程式設計(112-1) 期末專案

遊戲主題:不公平的 21 點

組別:第 29 組

B09610044 生傳三 林泊里 ｜ B11104001 心理二 吳姵蓁

B12701224 工管一 陳凱維 ｜ R12K45012 奈米工科碩一 支昱丹

一、專案主題

本專案主題為「不公平的 21 點」, 主要將以兩位玩家與莊家(**`Enemy`**)對決, 玩家則有兩種角色可以選擇, 分別為觀察者(**`Seeker`**) 與追跡者(**`Targetor`**)。玩家可以在 21 點牌局中使用角色技能, 並於使用技能後決定是否要加牌。遊戲將進行至兩位玩家皆不再加牌或是爆點出局為止, 最後在比較大小的階段後, 輸家將可獲得道具卡並嘗試翻轉勝負。

(一) 規則介紹

1. 目標:盡可能讓手牌的點數總和接近 21 點, 但不超過 21 點。
2. 遊戲回合:可選擇是否使用技能、是否要加牌。
3. 爆點出局:若手牌點數總和超過 21 點, 則失去勝利資格。
4. 使用道具卡:輸家可在 Item Round 使用道具卡, 嘗試翻轉勝負。
5. 最終勝者:使用道具牌後, 再次比較點數, 最接近 21 點的玩家獲勝。

(二) 遊戲流程(此部分將在說明系統設計時詳細討論)

1. 玩家創建名稱與選擇 character, 這時系統會使用 try-catch 確保玩家輸入正確, 並使用玩家名稱去讀取或創建一份記錄玩家戰績的 .txt 檔案。
2. **`playerMove()`**:玩家先選擇是否使用技能
    1) 使用技能:選擇使用幾號技能, 使用技能後進入抽卡
    2) 不使用技能:系統詢問是否加牌
3. **`playerDraw()`**:玩家選擇是否要加牌
    1) 加牌:隨機抽取一張, 並檢查是否超過21點。
    2) 不加牌:輪到下一位玩家。
4. 若任一玩家超過 21 點或是所有玩家皆選擇不再加牌:
    1) 進行勝負判定
    2) 輸家可決定是否進入 Item Round
5. Item Round:本遊戲的 Bonus 回合, 輸家可使用道具卡嘗試翻轉勝負。

二、系統設計

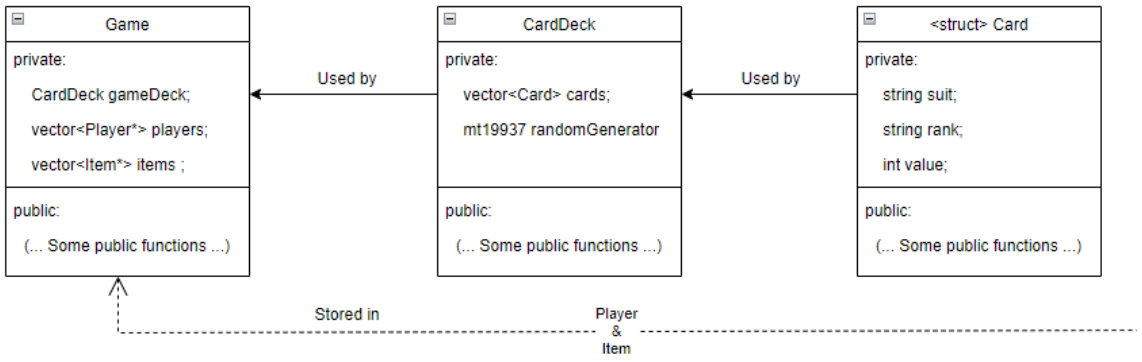
（一）Class Introduction

1. 撲克牌 **Card**:成員變數包含花色 **suit**、牌面數字 **rank**、換算點數 **value**。
2. 牌組 **CardDeck** :一個 **CardDeck** 物件管理著 52 個 **Card** 物件。
3. 玩家 **Player**:是一個純粹抽象類別, 被 **Seeker**、**Targetor**、**Enemy** 等三個衍生的子類別繼承, 分別代表本遊戲中不同的 characters, 可以使用不同技能。
4. 道具卡 **Item** :是一個純粹抽象類別, 被 **RandomSwitch**、**DrawOneFoldOne**、**BothFoldOne** 等三個衍生的子類別繼承, 分別代表不同功能的道具卡。
5. 牌局 **Game** :控制整個遊戲的進程, 一個 **Game** 物件會管理一個 **CardDeck** 物件、一個存放所有 **Player*** 的 Vector、以及一個存放所有 **Item*** 的 Vector。
6. 記錄檔 **Record** :用於紀錄玩家的戰績數據, 需要讀取和寫入 .txt 檔案。

（二）Class Diagram

Game 與 **Player** 為此遊戲中最重要的兩個 class, 與其他的 class 密集互動。

1. Game class

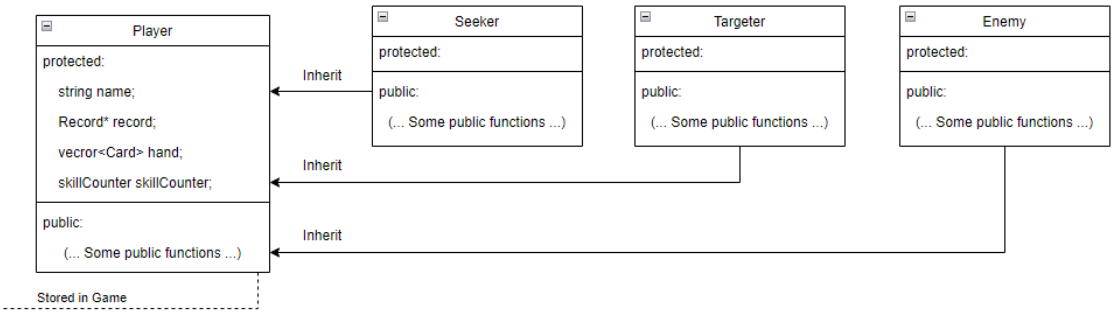


Game 在成員變數中存放 **CardDeck**、所有的 **Player** 、以及所有的 **Item** 物件, 並透過眾多 public functions 控制遊戲的進程, 是整個 project 最繁複的部分。

- **重要 functions**:

1) **initialDeal()**:牌局的初始化, 包含清空手牌、洗牌、並發給每位玩家兩張牌。
2) **drawRound()**:重複進行21點牌局, 詢問玩家是否使用技能與加牌。
3) **itemRound()**:遊戲的 Bonus 回合, 讓玩家使用 **Item** 物件。
4) **addPlayers(Player* player)**:讀入玩家資訊, 建立指定種類的 **Player** 物件。
5) **addItem(Player*& player)**:詢問玩家要新增何種 **Item** 物件到遊戲。
6) **result(bool whoWins)**:使用 operator overloading 判斷玩家的勝負。

2. Player class



`Player` 作為一個 pure abstract class, 被三個衍生的子類別 **`Seeker`**、**`Targetor`**、**`Enemy`** 繼承, 其中 **`Enemy`** 在本遊戲中作為莊家, 另外兩種則是玩家可以選擇的 characters。經由 **`Game`** 和 **`Player`** 的 public functions 控制, 玩家可在合適的時機使用以下的角色技能:

1) Seeker 觀察者

   `SeekDeck()`:查看牌組剩餘哪些牌

   `SeekAnotherPlayer()`:或查看一個玩家的手牌

2) Targetor 追跡者

   `takeSpecificCard()`:取得牌組中某一張牌

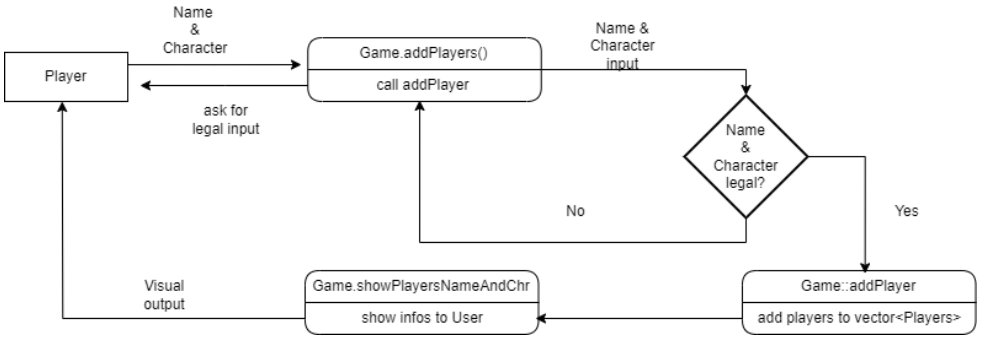
   `discardCard()`:或指定一個玩家必須丟棄某個 value 的牌

其它重要 functions:

1) **`playerMove(gameDeck, game)`**:玩家行動, 決定是否使用角色技能
2) **`playerDraw(temp, gameDeck)`**:玩家要牌, 使用完技能決定是否要加牌

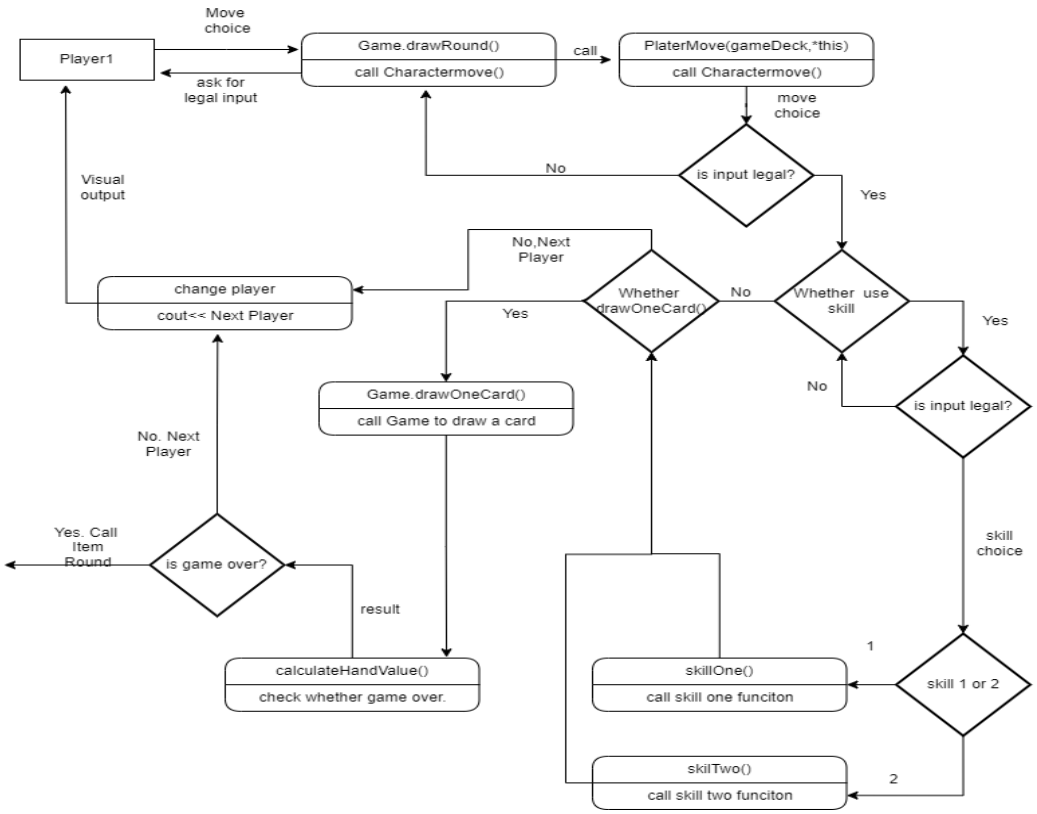
(三) Some Data Flow Diagram

1. `addPlayers()`: About Input Controls



本遊戲在程式碼中大量使用了 try-catch 以判斷玩家的輸入是否合裡。如圖, 例如在系統執行 **`addPlayers()`** 時, 玩家必須輸入合理的 char 以選擇對應的 character, 若新增成功會由 **`showPlayersNameAndChr()`** 印出建立的 **`Player`** 資訊, 否則會 throw errors 並重複迴圈。

2. `drawRound()`：21 點牌局設計



21 點牌局的進程將由一系列的判斷決定, 玩家可先決定是否使用技能, 若決定使用技能, 則在選定後呼叫該 character 的技能 function。在使用完技能或決定不使用技能後, 都將進入加牌環節, 玩家會被詢問是否選擇繼續加牌, 若選擇加牌則系統會呼叫 **`Game`** 類別的 **`drawOneCard()`**, 並在加牌後執行總手牌點數 **`totalValue`** 的檢查。若未超過21點, 回合將繼續進行, 輪到下一位玩家;若超過21點, 則不能再繼續加牌。倘若在加牌環節時, 兩位玩家都選擇不繼續加牌, 就結算各玩家的手牌點數, 輸家選擇是否進入 Item Round。

以下為 **`Game`** 中的 **`drawRound()`** 與 **`playerMove()`** 的 pseudocode:

```
procedure drawRound():  // 21 點牌局迴圈
    repeat
        // 由敵人先叫牌
        getPlayerAtIndex(2)->playerDraw(tempEnemy, gameDeck)
        getPlayerAtIndex(2)->showHand()

        for i = 0 to getNumPlayers() - 1:  // 玩家輪流叫牌
            getPlayerAtIndex(i)->playerMove(gameDeck, *this) // 是否使用技能
            if i == 0:
                getPlayerAtIndex(i)->playerDraw(tempPlayer1, gameDeck)// 是否加牌
            else if i == 1:
                getPlayerAtIndex(i)->playerDraw(tempPlayer2, gameDeck)
    until (tempPlayer1 != 1 and tempPlayer2 != 1) or (tempPlayer1 == 1 and
tempPlayer2 != 1) or (tempPlayer1 != 1 and tempPlayer2 == 1)  // 牌局結束的條件
```

```
function playerMove(gameDeck, game):  // 判斷是否使用技能
    此處先展示玩家的手牌與角色名稱
    char move = getUserMove()  // 判斷使用哪個技能
    switch move:
        case '1':
            skill1(); break
        case '2':
            skill2(); break
        case 'N':
            break
end function
```

```
function getUserMove(): // 判斷使用哪個技能
    char move;
    while true:
        print "Do you want to use your skill? (1: skill1, 2: skill2, N: Do
Nothing): "
        input move
        if move == '1' or move == '2' or move == 'N':
            break  // 輸入正確，跳出迴圈
        else:
            print "Invalid move. Please enter 1 or 2 or N."
    return move
end function
```

```
function playerDraw(temp, gameDeck): // 玩家抽卡，用 temp 紀錄玩家是否選擇繼續抽卡
// 若不抽卡或是手牌點數超過 21 點時則將 temp 設為 1 回傳。反之，則回傳 0

    if calculateHandValue() >= 21:  // 判斷是否超過21點
        print "- You cannot draw anymore..."
        temp = 1
    else:
        char move
        while true:  // 用 try-catch 直到輸入正確
                if move == 'Y':
                    randomlyAddOneCard()  // 隨機抽一張卡
                    temp = 0
                    if calculateHandValue() >= 21:  // 檢查是否超過21點
                        temp = 1
                        showHand(); break;
                else if move == 'N':  // 不動作
                    temp = 1; break;
                else:                          // 用 try-catch 控制 input
                    temp = 0;
                    print "Invalid move. Please enter Y, N."
                    clearInput();
end function
```

（四）Some other Algorithm

- **勝負判定**:

在判定勝負時, 有可能遇到玩家點數平手的情形, 依照21點規則, 此時應比較玩家的手牌數量來決定勝負, C++ 內建的 operator 無法直接達成需求, 故需使用 operator overloading , 在 **`Player`** 類別中定義每一個運算子(<、>、==、>=、<=) 的判斷條件為:

1. 先比較兩個玩家的手牌值(**`calculateHandValue()`**)是否相同。
2. 若相同, 則進一步比較兩者之手牌數量(**`hand.size()`**)。
3. 以下為判斷完上述條件後, 各運算子的運行邏輯:

   operator<(小於):
   - 比較手牌數量後返回 **`hand.size() < other.hand.size()`**
   - 如果手牌值不同, 則直接返回 **`calculateHandValue() < other.calculateHandValue()`**

   operator>(大於):
   - 比較手牌數量後返回 **`hand.size() > other.hand.size()`**
   - 如果手牌值不同, 則直接返回 **`calculateHandValue() > other.calculateHandValue()`**

   operator>=(大於等於):
   - 比較手牌數量後返回 **`hand.size() >= other.hand.size()`**
   - 如果手牌值不同, 則直接返回 **`calculateHandValue() >= other.calculateHandValue()`**

   operator<=(小於等於):
   - 比較手牌數量後返回 **`hand.size() <= other.hand.size()`**
   - 如果手牌值不同, 則直接返回 **`calculateHandValue() <= other.calculateHandValue()`**

   operator==(等於):
   - 同時比較兩個玩家的手牌值和手牌數量。
   - 返回 **`( calculateHandValue() == other.calculateHandValue() ) && ( hand.size() == other.hand.size() )`**

四、實作到之技能

1. Operator Overloading:

利用 >、<、==、>=、<= 來進行手牌值的大小比較。

2. File I/O:

1) 在遊戲開始時讀取介紹 characters 的 txt 檔案
2) 利用 **`Record`** 類別來記錄玩家的最佳戰績, 包含讀取及寫入 txt 檔案

3. Inheritance:

1) 從 **`Player`** 類別衍生出 **`Seeker`**、**`Targetor`**、**`Enemy`** 等三個子類別, 分別代表本遊戲中不同的 characters。
2) 從 **`Item`** 類別衍生 **`RandomSwitch`**、**`DrawOneFoldOne`**、**`BothFoldOne`** 三個子類別, 分別代表遊戲中不同的道具卡。

4. Polymorphism:

1) 用 **`Player*`** 的 vector 來儲存不同子類別的 object。

   在 **`Player`** 類別中宣告 pure virtual function

   ```
   virtual void playerMove( ... ) = 0;
   virtual void playerDraw( ... ) = 0;
   ```

   並在子類別 **`Seeker, Targetor, Enemy`** 中實作不同的 function 定義。

2) 用 **`item*`** 的 vector 來儲存不同子類別的 object。

   在 **`Item`** 類別中宣告 pure virtual function

   ```
   virtual void useItem( ... ) = 0;
   ```

   並在子類別 **`RandomSwitch`**、**`DrawOneFoldOne`**、**`BothFoldOne`** 中實作不同的 function 定義。

5. Exception Handling:

使用 try-catch 區塊來處理無效的輸入, 例如:

1) 在 **`addPlayers()`** 中, 捕捉對玩家人數的無效輸入的例外。
2) 在 **`playAgain()`** 中, 捕捉對是否再次遊玩的無效輸入的例外。

五、分工方式

林泊里:

- class diagram 繪圖、data flow diagram 繪圖、製作書面報告
- coding: **`Player`** class, **`Seeker`** class, **`Targetor`** class, main 與 **`Game`** class 的 **`addPlayers()`**、**`playerMove()`**

吳姵蓁 :

- 錄製展示影片、製作展示簡報
- coding : **`Item`** class 與相關 code , old version main funtion , 中文版修改

支昱丹:

- 書面報告增修
- coding:簡易 21 點草案、**`playerDraw`**(enemy, seeker, targetor)、operator overloading

陳凱維:

- 錄製家庭專案影片、書面報告增修與排版
- coding:**`Record`** class、遊戲排版設計與相關 code、整理 main.cpp 和 header files、與 **`Game`** class 的 **`itemRound()`**、**`addItem()`**

六、心得

- 林泊里:在這次的專案中, 每個成員都有獨特的思維模式, 對同樣的問題有不同的解決方案。此外, ChatGPT 的出現讓我們能夠簡單的獲得程式碼問題的解決方案。雖然提高了效率, 但我也擔心個人 coding 能力的削弱。總而言之, 合作、思考、創新、溝通、工具, 都是專案中的重要因素, 這次的經驗讓我提昇了對這些面向的能力, 也增加更多的思考方向。

- 支昱丹:在本次期末專案中學到如何互相合作, 困難的地方大概是時間上不是很充裕, 因為剛好碰到期末考週, 所以大家都很爆肝, 以及我衷心感謝組員們都很用心而且很可靠的寫出了很棒的程式。

- 吳姵蓁 **:** 這次的專案製作過程非常爆肝, 當初決定主題時以為會挺容易的, 實際上手才發現了諸多困難, 編譯上總是能遇到各種問題, 很感謝我的組員人都很好也很配合, 有大家的努力才能共同完成這份作品。

- 陳凱維:這是我第一次學習用 c++ 物件導向的概念製作遊戲, class 之間的互動有時會讓我頭痛, 此時必須好好思考要代什麼參數進入函式才更方便達成目的。這也是我第一次和一個團隊分工合作完成一個較為大型的程式, 雖然因期末考將至, 為了想早點完成這份專案很爆肝, 不過對我而言, 這次專案是我在學習程式設計的路上相當寶貴的一段經驗, 我也很幸運這次能遇到三位熱心配合的組員, 獲得一次良好的合作體驗。