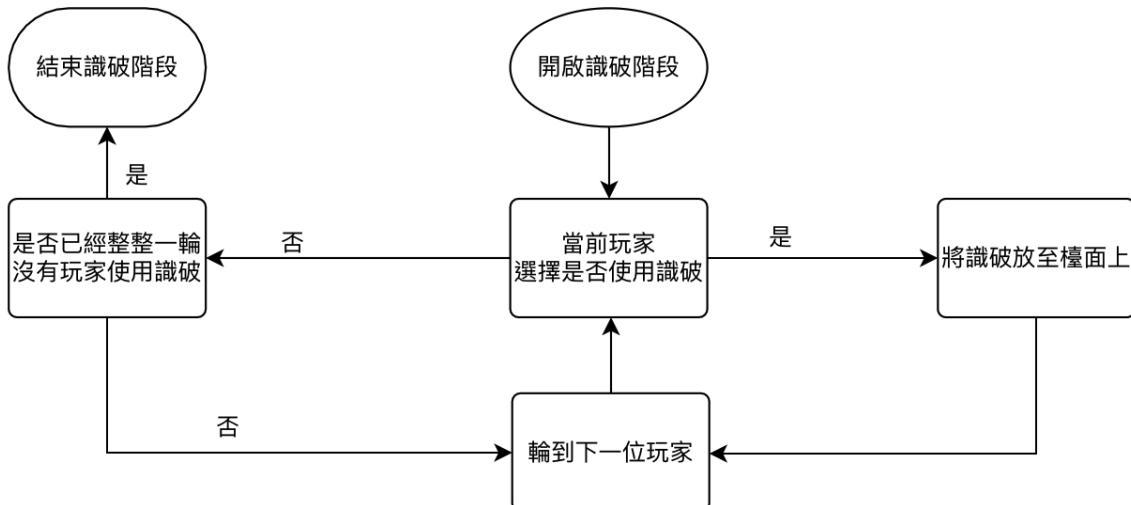
功能牌回合 (GameCardRound)



傳遞情報回合 (IntelligenceRound)



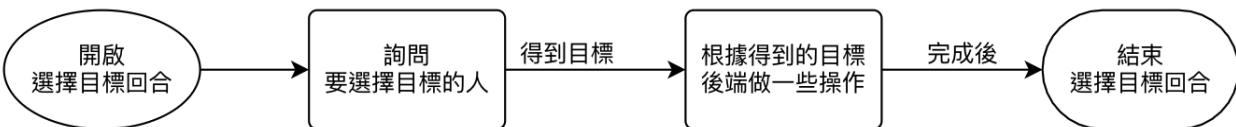
識破回合 (CounteractRound)



試探回合 (ProveRound)



選擇目標回合 (TargetSelectRound)



介面設計 (前端)

使用 ReactJS 進行撰寫、UI 設計方面引入 Material UI 來使用。

前端會打 api 來反應玩家動作，而後端則會透過 websocket 來廣播或通知前端更新UI

首頁

輸入 username 登入

→ 建立房間 or 輸入 房間代碼 加入房間



房主可選擇開始遊戲，其他人只能選擇離開房間；房間上方有代碼，可讓其他人用以加入房間。



遊戲介面

底下的區塊為己方的介面，包含左邊的遊戲訊息(每位玩家看到的訊息會客製化)、中間的手牌和提示訊息(同樣客製化)、和右邊的角色資訊(使用者姓名、角色圖片、陣營、情報數量和手牌數量)；中間的區塊為這一輪被打出的功能牌，在功能牌失效後就會消失；其餘兩邊為其他玩家的資訊(當玩家人數不同時會自動排成不一樣的版面)，若角色為潛伏角色就不會顯示圖片。圖上左方小張的卡牌為正在傳遞中的情報，會按照玩家順序傳遞給下一位玩家。



游標在玩家上時，可查看角色詳細資訊



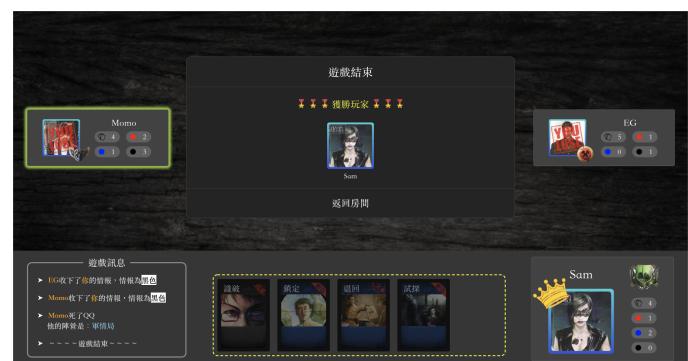
當某些功能卡作用於玩家時，會顯示狀態，
圖為調虎離山的效果



游標在卡牌上時，可查看卡牌詳細資訊



遊戲結束時，會顯示勝利玩家，所有玩家
陣營也會出現。



軟體設計(後端)

The Relations Between the Classes

User

定義了使用者的基本資訊，包含名字、唯一的 ID、位於哪間遊戲房間以及 WebSocket 連線的資訊。這個 class 與 Player class 進行區分是為了讓之後擴展遊戲需要帳號密碼登入時，能夠直接使用這個 class 進行擴充而不是使用Player class(屬於遊戲的一部分)，以達到更好的區隔。

| User |
|----------------------|
| - name : char[] |
| - id : char[] |
| - currentRoom : Room |
| + leaveRoom() : void |

Room

定義了玩家玩遊戲的房間和這個房間能最多能容納的使用者數目，同時定義了一個 Game class 當作這間房間所開啟的遊戲。

| Room |
|--------------------------------|
| - id : char[] |
| - users : User[] |
| - game : Game |
| - isPlaying : boolean |
| - minPlayerNum : int |
| - maxPlayerNum : int |
| + joinRoom(user : User) : void |
| + startGame() : void |
| + onGameOver() : void |

Game

負責紀錄整個遊戲的一些資訊(包含玩家、手牌、已打過牌堆、抽牌堆、場上已打出尚未生效的牌、死亡的玩家)，並包含了許多查詢與判斷的 function 提供給Round class(遊戲流程的 class)使用。

| Game |
|--|
| - players : Player[] |
| - users : User[] |
| - round : Round |
| - characterCardDeck : Character[] |
| - gameCardsDeck : Gamecard[] |
| - playedCards : Gamecard[] |
| - currentActionsOnBoard : GameCardAction[] |
| + initializeStage() : void |
| + createCharaterCardDeck() : void |
| + createGameCardDeck() : void |
| + onGameOver() : void |
| + placeGameCardActionOnBoard(action : GameCardAction) : void |
| + takeActionsOnBoard() : void |
| + getWinners() : Player[] |

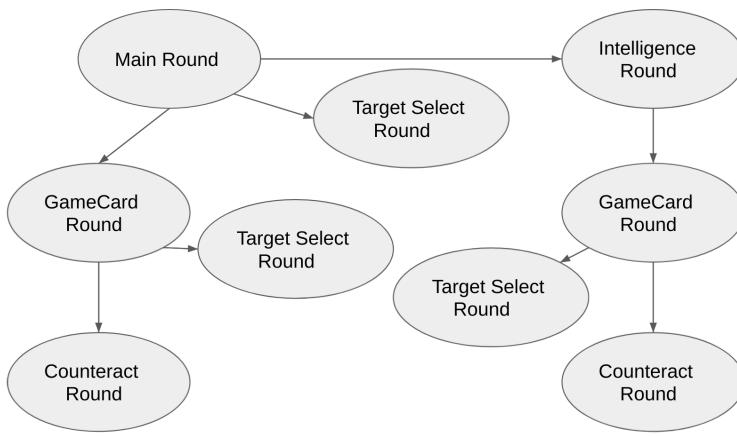
GameCard

定義了卡牌的基本資訊，可以讓各種不同的功能牌繼承，分別定義每張不同功能牌的效果，達到OCP的效果。

| Gamecard |
|---------------------------------------|
| - id : String |
| - name : String |
| - timingDescription : String |
| - effectDescription : String |
| - intelligenceType : IntelligenceType |
| - needTarget : boolean |
| - targetSelf : boolean |
| + isValid() : boolean |
| + perform() : void |

Round

負責掌管整個遊戲的流程，屬於整個遊戲的核心。我們在這個 class 中將整個遊戲分成多個階段，並且巧妙地利用了 LinkList 的想法，將整個複雜地遊戲很好地化簡。我們在這個 class 中定義了 parent Round、child Round，讓整個遊戲流程能夠有架構，同時讓有一些會改變 game flow 的功能牌，能夠在這個架構上進行擴充，達到 OCP 的效果。



| Round |
|---|
| <ul style="list-style-type: none">- endPlayer : Player- creator : Player- parentRound : Round- childRound : Round- direction : int- game : Game- name : String <ul style="list-style-type: none">+ Round(game : Game)+ Round(parentRound : Round)+ setEndPlayer(endPlayer : Player) : void+ getEndPlayer() : Player+ setCurrentPlayer(currentPlayer : Player) : void+ getCurrentPlayer() : Player+ getGame() : Game+ getParentRound() : Round+ setChildRound(childRound : Round) : void+ getNextPlayer() : Player+ onRoundStart() : void+ onTurnStart() : void+ onTurnProgressing(action : Action) : void+ doWhenLeaveChildRound() : void+ onTurnEnd() : void+ onRoundEnd() : void+ satisfyRoundEndCondition() : boolean+ getName() : String+ isGameCardRound() : boolean+ isCounteractRound() : boolean+ parentRoundIsMainRound() : boolean+ parentRoundIsIntelligenceRound() : boolean+ playerIsOwnerOfParentRound() : boolean+ playerIsCurrentPlayerOfParentRound() : boolean |

Character

定義了角色的一些基本資訊，包含名字、性別、機密任務、技能與是否為潛伏的資訊。不同角色繼承同個 Character class，使得不同地角色擁有不同的機密任務、技能、性別與潛伏資訊，能夠很好地擴充能力，達到 OCP 的效果。

| Character |
|--|
| <ul style="list-style-type: none">- name : String- gender : Gender- mission : Mission- hidden : boolean <ul style="list-style-type: none">+ cover() : void+ uncover() : void+ missionComplete() : boolean |

Mission

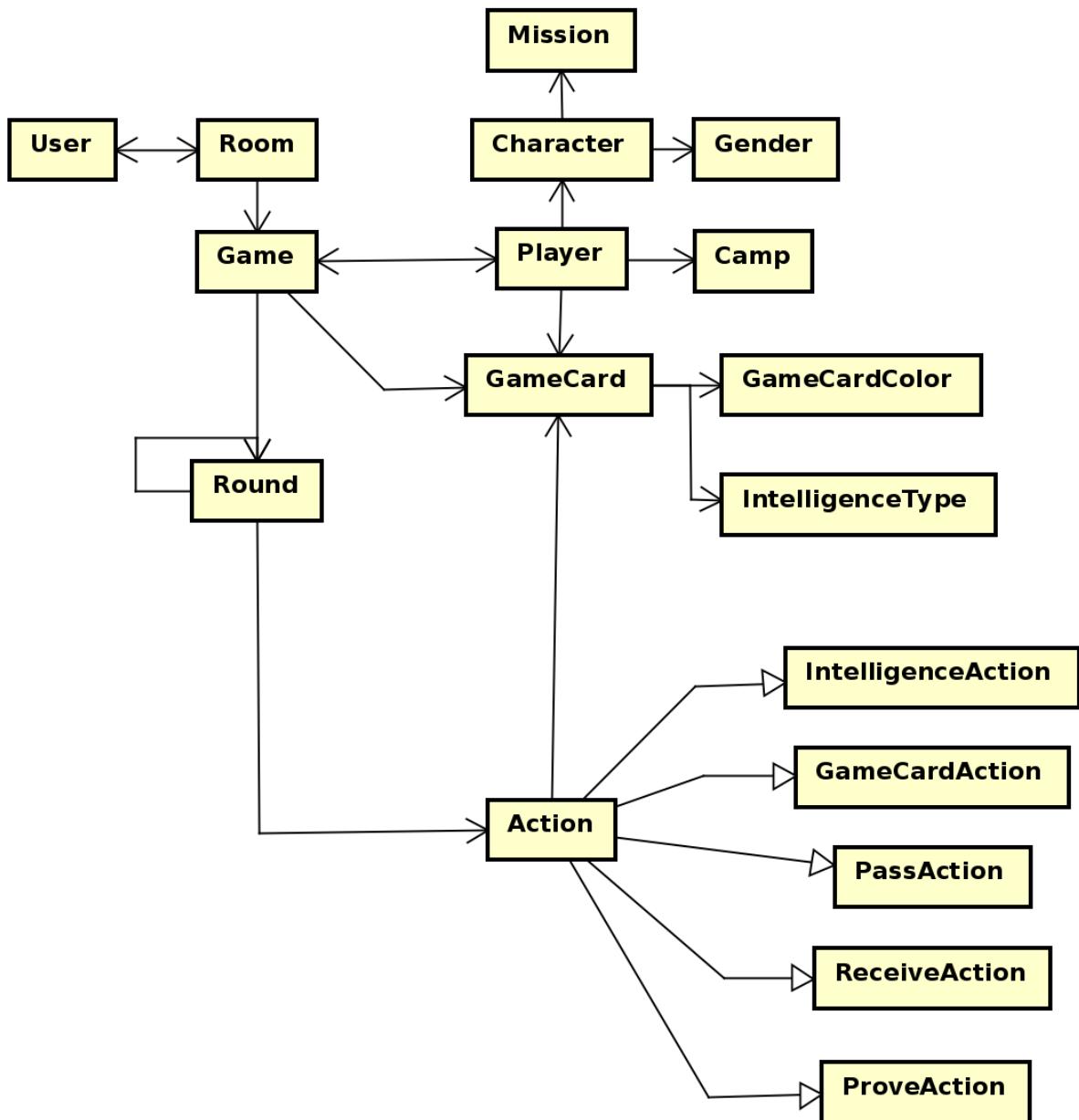
定義了角色的機密任務，每個不同的 character class 對應到一個 mission (e.g. 角色「浮萍」對應到一個 class FuPingMission)。

| Mission |
|--|
| <ul style="list-style-type: none">- description : String <ul style="list-style-type: none">+ isCompleted() : boolean |

Player

定義了遊戲玩家擁有的基本訊息(包含唯一的Id、角色、陣營、狀態、持有的情報、屬於哪位使用者、殺死了哪些 Players、被哪個Player 殺死)。

| Player |
|--|
| <pre>- id : char[] - gender : Gender - mission : Mission - camp : Camp - status : PlayerStatus - handCards : Gamecard[] - intelligences : Gamecard[][] - game : Game - user : User + drawCards() : void + removeCardFromHandCards(card : Gamecard) : void + receiveIntelligence(receiveAction : ReceiveAction) : void</pre> |



The Advantage of Our Design

1. 在處理每種功能卡時，我們定義功能卡有共同的方法execute()，使得功能卡多型的方式，以達成功能卡擴充OCP。
2. 在處理gameflow的時候，我們將 Round 系分成 Start 和 End 兩個階段，而在每個Round 輪到誰我們將之定義為Turn，同時我們又將 Turn 細分成 Start / Progressing / End 三階段，Start 負責通知輪到的玩家進行動作，Progressing 負責處理由玩家選擇傳過來Server的行動，最後在End 進行一些結束條件的檢查，並輪到下一位玩家，並回到 Turn Start 廣播通知下一位玩家進行動作。這樣好處是在如果功能牌會破壞遊戲邏輯時，便可自己定義新的Round，並實作好Round Start / End 以及 Turn Start / Progressing / End 方法，即可改變gameflow邏輯，達到OCP擴充的效果。

The Disadvantage of Our Design

1. 在處理機密任務 (mission) 的時候，因為每個角色的機密任務相差甚遠，因此在某些部分的處理有根據不同角色做出些特殊判斷的地方，在這個部分沒辦法做到非常 OCP。
2. Skill 的部分亦同，特判的部分太多，因此最後在沒辦法想到如何接入的情況下，我們將之從原本的風聲桌遊中移除，因此玩起來變化較少。
3. 我們在程式前後端的連接技術部份不夠強，因此只能回合制的詢問出功能牌或識破，但真正在桌遊中是直接喊出來。相較起來，我們的版本節奏會慢很多，這是我們未來需要加強的。

Other Packages We Have Used

1. **Org.json**: 來為前後端溝通所需的json格式進行轉換。
2. **javax.websocket**: 建立WebSocket與前端進行連線，使server能夠主動傳訊息給client。
3. **Spring boot web RestAPI**: 使用spring boot所提供的controller功能進行傳訊。