2021FOOP 期末專題

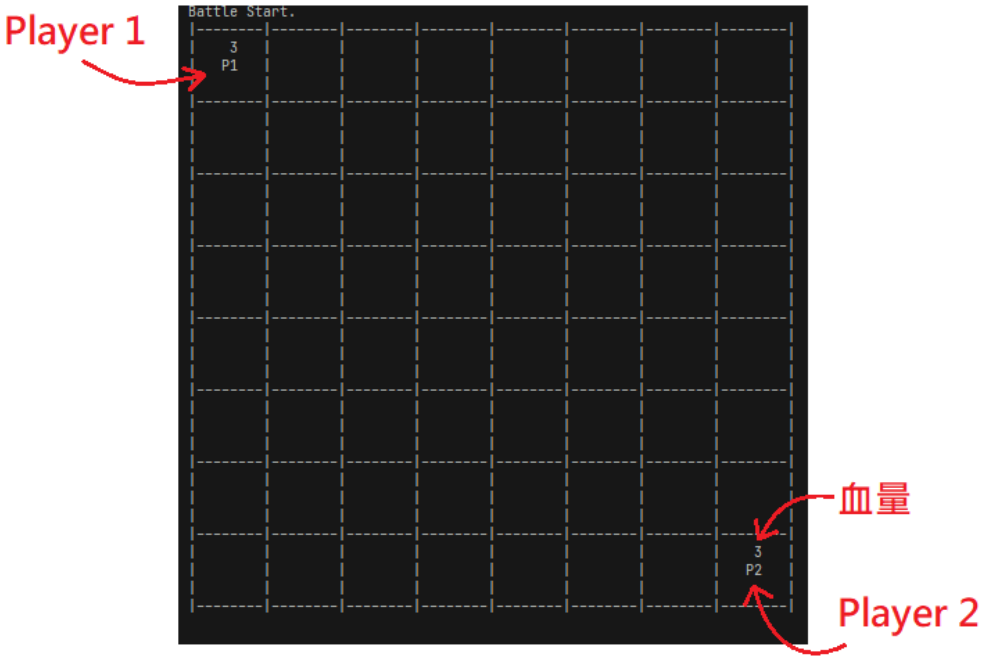
B06902076 林秉駿

分工：

1. 遊戲構思、設計、編寫：林秉駿

遊戲介面與功能：

目前僅有 Command-Line Interface，不過將來可以經由遵守 Open-Close Principle 的方式擴充到圖形介面。



此遊戲格狀的地圖（Map）上完成，地圖由 8x8 的方格（Square）構成，地圖上有兩個玩家，分別為 Player1 以及 Player2，他們的目標是要使另外一個玩家的血量歸零。玩家起始的血量是 3。

遊戲流程類似棋類遊戲，雙方交替地選擇行動（Action），最先使對方玩家的血量歸零者勝出。

而玩家可以選擇不同的角色（Character），原本計畫加入多一點角色，但目前只有兩種角色，騎士（Knight）與史萊姆（Slime）。不同的角色可選擇不一樣的行動，而所有角色都有一個共同的行動：Move，只要輸入的指令是 w,a,s,d 即可，它們分別對應上、左、下、右四個方向的移動，

遊戲一開始會需要先決定 Player 1 以及 Player 2 的角色為何，只要根據提示輸入對應的數字即可，遊戲開始後 Player 1 會是第一個可以動的玩家。兩位玩家的起始位置如上圖。

輪到該名玩家時，可以根據提示訊息輸入想選擇的行動，或是輸入 w,a,s,d 來移動，比方說

```
Player 1's move
[1] SwordAttack [2] Shield
1
Choose direction:
[d]: right, [s]: bottom, [a]: left, [w]: top
d
```

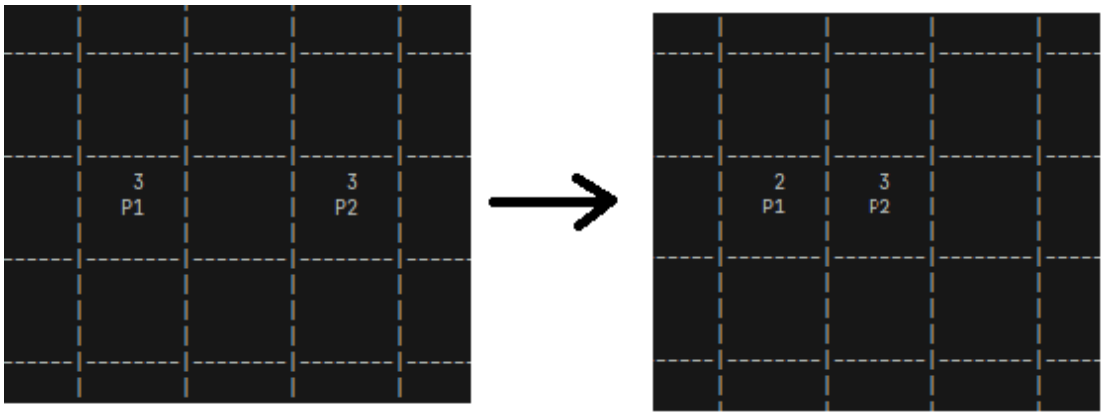
原本固定每個角色有三個 Action（除了 Move 以外），但是只來的及做以下非紅色的部分：

騎士 Knight：

1. Sword Attack 揮劍：向所選擇的方向攻擊三個格子的範圍(正前方以及左右各一格)，擊中目標會造成 1 的傷害。
2. Shield 舉盾：可以防禦對方造成的攻擊，只能有效一個回合。

史萊姆 Slime：

1. Lunge 飛撲：向所選擇的方向飛撲，最大距離兩個方格，如果撞到玩家會對該玩家造成 1 的傷害並停在前一格的地方。這個行動有冷卻時間 1 回合。也就是說史萊姆無法連續兩回合使用這個行動。如下圖所示(P2 是史萊姆)



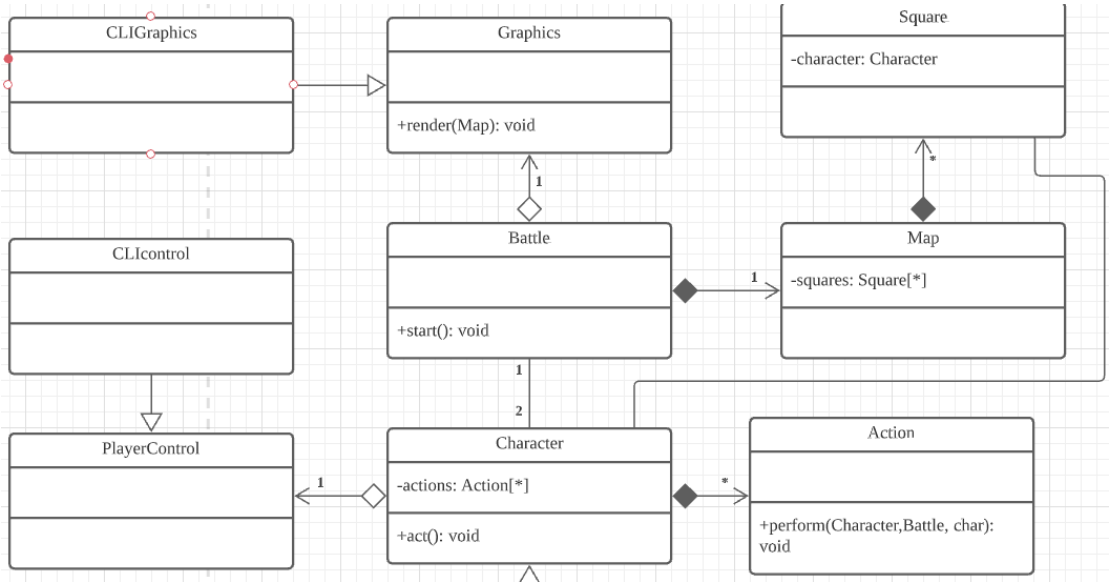
2. Split 分裂：史萊姆分裂成兩塊碎片，這兩塊的血量都是 1。分裂以後的史萊姆在選擇行動的時候需要對兩塊碎片都選擇一個行動。這個行動只能使用一次。
3. Poisonous Track 劇毒軌跡：史萊姆會把當下所在的方格變成有毒的方格，停在上面的角色每回合會造成 1 的傷害。史萊姆對有毒的方格免疫。

此外，如果角色的移動目標上面已經有其他角色在了，作為懲罰，執行移動的角色會受到反擊 1 的傷害。

因此可能會有 variation 的是：方格（Square）的效果、行動的範圍（Range）、行動（Action）的行為。

類別：

以下是內容物簡化過後的 class diagram：



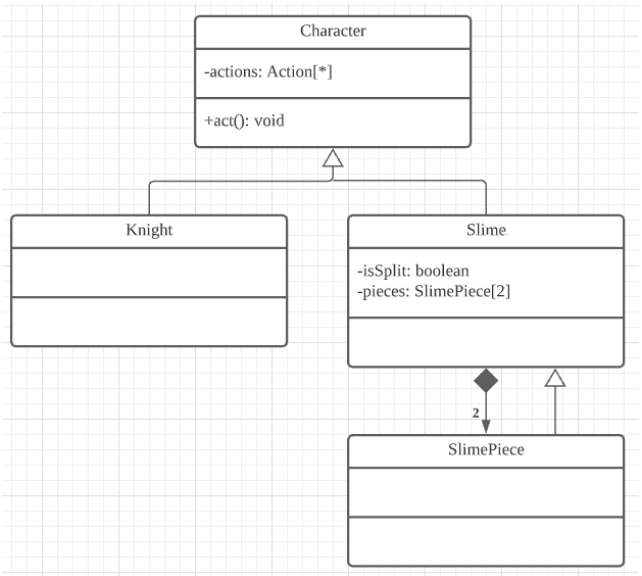
首先正中央的 Battle 表示戰鬥的流程，而它上方的 Graphics 是一個 Interface，負責畫面輸出，藉由 Dependency Injection，Battle 握有一個 Graphics 實例的 reference，代表 command-line interface 的 CLIGraphics 透過繼承 Graphics 完成擴充。

Battle 下方的 Character 表示角色，與 Battle 的關係為 Association，它持有一個 PlayerControl 的實例，同樣地，表示 command-line 輸入的 CLIControl 經由繼承 PlayerControl 完成擴充。

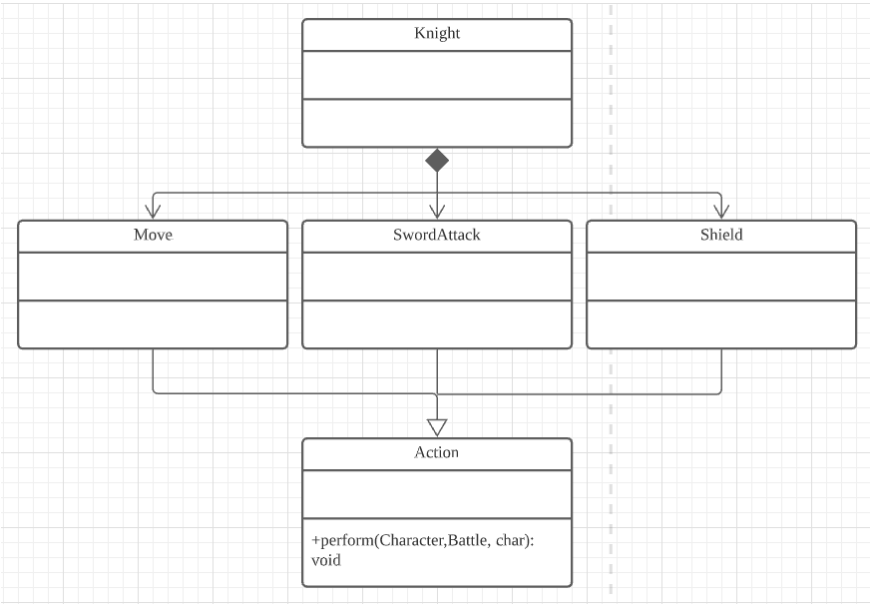
Character 需要自行將自身的 Action 實例化，因此它與 Action 的關係是 Composition。

最後 Battle 右側的 Map 是 8x8 的方格地圖，Map 是由眾多 Square 所組成，因此它們之間也有 Composition 的關係。Square 需要紀錄目前哪一個 Character 位於此，且 Character 也需要紀錄本身的位置，所以它們兩個有 Association 的關係。

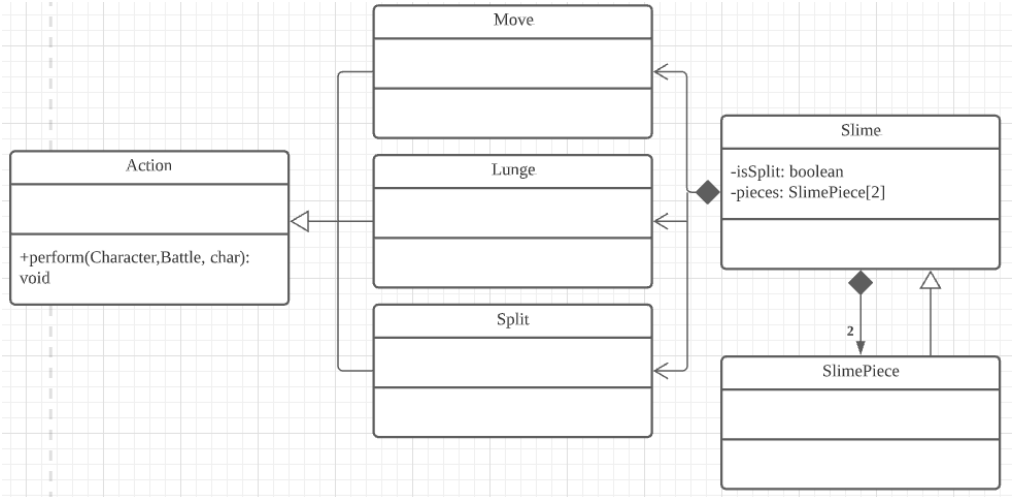
至於不同的角色這方面就是單純的繼承關係。



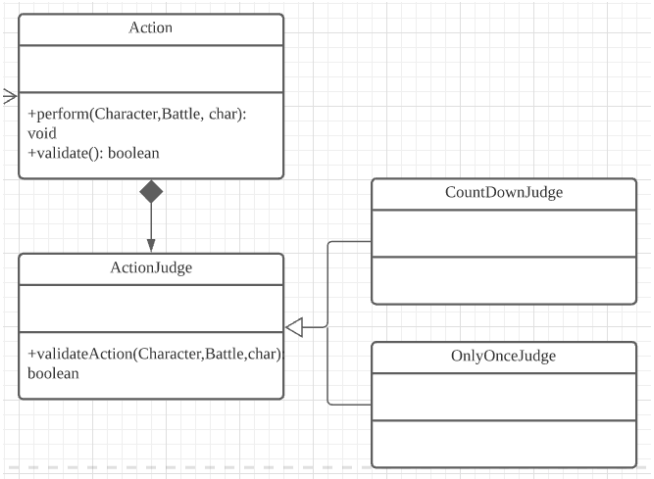
Knight 現有的三個 Action 都繼承了 Action 且與 Knight 之間保有 Composition 的關係。也就是說，每個角色都要實例化自己的 Action。



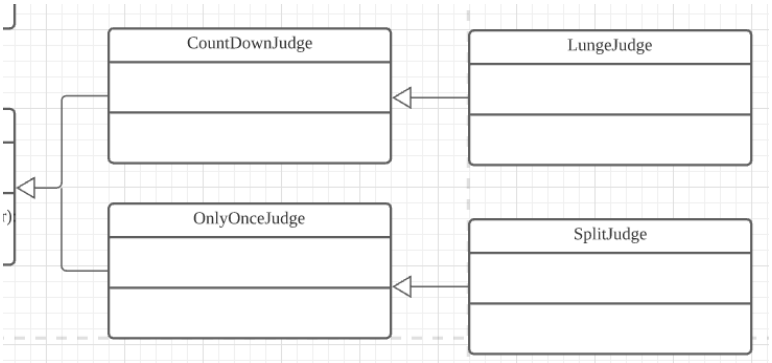
史萊姆也一樣。



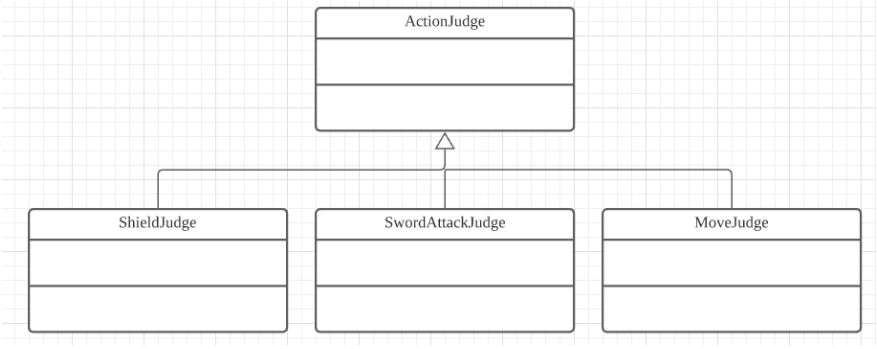
此外，這邊有一個特殊的 Class 叫做 ActionJudge，它負責判斷這個 Action 是否可以被執行。這個功能可以用在具有冷卻時間或是只能使用一次的 Action 上，因此分別衍生出了 CountDownJudge 跟 OnlyOnceJudge 這兩個類別。而如此一來，Action 這個類別的功能就變得更為單一——規定該 Action 是如何被執行的，而不必再同時肩負檢查是否可以執行的任務。



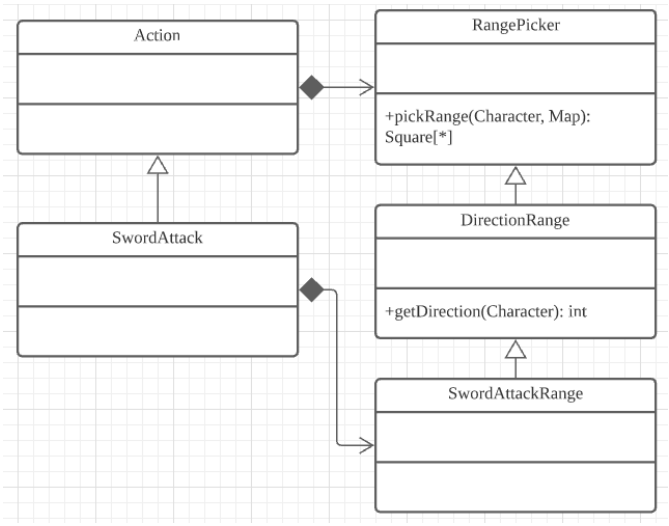
以史萊姆的兩個 Action：有冷卻時間的 Lunge 以及只能使用一次的 Split 為例，藉由分別繼承 CountDownJudge 與 OnlyOnceJudge，可以達成箭頭由外往內指的擴充。



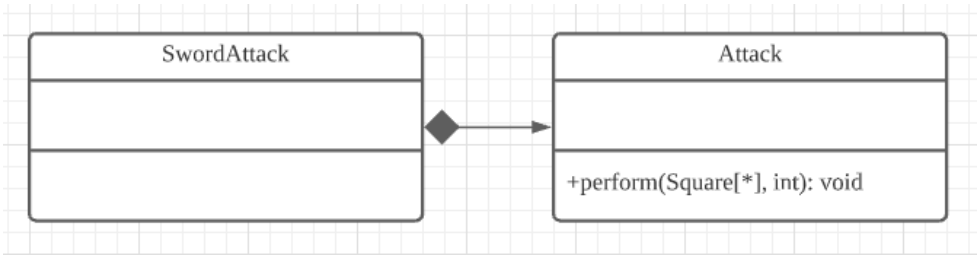
其他的 ActionJudge 為：



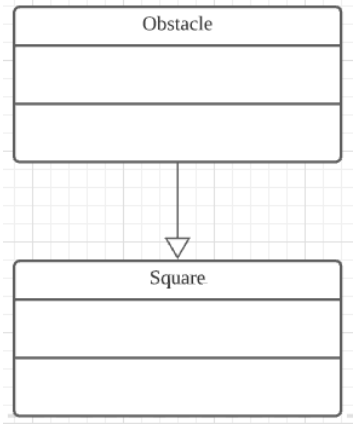
接著是 Range 的部分，這邊僅以 SwordAttack 為例。有鑑於某些 Action 有範圍這項特性，以及某些 Action 需要玩家進一步發出指令，受到 HW2 所啟發，為了能更有效重複使用程式碼，這邊誕生了 RangePicker 這個 Interface，而實作了這個 Interface 的 DirectionRange 代表繼承了它的類別需要玩家再次選擇方向已完成 Action 執行。由於理想狀態為 SwordAttack 同時繼承 Action 與 SwordAttackRange 來直接使用裡面寫好的程式碼，但礙於 java 無法多重繼承，這裡使用 Aggregation 來解決此問題。



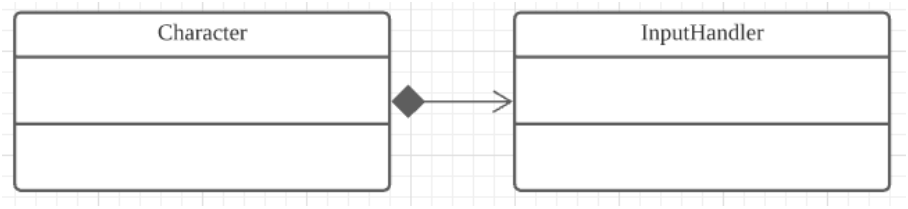
此外，還有一個類別名為 Attack 表示某個 Action 的行為是會對它選取的 Range 裡的角色造成攻擊，典型的例子為 SwordAttack，再一次地，透過 reference 的方式迴避多重繼承。



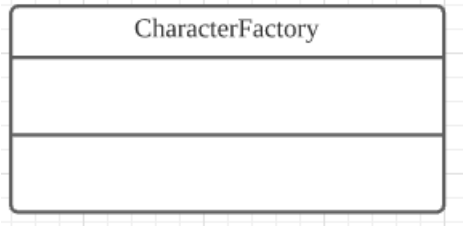
目前 Square 有的變種為 Obstacle，表示該 Square 為障礙物，無法移動進入。



最後為幾個用以輔助的類別：



InputHandler 負責將收到的 input 與要執行什麼樣的 Action 進行關聯。



CharacterFactory 在 main 函數裡使用，用以將使用者輸入的數字化為 Character 的實例，就如同一般的 Object Factory。

設計優點：

1. Action 與 ActionJudge 分開，可以有彈性地變更 Action 的種類
2. 把移動 Move 這個行為獨立成一個類別就不必在 Character 內寫 move()之類的函式，各類別的職責更為單一。

設計缺點：

1. Action 過於依賴 Character：例如 Split 這個 Action 只能給 Slime 使用，需再探討更直觀、安全的整合方式。
2. 難以跟畫面結合：由於目前僅有 Battle 握有 Graphics 的 reference，若是要在別處更新畫面，恐怕需要將 Graphics 當作參數經過層層的傳遞。

Other Package：

無

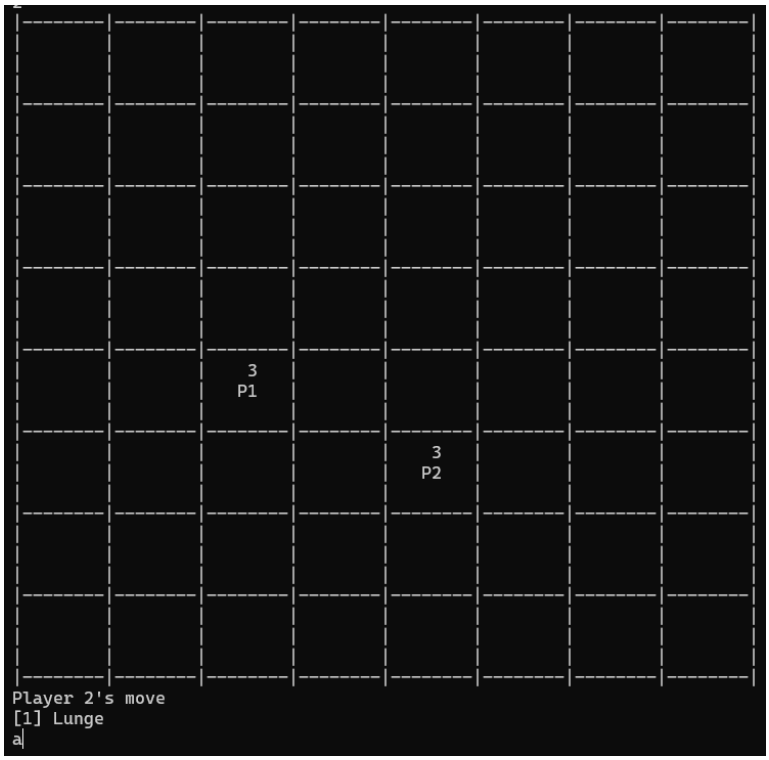
How to play：

1. 如同作業要同的，在含有 Makefile 的資料夾執行 make run 的指令
2. 依照訊息提示選擇 Player 1 以及 Player 2 的角色，如下：

```
Choose the character of player 1:
[0] Knight, [1] Slime
0
Knight
Choose the character of player 2:
[0] Knight, [1] Slime
1
Slime
```

3. 由於還沒有 AI 或者連線的功能，預設兩位玩家的控制都是透過同一個終端

來執行，透過單機多人遊戲的方式，在終端輸入指令輪流操控 Player 1 以及 Player 2：



4. 分出勝負，顯示結果訊息：