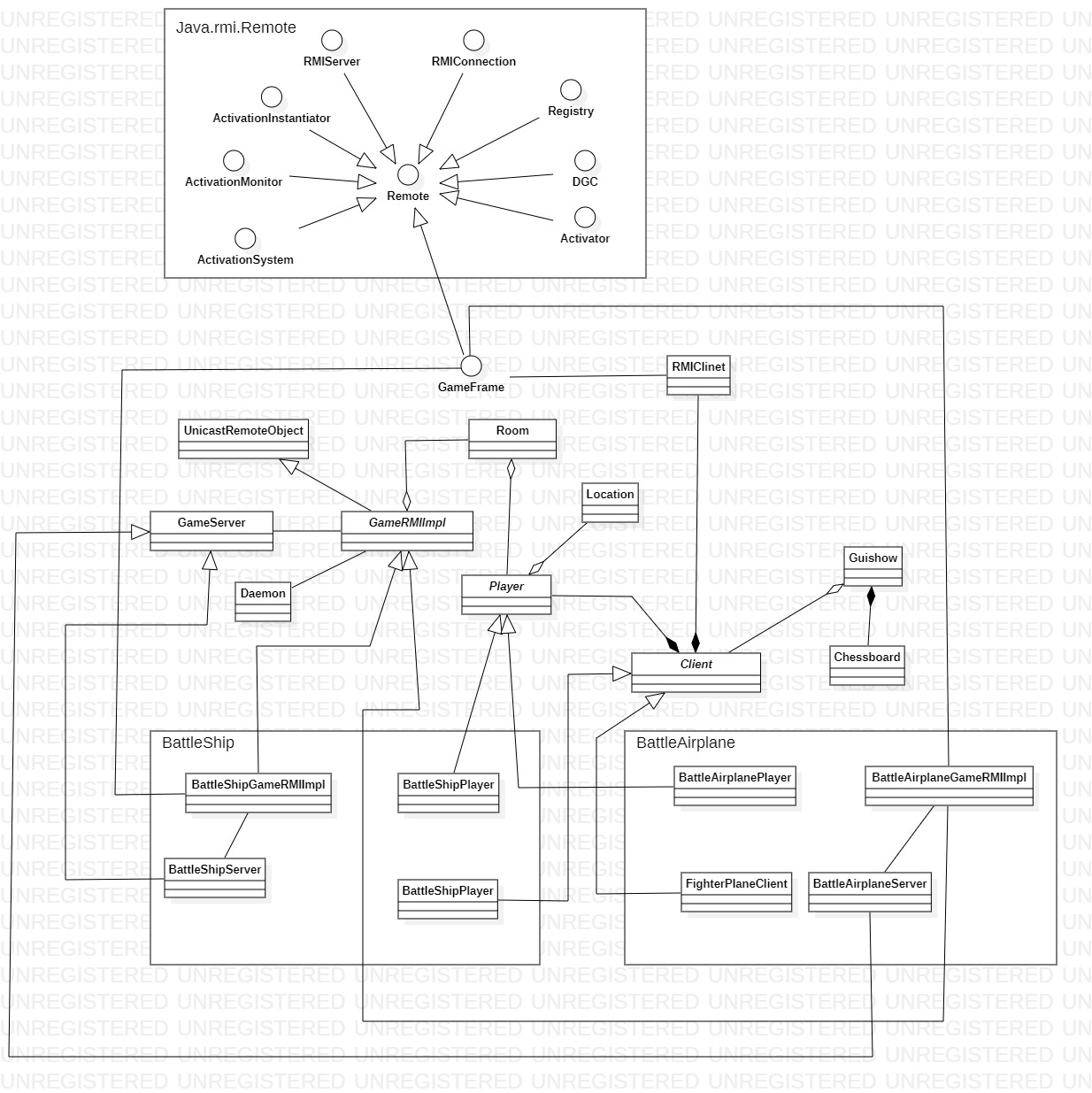
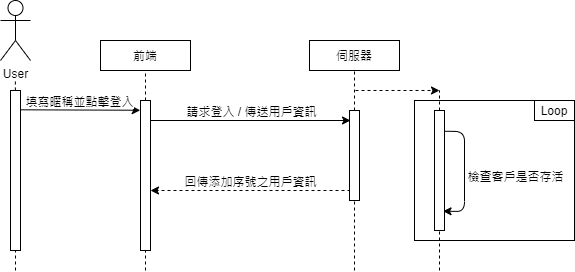
物件導向軟體工程期末報告

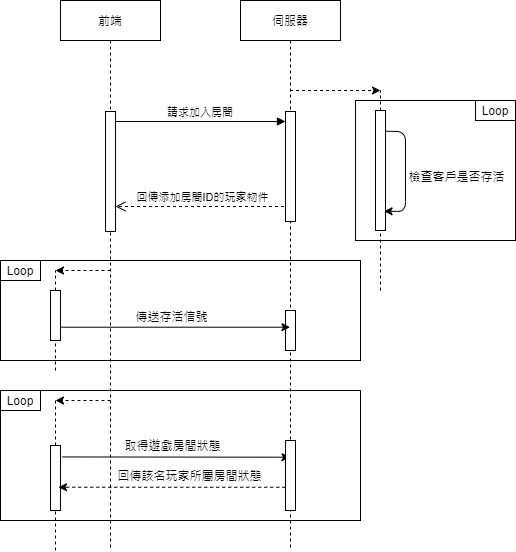
題目說明:

1. Framework : 戰棋
2. Application01 : 海戰棋
3. Application02: 空戰棋

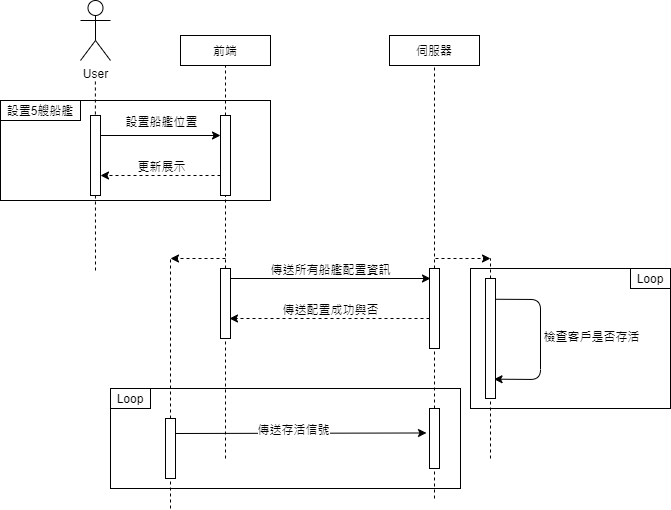
系統設計（UML 圖):  
  
🔺專案設計架構物件圖



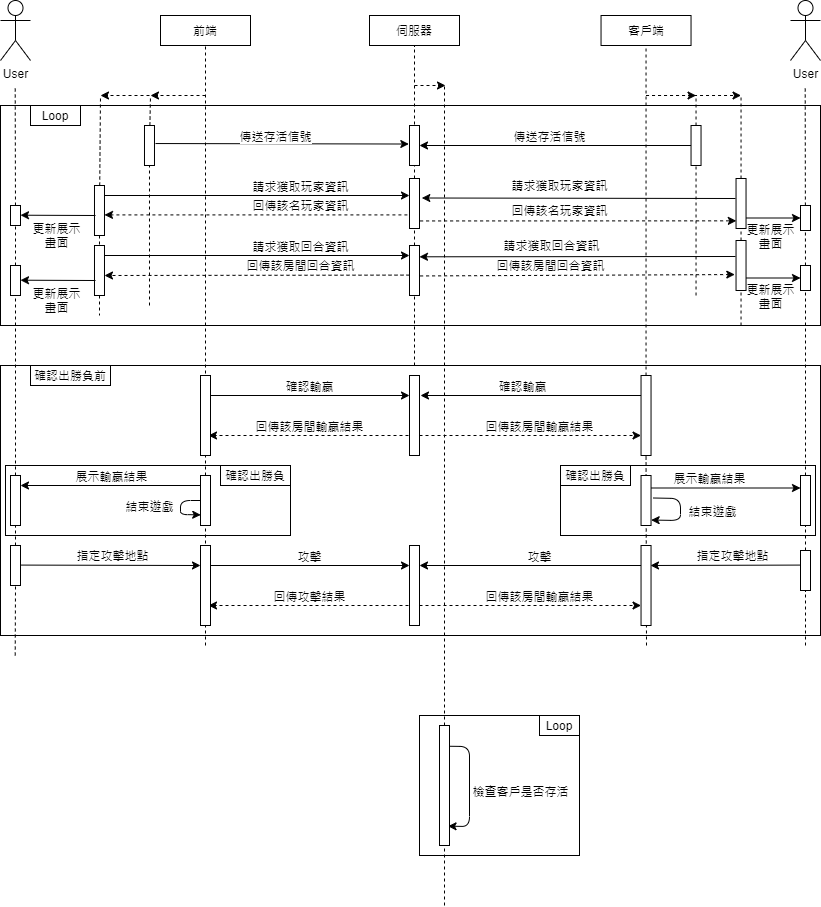
🔺登入循序圖



🔺使用者加入房間循序圖



🔺部屬陣行循序圖



🔺攻擊與勝負判斷循序圖

系統設計說明:

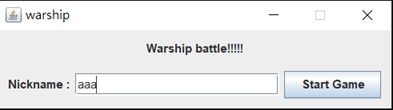
1. 框架功能與應用程式功能  
    GameFrame是為此次專案兩個應用的框架，但同時也為RMI的應用。我們兩個應用分別為海戰棋與空戰棋，其最大的不同點在於遊戲規則上的改動。
2. 如何應用設計原理與設計樣式  
    使用一般化原則，善用委託，避免程式臭味中的重複的程式碼。

在GameSever的部分使用到FACTORY METHOD。

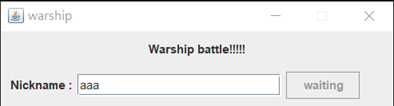
程式狀態說明:

1. 測試與執行畫面說明
2. 共同畫面

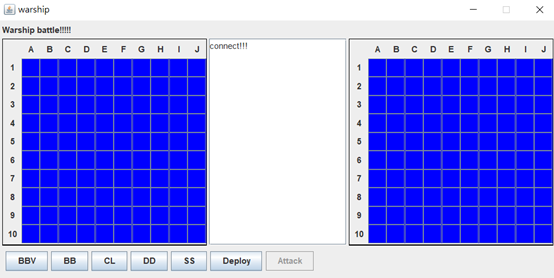
登入：



加入房間並等待對方玩家：

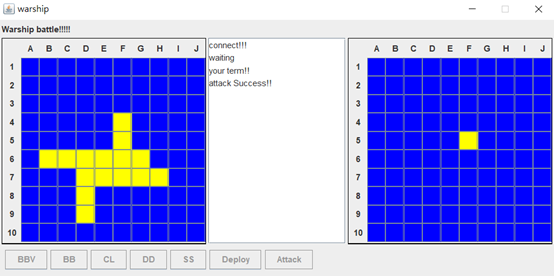


進入遊戲畫面，並部署船艦

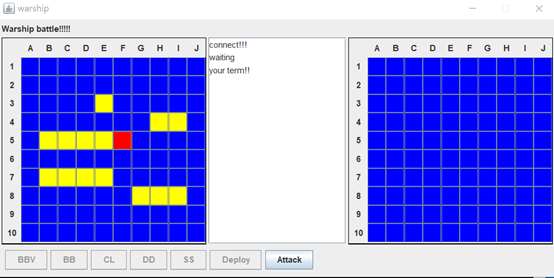


1. 海戰棋

發動攻擊

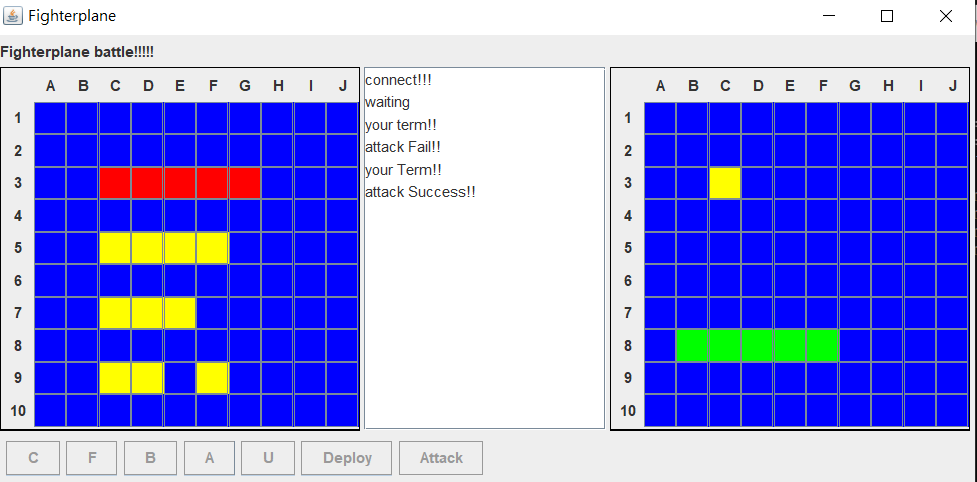


受到攻擊



1. 空戰棋

發動攻擊與受到攻擊



1. 完成度說明

在完成度方面，將兩個應用皆能夠正常執行，並且有針對使用者可能會輸入錯誤的部分進行防護，以避免使用者進行意料之外的行為導致應用崩毀。

1. 程式碼長度說明（行數）

將兩個應用的伺服器做一般化原則，雖然並沒有簡化多少的程式碼，但是讓設計變得易於理解與維護。

將兩個GameFrame的實作做一般化，因為他們只有一部份的邏輯不同，因此做了一般化的處理後，減少了不少的行數，在之後的修改也不需要特別到兩個檔案中做修改，只須找到被繼承的類別進行修改即可。因為兩種遊戲的Player中，有大部分的資料型態是相同的，因此也同樣做一般化，並在父類別中規定子類別應該履行的方法，而兩個Player的子類別則去定義各自特殊的資料型態，並根據特殊資料各自履行父類別訂定的方法。

客戶端的部分，空戰棋和海戰棋最大的不同是在於擊中判定的部分，空戰棋是只要擊中戰機的其中一個位置就算擊落，判定就會變成會處理一個陣列而不是一個點，而海戰棋相對起來就比較簡單，只需要處理一個點就可以，空戰棋跟海戰棋的客戶端大部分是相同的，所以只需要處理不同的判定的部分，所以我們就直接繼承AbstractClient，再根據判定不同的部分作出不同的方法。