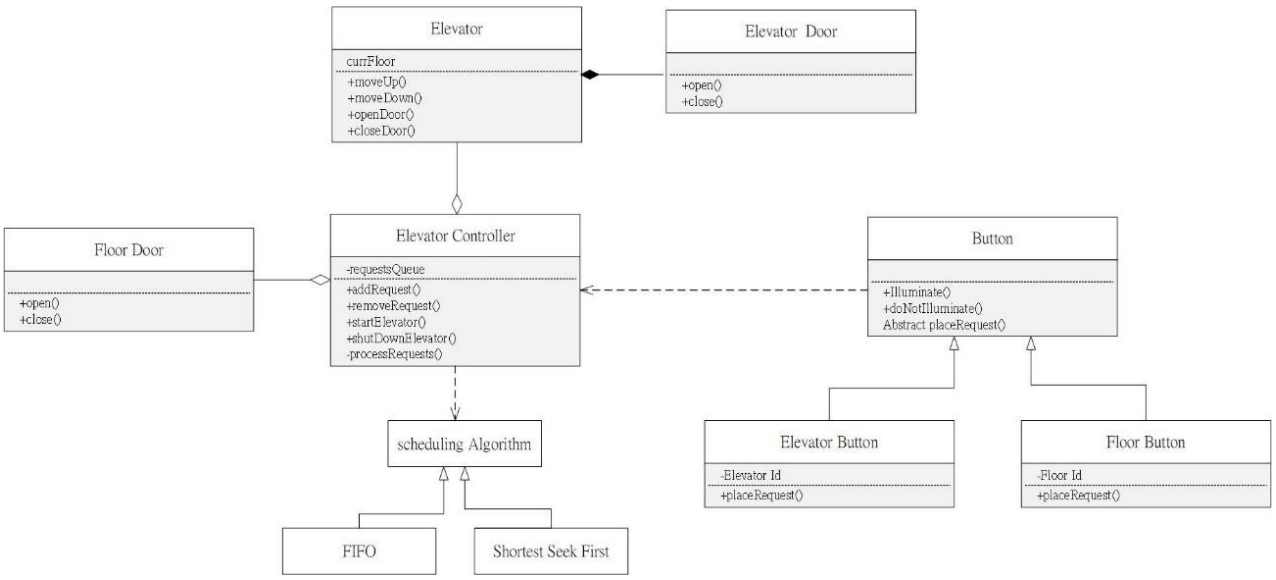
CSIE5142/CSIE4302 Software Engineering

Homework # 3 105599004 林慧琪

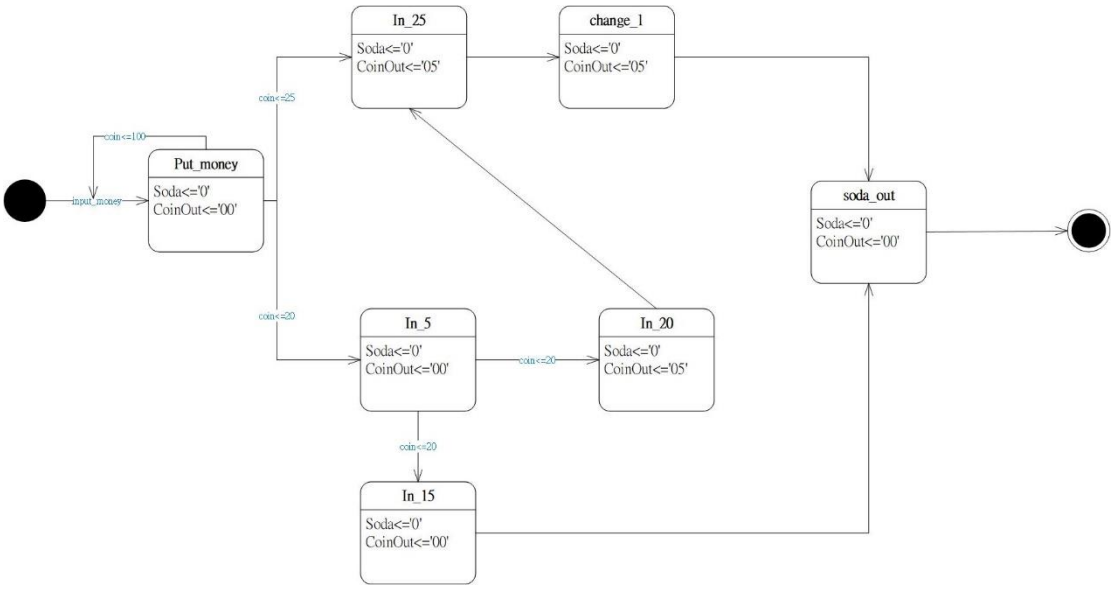
Due on 12/13/2017

1. (20%) You have been appointed to develop a system that controls the elevators inthe NTUT Technology Building. Basically, the system requires to move ***n*** elevators between ***m*** floors according to the following constraints:
   - Each elevator has a set of m elevator buttons, one for each floor. These buttons illuminate when pressed and cause the elevator to visit the corresponding floor. The illumination is canceled when the corresponding floor is visited by the elevator.
   - Each floor, except the first floor and the top floor, has two floor buttons, one to request an up-elevator and one to request a down-elevator. The first floor has only one elevator button to request an up-elevator and the top floor has one elevator button to request a down-elevator. These elevator buttons illuminate when pressed. The illumination is canceled when an elevator visits the floor and then moves in the desired direction.
   - When an elevator has no requests, it remains at its current floor with its door closed.

   Please identify the possible object classes of the elevator system and draw the domain model of the system.



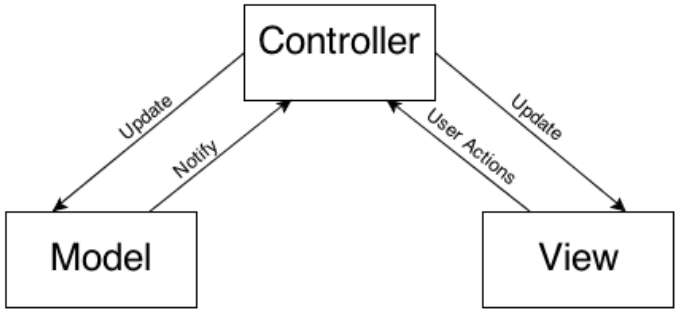
2. (20%) Imagine a vending machine that accepts combinations of $5, $10, and $50coins to get a bottle of beverage. The cost of a bottle of beverage is $20 and the sale button of the beverage is enabled when the deposited money is equal to or greater than $20. Once the sale buttons of beverages are enabled, a customer then can press the button that goes with the beverage of choice. The selected drink will appear in the opening at the bottom of the vending machine and the changes (if any) will be dispensed automatically. Draw a state machine (or statechart) for this vending machine. Make sure that you specify the states and transitions of the state machine. (Note: make sure your state diagram is correct and complete and specify your assumption if necessary.)

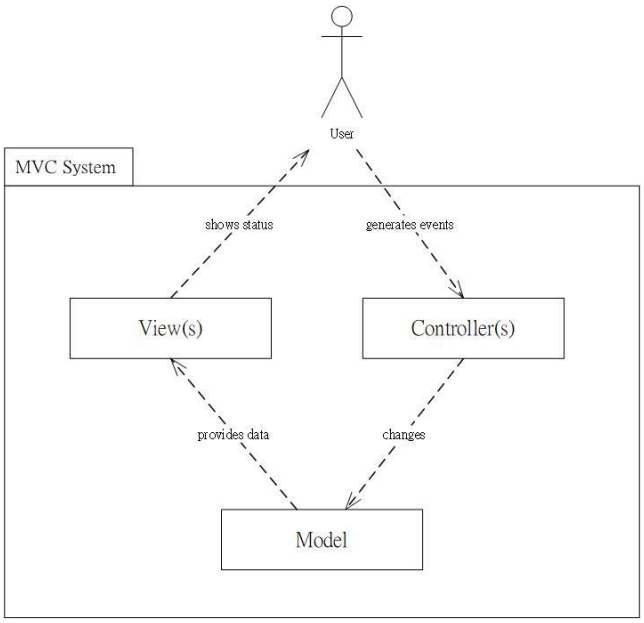


3. (20%) Draw and describe your architecture design for the PASS class project using the Model-View-Controller (MVC) architecture style. Please clearly identify the model, view, and controller in your architecture design of PASS.

MVC 模式將互動系統切割成三個元件：

a. processing（Model）、output（View）、input（Controller）。Model 封裝核心資料與功能，並且獨立於特定的顯示方式與輸入行為。
b. View 負責顯示 Model 的資料，一個 Model 可以包含多個 View。每一 個 View 有一個 Controller 用來接收輸入（例如滑鼠或鍵盤事件），並將其轉換成對 Model 或 View 的呼叫。
c. 區分 Model 與 View/Controller 讓我們可以使用多個 View 來顯示相同 Model。如果使用者透過某個 View 的 Controller 改變 Model 的資料，所有其他相依於該 Model 的 View 將會收到 Model 關於資料異動的通知，如此一來這些 View 便可從 Model 接收異動的資料並用來更新自己所顯示的資訊。





4. (20%) Draw and describe the Observer pattern and give an example of applying the pattern.

設計一個足球遊戲，其建構起始如下：

(1) 一開始進入遊戲時之選項：

- 開始遊戲
- 選擇兩個團隊
- 在球隊中添加或移除球員
- 選擇一個遊戲場
- 開始遊戲
- 在系統中可能有多個 PlayGrounds，一些團隊等等。

(2) 選擇開始進行遊戲時之角色設定：

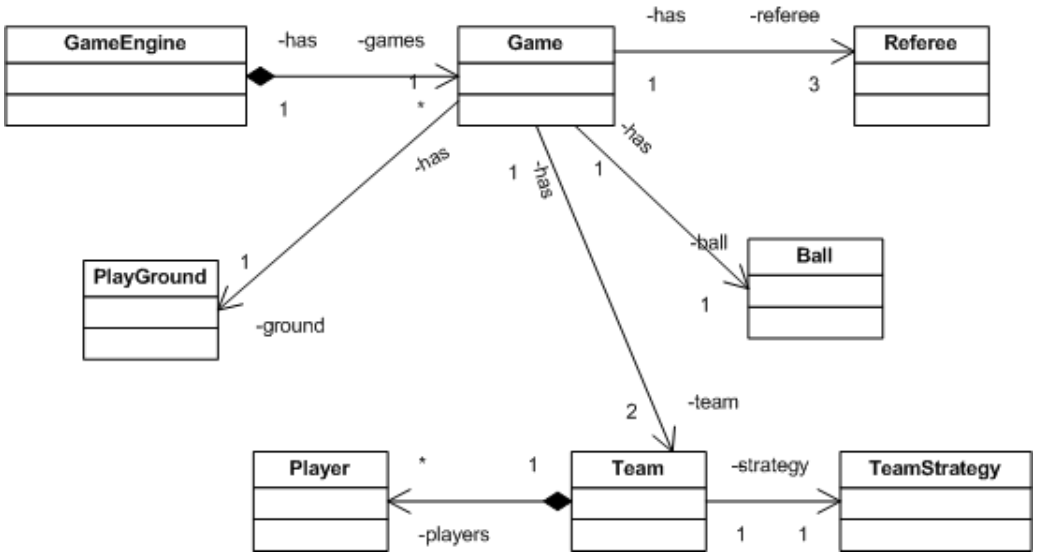
- 踢足球的球員
- 與其中的各種球員的團隊
- 由各種球員處理的球。

- PlayGround 比賽發生的地方。
- 裁判在地面上控制比賽。

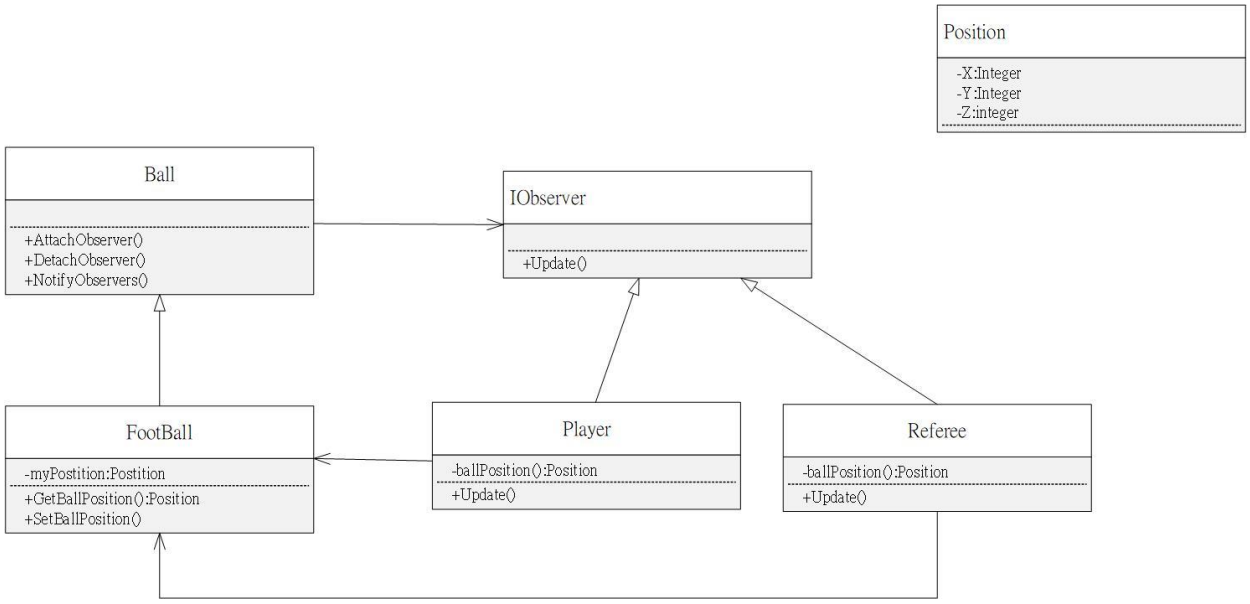
(3) 在遊戲中加入一些邏輯斷條件

- 定義一場足球比賽，這構成了球隊，球，裁判，操場等。
- GameEngine 一次模擬許多遊戲。
- TeamStrategy 在比賽中決定一個球隊的戰略

下圖為此遊戲 abstract view ，每個物件之間擁透過連接符號描述“has”關係和多樣性。即，一個 GameEngine 有（可以模擬）遊戲。一場比賽有三名裁判，一名球員，兩名球隊和一名球員。一個團隊可以有多個參與者，一次可以有一個裁判。



建立此足球遊戲，當使用 SetBallPosition 球的函數來設置新的位置時，它會調用 Notify Ball 類中定義的函數。Notify 函數迭代列表中的所有觀察者，並調用其中 Update 的每一個函數。當 Update 函數被調用時，觀察者將通過調用 GetBallPosition Foot ball 類中的函數來獲得球的新的狀態位置。



5. (20%) The SOLID stands for five basic principles of object-oriented programming and design. Please describe the main concept of each design principle in SOLID below.

Initial	Stands for	Name	Concept
S	**SRP**	Single-responsiblity principle	一個 class 內應該只有一個改變的理由，意思是一個 class 應該只需要專注做一件事情，這個理由是當 class 內需要做兩件以上的工作時，一旦程式需要改變，因為他的耦合度比一件工作還要高，造成變更上的困難，所以盡量避免。
O	**OCP**	Open-closed principle	當你新增功能的時候，需能在不更改原本程式碼的情況下進行擴增，如此一來，可以增加維護性，降低原有的功能發生錯誤的機率，重用性的機率也會隨之提高。
L	**LSP**	Liskov substitution principle	若是使用繼承，子類別實作的行為必須要與父類別或是介面所定義的行為一致，並且子類別要能夠完全取代掉父類別，若子類別沒有遵守 LSP 的情況下，使用者呼叫某個功能時可能會出現與預期的功能不相符

			合的問題。
I	**ISP**	Interface segregation principle	在設計介面時，不應該放入不相關的 method，例如在一個叫 animal 的 abstract class 內含有 run()和 fly()的 method，由狗和鳥繼承，但是狗會跑卻不會飛，這樣 fly()的 method 卻能被狗使用不就很奇怪嗎? 所以這個原則主要用途是在降低耦合度，減少不當的設計。
D	**DIP**	Dependency Inversion Principle	高階模組不要依賴低階模組，兩者都必須要依賴抽象模組；抽象不能依賴細節，細節必須依賴抽象，簡單來說，假設我有多個廠牌亮度不相同的燈泡，但是只要符合燈座的規格，都可以安裝上去，以這個範例來說，燈座的規格就是抽象模組。