

組員

系級	學號	姓名
資工二	B06902117	高聖傑
資工二	B06902131	吳冠穎
生工二	B06602035	李晴妍
圖資二	B06106043	倪爾佑

主題

網路旅館訂房系統

系統架構及功能簡介

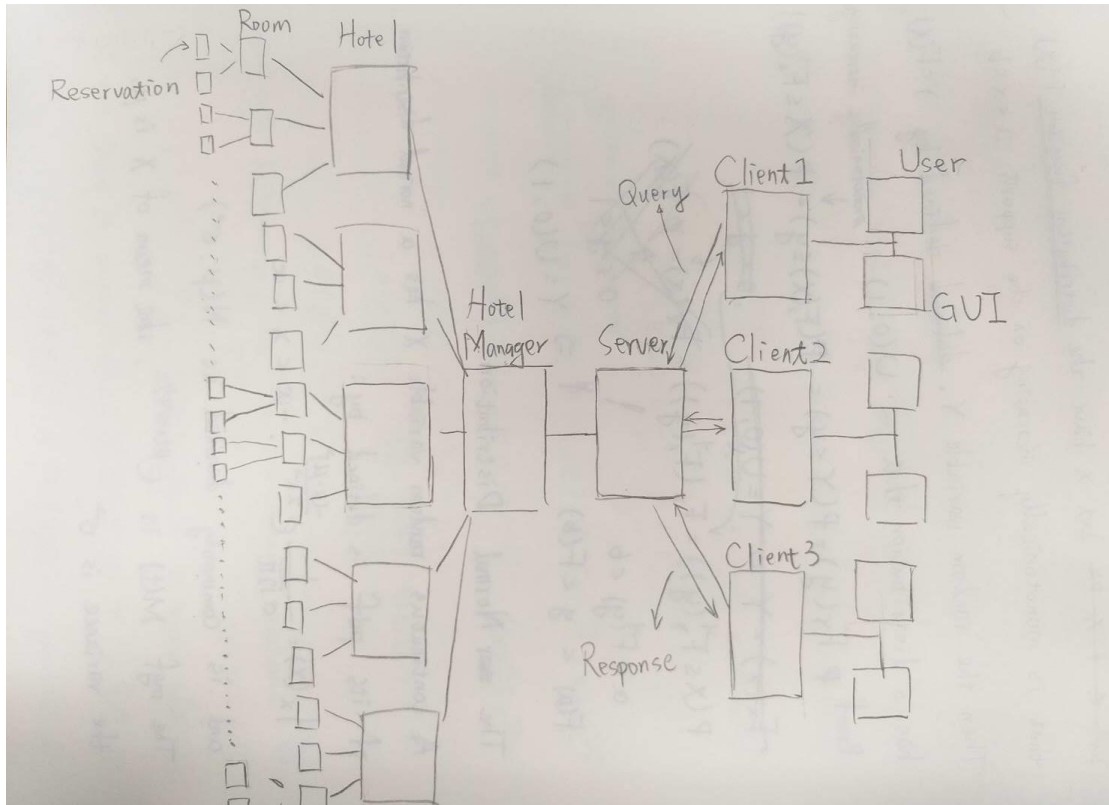
此系統為「一個飯店管理系統伺服器」連接「無數個客戶端應用程式」的旅館訂房系統(詳細架構參考下方手繪流程圖)，此系統包含非會員的訪客可自訂查詢條件，查詢相關符合條件的旅館，或是會員可使用上述功能，並且根據查詢結果預定自己想要住宿的旅館。本系統並且提供根據訂單編號來查詢訂單的細節，來進行訂單的修改、取消。除此之外，本系統也提供排序功能，使用者可根據自己的需要使用不同的條件來排序尋找結果。並可以維持在多人、每人多台機器登入時還具有高度 **interactive** 的運作。

使用到的套件

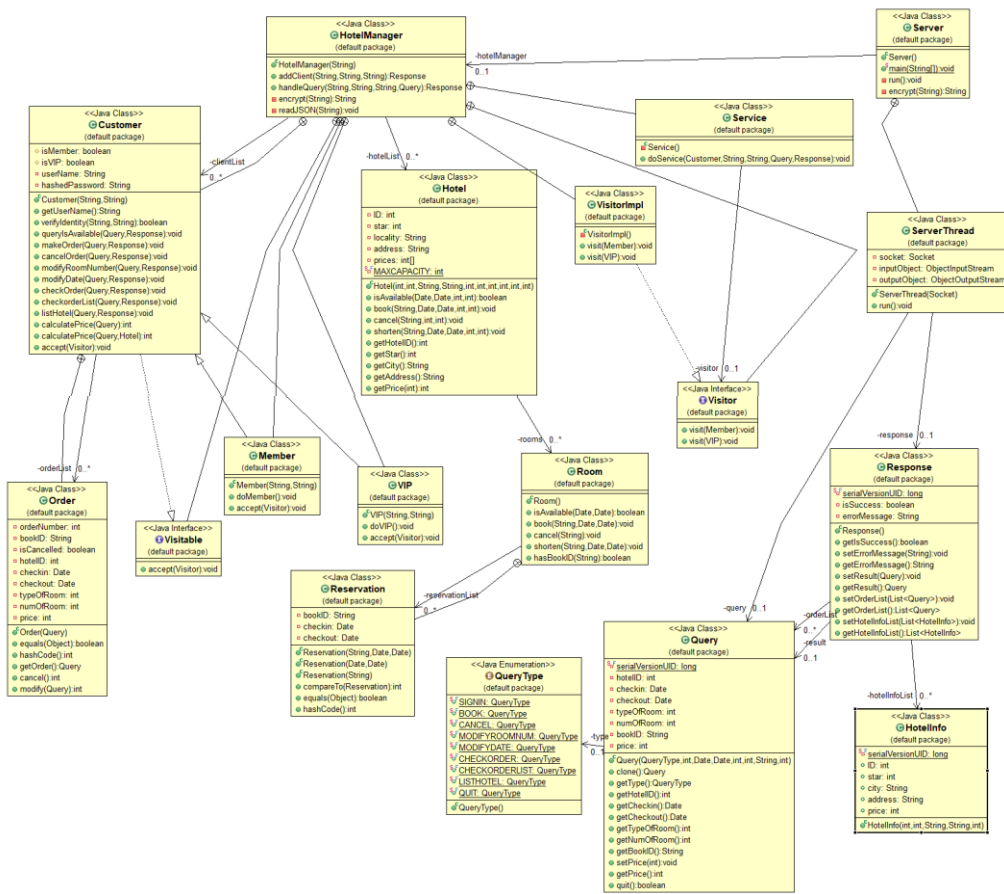
Window Builder
ObectAid
Json.jar

流程圖

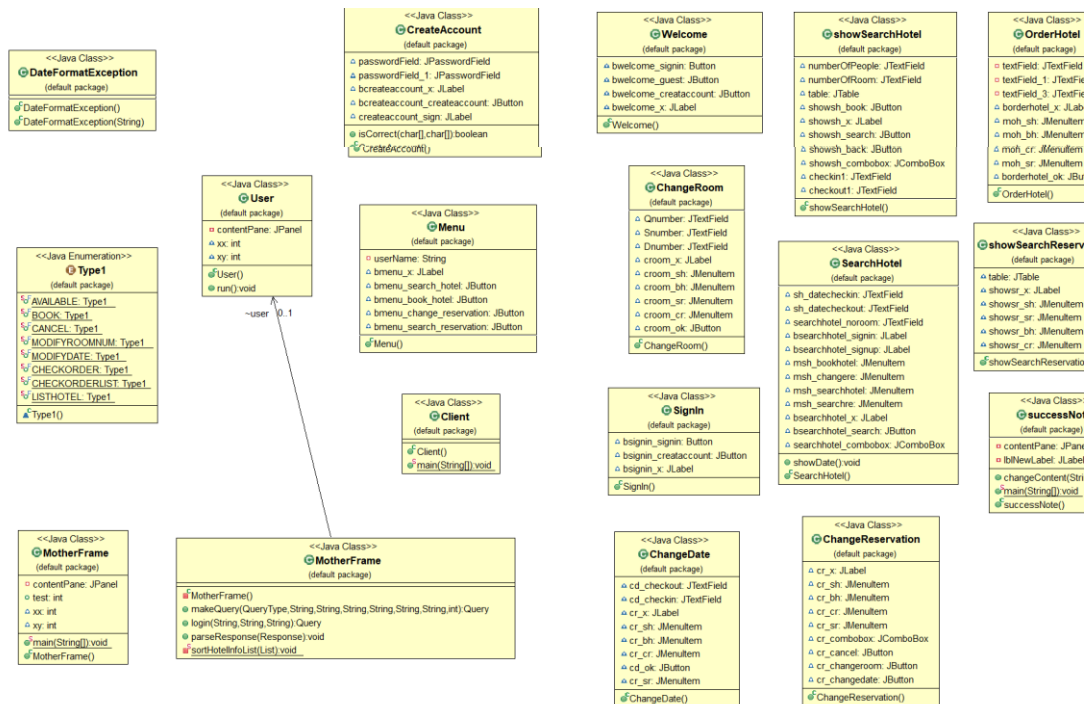
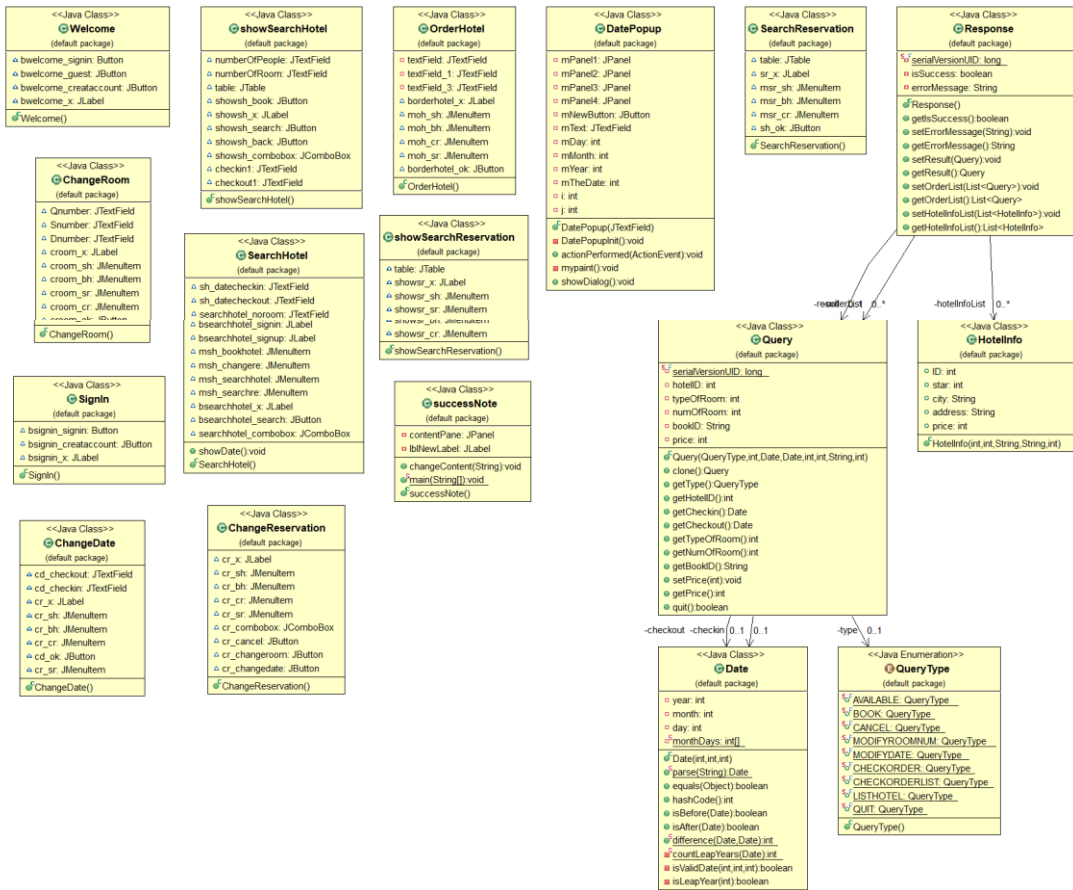
(見下頁)



Server



Client



Package 規劃(Optional)

package 的結構詳見以下連結：

<https://hackmd.io/ioYA1mWJR5SOBwkX0eg4cA?both>

訂房系統僅需擁有 **Server** 資料夾內的檔案，而客戶僅需擁有 **Client** 資料夾內的檔案。訂房系統執行 `runServer.sh` 後可以連接數個客戶執行 `runClient.sh`。

Design Pattern(Optional)

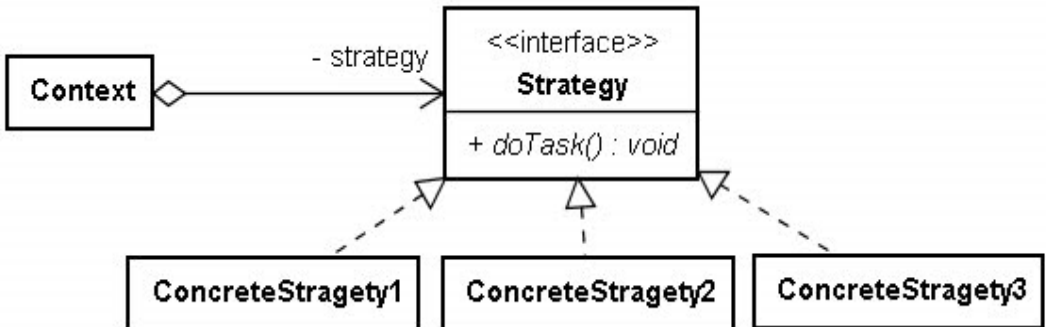
功能：多人連線系統、搶票	設計模式：Guarded Suspension
情境：多人連線系統的 Server 必須同時處理來自多個客戶端的 Query ，為了迅速接受客戶的 Query ，它要維持一個 Query queue ，客戶的 Query 會先儲存至 Query queue 中， Server 會從緩衝區中取出 Query 並執行，如 Query queue 中沒有 Query ， Server 就等待，直到被通知有新的 Query 存入 Query queue 中。 Server socket thread 和 Client socket thread ，所以必須對 Query queue 進行防護，當 Query queue 中沒有 Query 時，服務元件必須等待直到被通知有新的 Query 。強調的是對共用資料的防護，避免共用存取的競爭問題。	
功能：Interactive 服務	設計模式：Thread-Per-Message
情境： Server 對於每個資料接收之後，馬上使用一個 Thread 來處理，因此 <code>accept()</code> 方法會很快的返回， Server 對於客戶端可以有很高的回應性。可以讓 Server 需要有高回應性。服務端對每個 Query 使用一個 Thread 來處理，伺服器端接受連線後，使用一個 Thread 來處理該次連線， Server 端可馬上傾聽下一個 Client 端連線。	
功能：根據身分提供服務	設計模式：Visitor
情境：如果系統中有訪客、會員與 VIP ，假設經過設計考量，讓訪客只能查詢，會員可以進一步訂房間和取消訂單， VIP 可以有修改訂單和退款的特殊服務。	
<pre> classDiagram class Client class Visitable { <<interface>> +accept(visitor: Visitor) void } class Visitor { <<interface>> +visit(element: Element1) void +visit(element: Element2) void } class Element1 class Element2 Client --> Visitable Client --> Visitor Visitable < .. Element1 Visitable < .. Element2 note for Element1 "accept(visitor) {\n visitor.visit(this);\n}" </pre>	

功能：接收和回應網路封包	設計模式：Proxy
情境：在 Client 端送出 Query 或是 Server 端送回 Response 前，建立的是 queryProxy 或 responseProxy 實例，真正需要處理服務時，才會真正建立 Instance 並馬上交給網路服務，使用這種方式，即使用戶操作很多，也不至於拖累發送請求的速度。	

功能：部分耗時工作交給客戶端	設計模式：Command
情境：當客戶要求符合條件的旅館清單時，由於處理旅館搜尋和排序非常耗時，除了用前面使用到的 Thread-Per-Message 處理各個客戶的需求保持高回應性之外，還將旅館清單交由客戶端進行排序，但由於程式並部會預先知道客戶會使用什麼搜尋條件，所以先將排序方法抽象化出來，實際要排序的時候，才根據條件時做排序方法。	

功能：分離 Server 端和 Client 端實作	設計模式：Facade
情境：在網路的 Server 端和 Client 端分別釐出一個入口（Facade）介面，讓對程式庫的依賴實現在對介面的實作上。GUI 不需要再知曉 HotelManager 各種 API 的存在，因而不會對程式庫產生耦合，從另一個角度來想，Server 由的訂房系統的開發人員所撰寫，提供給另一個撰寫 GUI 的開發人員所使用，則後者並不用一定得知曉訂房系統如何使用，有利於分工合作，將來就算開發訂房系統的開發人員改寫或重新實作了另一個 Service，撰寫 Application 的開發人員也無需修改寫它的程式。	

功能：管理 Reservation	設計模式：Chain of Responsibility
情境：多個物件都有機會處理 Query，除了可以自由組合處理 Query 的物件之外，也可以避免 Query 的發送者與接收者之間的耦合關係，將這些物件組合為一個鏈，並沿著這個鏈傳遞該 Query，每個都可以物件處理它，決定是否傳遞給下一個處理物件。在處理訂房系統物件之間的職責時，從細粒度至粗粒度的方式來組織，從特殊到抽象化，讓 HotelManager 將房間視為旅館的特殊化，每個時間段的 Reservation 又為房間的特殊化。	

功能：分離 Server 的服務和計算	設計模式：Strategy
情境：著重在服務細節或演算法流程的封裝，將服務與演算法封裝為一個個的 Strategy 物件，讓使用服務或演算法的客戶端可以依需求抽換服務或演算法，而不用關心服務或演算法的實作方式。訂房網站實際連線之後 HotelManager 所提供的服務，Server 並不用知曉。	
 <pre> classDiagram class Context class Strategy { <<interface>> +doTask() void } class ConcreteStrategy1 class ConcreteStrategy2 class ConcreteStrategy3 Context o--> Strategy : - strategy Strategy < .. ConcreteStrategy1 Strategy < .. ConcreteStrategy2 Strategy < .. ConcreteStrategy3 </pre> <p>The diagram illustrates the Strategy Design Pattern. It features a Context class and a Strategy interface. The Context class has a composition relationship with the Strategy interface, indicated by a solid line with an open diamond at the Context end and an arrow pointing to the Strategy interface, labeled with the attribute <code>- strategy</code>. The Strategy interface is marked with <code><<interface>></code> and defines a method <code>+ doTask() : void</code>. Below the interface, three concrete classes are shown: ConcreteStrategy1, ConcreteStrategy2, and ConcreteStrategy3. Each of these concrete classes has a dashed line with an open arrow pointing to the Strategy interface, signifying that they implement the Strategy interface.</p>	

功能：User 協調 Client 和 GUI 工作	設計模式：Mediator
<p>情境：在發送 Query 的同時，Client 網路端和 GUI 操作彼此之間可能知道彼此的存在，並相互依賴，這就使得物件之間的耦合性相對的提高，這會使得系統的重用性降低。用一個 User 來封裝物件彼此之間的交互，物件之間並不用互相知道另一方，這可以降低物件之間的耦合性，如果要改變物件之間的交互行為，也只需要對 User 加以修改即可。</p>	

功能：穩定的訂房系統 Server	設計模式：Two-phase Termination
<p>情境：有一個 Thread 正在週期性的運作，在「運作階段」Client 送出了停止連線的 Query(例如關閉 GUI)，這時候 Thread 不會慌張的馬上終止目前的工作，而是先完成這一次週期的工作，然後進入「善後階段」處理 Serialized 的物件和網路串流，即中止「運作階段」，並完成「善後階段」，完整的完成 Thread 的工作。保持 Server 可以在對某個 Client 的連續或服務無預警的 crash，不會影響整體運作，可以繼續接受其他 Client 的連線和提供服務。</p>	

功能：保證同步時的資料一致性	設計模式：Read-Write-Lock
<p>情境：如果有一個房間有可能同時會有許多 Client 的服務想要對它進行讀取與寫入，則必須注意資料的同步問題，像是兩個寫入者進行寫入時，後一個寫入者的資料會有可能將次一個寫入者的資料覆蓋掉；而有時又希望讀者看到的是最新的資料，如果在讀取的時候，有寫入者想要對資料進行寫入，則最好等待讀取者讀取完畢，相反的如果在寫入時有客戶想要讀取資料，則最好等待，以確保讀出來的資料是最新的資料。另外，於前面的 Chain of Responsibility 有提的 Reservation 的概念，若讓 Service 在讀取或寫入 Room 時對 Reservation 進行鎖定，我們還可以進一步只鎖定 Room 的某個時間段，而不是將整個 Room 的讀寫都進行鎖定。</p>	

功能：共用資料的同時進行安全分工	設計模式：Thread-safe
<p>情境：為了讓客戶能在時間上得到公平的服務，在 HotelManager 使用多個 Thread 進行分時工作，但要編寫一個保持多 Thread-safe 的，使用的共用資源，必須小心的對共用資源進行同步，同步會帶來一定的效能延遲，而另一方面，在處理同步的時候，要注意物件的鎖定與釋放，避免產生死結。</p>	

功能：客戶端低運算成本待命	設計模式：Producer Consumer
<p>情境：每個連線後的 user，都會建立一個 Query 物件封裝客戶端的相關資訊，並且加入一個 Query queue 中集中管理。現在 Client socket 可能準備好對 Query queue 中 Query 做移除，並且希望 Query 加入時都可以收到通知，以便作一些處理。Query queue 可被任何 user 加入 Query，當 Query queue 的狀態發生變化時，它必須通知 Client socket，Client socket 會檢視 Query queue 主題的狀態變化，作出對應的動作或待會再回來處理。</p>	

資料庫設計(Optional)

無

APP/GUI 介面截圖(Optional)

分工表

高聖傑 50 % - 旅館訂房管理系統 & Design pattern & 網路連線

李晴妍 45 % - 客戶的 GUI

倪爾佑 5 % - json 轉檔 & 程式註解

吳冠穎 0 % - 做自選題

心得(optional)

OOP 的期末 Project，絕對是課程本身最重要一環，不僅讓我們能親手實作這一學期來教授所講過的所有語法、觀念和物件導向這個程式設計的典型方針，其中自選題更可以看出教授和助教的用心，題目和所要求的一些 Function 都是為了讓我們能熟習物件導向這個抽象卻又實用至極的程式編寫方向。最後我們也在完成這個 Project 的同時對於如何多人一起完成一隻專案、撰寫一隻程式有了初步的認識，希望未來有機會能運用到這學期所學到的任何知識~

其他

考慮以下兩點：

1. 網路封包可能被攔截
2. 訂房網站系統可能有內鬼 [註一]

我們針對訂房系統中的使用者密碼和訂單進行加密，即使訊息被攔截，也客戶的機密資訊流出，並且在訂房系統內部也做特殊處理，即使是操作訂房網站系統的內部人員，甚至任何可以看到 **private** 變數的開發人員，也無法看到真實的內容(Demo 時為了展示方便，部分操作是在未加密狀況執行的)。

我們使用的加密方式為 MD5，是一種密碼雜湊函式(cryptographic hash function)，這種密碼雜湊函式有四個主要的特性：(1) 對於任何一個給定的訊息，它都很容易就能運算出雜湊數值；(2) 難以由一個已知的雜湊數值，去推算出原始的訊息；(3) 在不更動雜湊數值的前提下，修改訊息內容是不可行的；(4) 對於兩個不同的訊息，不會有相同的雜湊數值。在資訊安全中，有許多重要的應用，都使用了密碼雜湊函式來實作，如數位簽章，訊息鑑別碼等等。

[註一] 2019 年 03 月 Facebook 被踢爆 2,000 名員工可瀏覽 6 億用戶密碼

<https://www.bnext.com.tw/article/52675/facebook-millions-users-passwords-stored-in-plain-texts>

參考資料

<https://openhome.cc/Gossip/DesignPattern/>

<https://stackoverflow.com/>

<https://developer.android.com/reference/org/json/package-summary>

<https://docs.oracle.com/javaee/7/api/javax/json/stream/JsonParser.html>