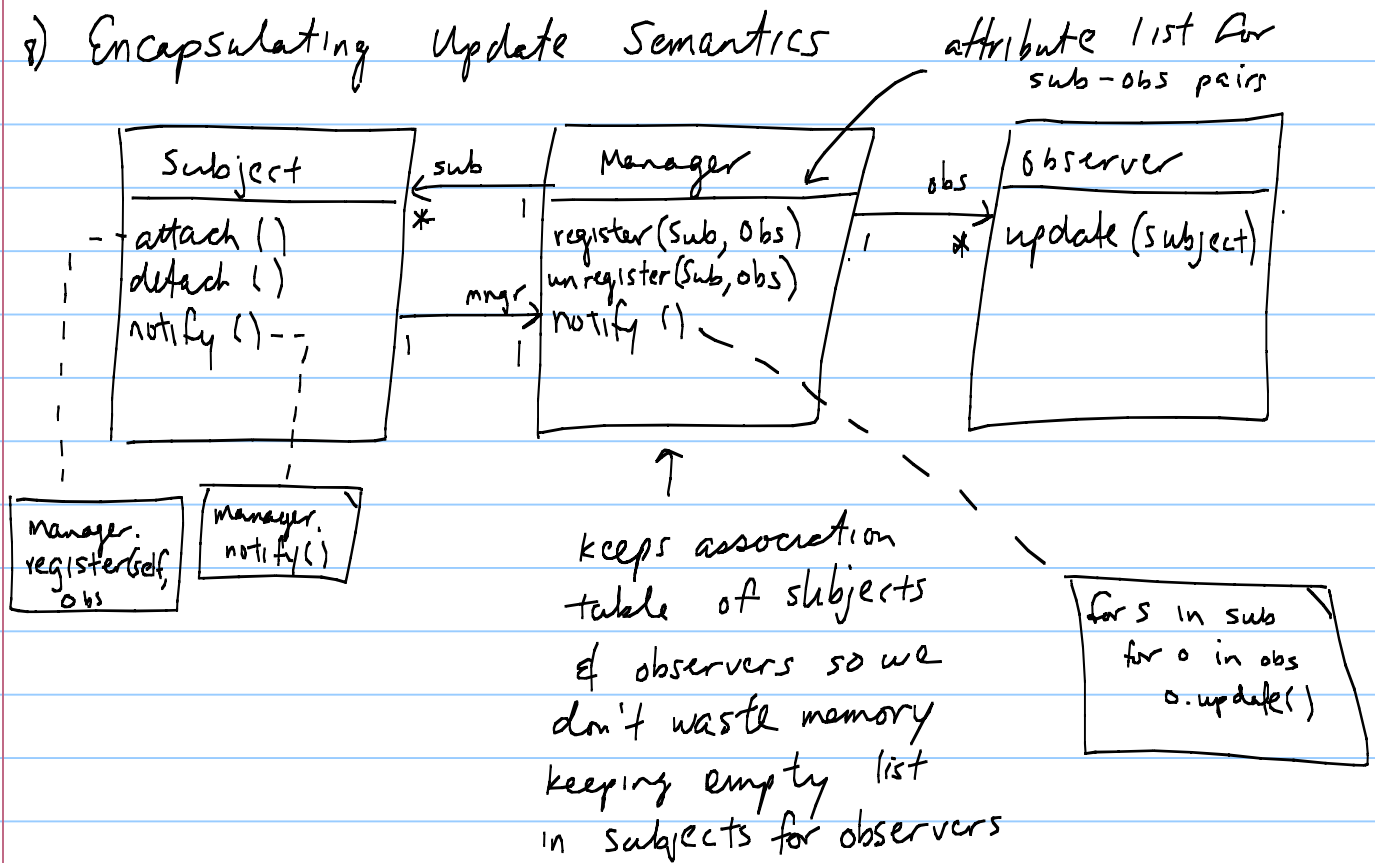
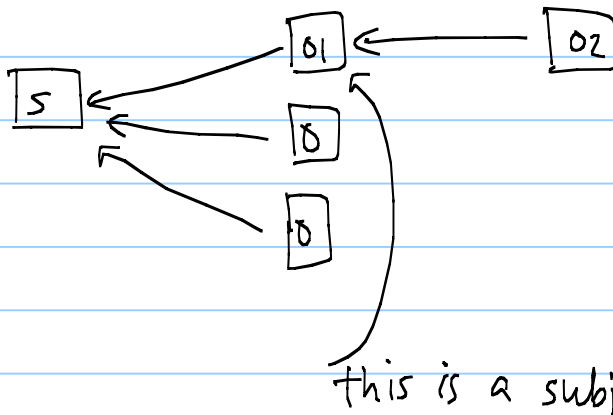


## 8) Encapsulating Update Semantics



## a) Both a subject &amp; an observer



Fine, when 01 does getstate, it will do setstate, so there will be a notify to 02 so, getstate & setstate again. What if s observed 02? then there would be a cycle of notifies & get/set states, make a special Manager class to handle this.

## State Space & Behaviour ("GOOD DESIGN")

Class: uniform abstraction of relevant properties of an object

↑  
applicable to all instances

↑  
dependant on your project  
(if software deals w/ shoes, then shoe size is very relevant)

↑  
state space ~ attributes  
behaviour ~ methods

State Space of Class C is the set of all permitted states of any object of class C

ie: if we have a class Square w/ attributes x, y and size.

Square
$x \in \{0, 1, 2, 3, \dots, 7\}$
$y \in \{0, 1, 2, \dots, 5\}$
$size \in [1, 3]$

So, to depict a state, we must give a tuple  $(x, y, size)$ .

$(2, 2, 2.2)$  is a valid state

$(2, 7, 1)$  is not since y

is out of range. A method may also violate a state space. So, BEFORE & AFTER a method, state space must be valid.

$$\begin{aligned} \text{So, } \text{StateSpace}(\text{Square}) &= \{ (x, y, size) \mid x \in \{0, 1, \dots, 7\}, y \in \{0, 1, \dots, 5\} \\ &\quad size \in [1, 3] \} \\ &= \{0, 1, \dots, 7\} \times \{0, 1, \dots, 5\} \times [1, 3] \end{aligned}$$

since  $A \times B$  is all combinations.

We can also draw a state space on a graph.