LECTURE 20

FRIDAY NOVEMBER 15

Preconditions $\rightarrow$ subclass

P_1 requires less than P_2

$$P_2 \Rightarrow P_1$$

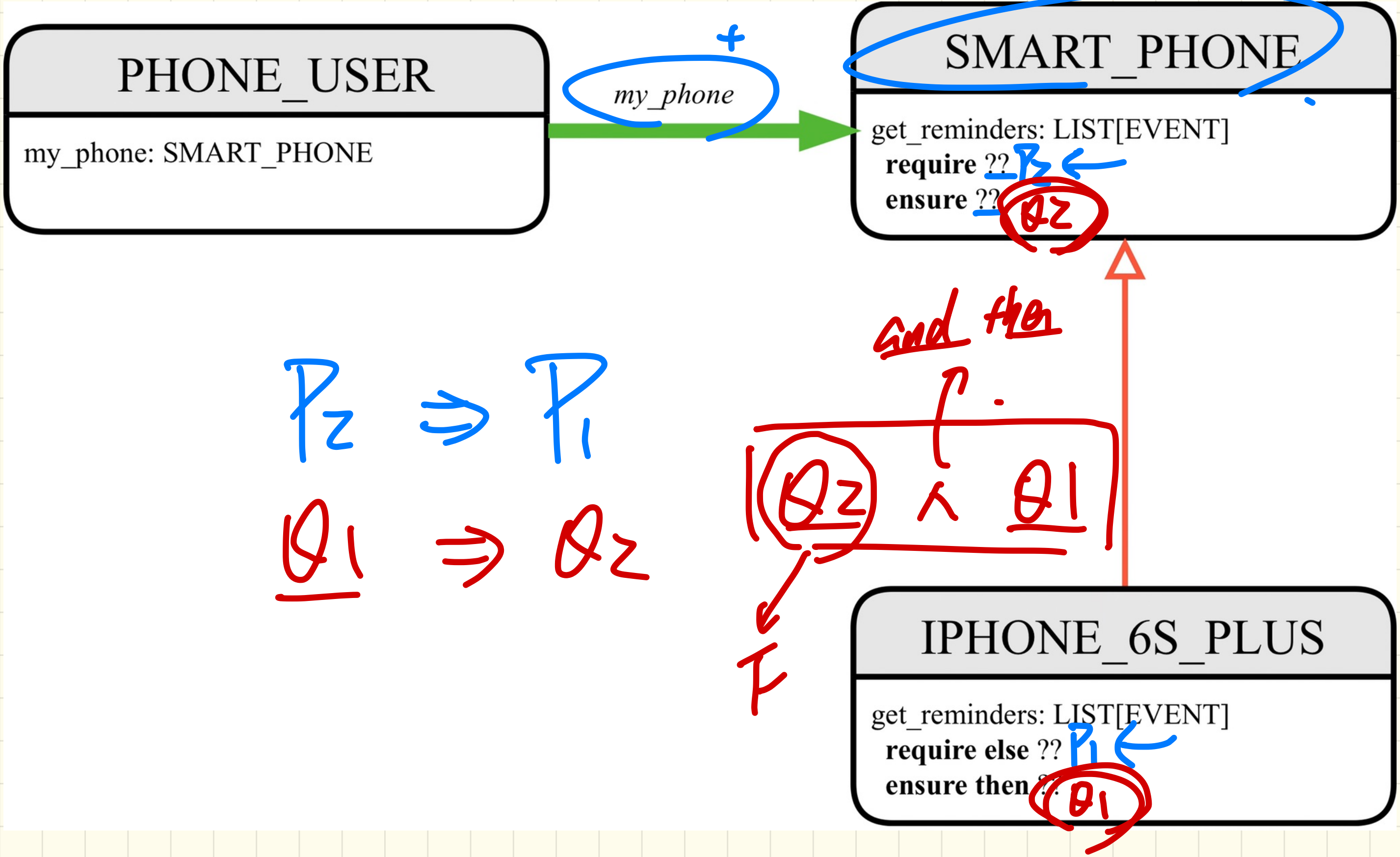
$x > 0 \qquad x \geq 0 \qquad$ e.g. $x = 9$

Postconditions

Q_1 ensures more than Q_2

subclass. $\quad Q_1 \Rightarrow Q_2$

Subcontracting: Architectural View



$$P_2 \Rightarrow P_1$$

$$\underline{Q_1} \Rightarrow Q_2$$

$$\boxed{Q_2 \wedge \underline{Q_1}}$$

Subcontracting: Example (1)

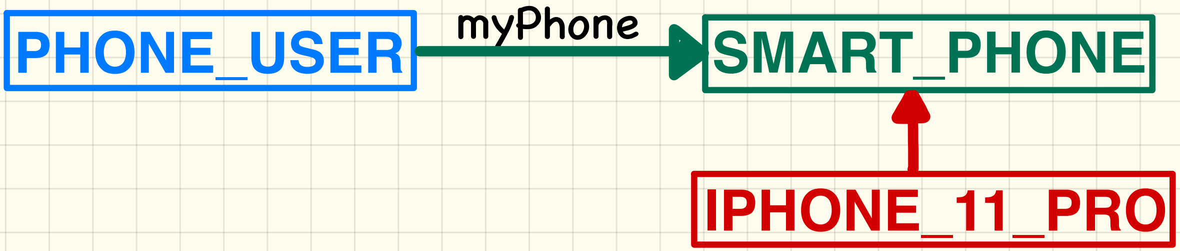
```
class SMART_PHONE
  get_reminders: LIST[EVENT]
    require
      α: battery_level ≥ 0.1  -- 10%
    ensure
      β: ∀e : Result | e happens today
end
```

level $\geq$ 10% $\boxed{13\%}$

level $\geq$ 10% $\Rightarrow$ lev $\geq$ 15%

```
class IPHONE_11_PRO
inherit SMART_PHONE redefine get_reminders end
  get_reminders: LIST[EVENT]
    require else
      γ: battery_level ≥ 0.15  -- 15%
    ensure then
      δ: ∀e : Result | e happens today or tomorrow
end
```

level $\geq$ 15%



```
class SMART_PHONE
  get_reminders: LIST[EVENT]
    require
      α: battery_level ≥ 0.1 -- 10%
    ensure
      β: ∀e : Result | e happens today
end
```

weaker

T

13%.

substitutability.

```
class IPHONE_11_PRO
inherit SMART_PHONE redefine get_reminders end
  get_reminders: LIST[EVENT]
    require else
      γ: battery_level ≥ 0.15 -- 15%
    ensure then
      δ: ∀e : Result | e happens today or tomorrow
end
```

stronger ← not appropriate

F

Runtime.

T

level ≥ 10% or else level ≥ 15%

T

```
class SMART_PHONE
  get_reminders: LIST[EVENT]
    require
      α: battery_level ≥ 0.1 -- 10%
    ensure
      β: ∀e:Result | e happens today
end
```

level ≥ 10% ⇒ level ≥ 5%

```
class IPHONE_11_PRO
inherit SMART_PHONE redefine get_reminders end
  get_reminders: LIST[EVENT]
    require else
      γ: battery_level ≥ ~~0.15~~ -- 15%
    ensure then
      δ: ∀e:Result | e happens today or tomorrow
end
```

5%

p: SMART_PHONE

create {IP_11_PRO} p.make

6% → p.get_reminders.

6%

level ≥ 10%

or else 6%

level ≥ 5%

F

T

f

require

p1

p2

$p1 \wedge p2$

ensure

q1

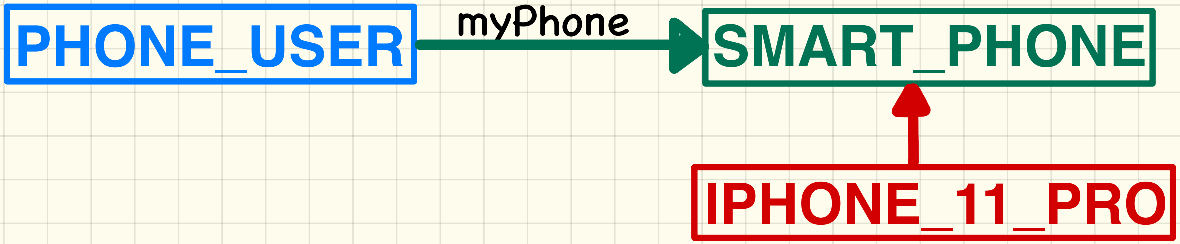
q2

$q1 \wedge q2$

Subcontracting: Example (2)

```
class SMART_PHONE
  get_reminders: LIST[EVENT]
    require
      α: battery_level ≥ 0.1 -- 10%
    ensure
      β: ∀e : Result | e happens today
end
```

```
class IPHONE_11_PRO
inherit SMART_PHONE redefine get_reminders end
  get_reminders: LIST[EVENT]
    require else
      γ: battery_level ≥ 0.15 -- 15%
    ensure then
      δ: ∀e : Result | e happens today or tomorrow
end
```



Contract Re-Declaration:

Missing Pre-Condition in Ancestor

true or else x>0

```
class FOO
  f   require true
    do ...
    end
end
```

```
class BAR
inherit FOO redefine f end
  f require else ~~new_pre~~ x>0
    do ...
    end
end
```

if in the parent class there's no precondition ⇒ no precond. in all descendants.

Runtime

① true or else new_pre

② false or else new_pre

weaken f at the Foo level.

Contract Re-Declaration:

Missing Post-Condition in Ancestor

```
class FOO
  f
    do ...
    end
end
```

```
class BAR
inherit FOO redefine f end
  f
    do ...
    ensure then new_post
    end
end
```

if no postcondition in parent class ⇒ expected that som postcondition will be added in descendants.

ensure

SP get_vpn_ta: LIST

IP get_vpn_i GROUP

today

F

never succeed at runtime

① false and then new_post

② true and then new_post

Contract Re-Declaration:

Missing Pre-Condition in Descendant

out

```
class FOO
  f require
      ~~original_pre~~ level ≥ 10%
    do ...
    end
end
```

```
class BAR
inherit FOO redefine f end
  f ▭ .
 →  do ...
    end
end
```

① original_pre or else true

T ≥ 10%

≡ true

↳ not appropriate ∵ no constraint on f.

Contract Re-Declaration:

Missing Post-Condition in Descendant

```
class FOO
  f
    do ...
    ensure
      original_post
    end
end
```

```
class BAR
inherit FOO redefine f end
  f
    do ...
    end
end
```

as if: ~~ensure~~ true

original_post and then false F.

↳ not appropriate

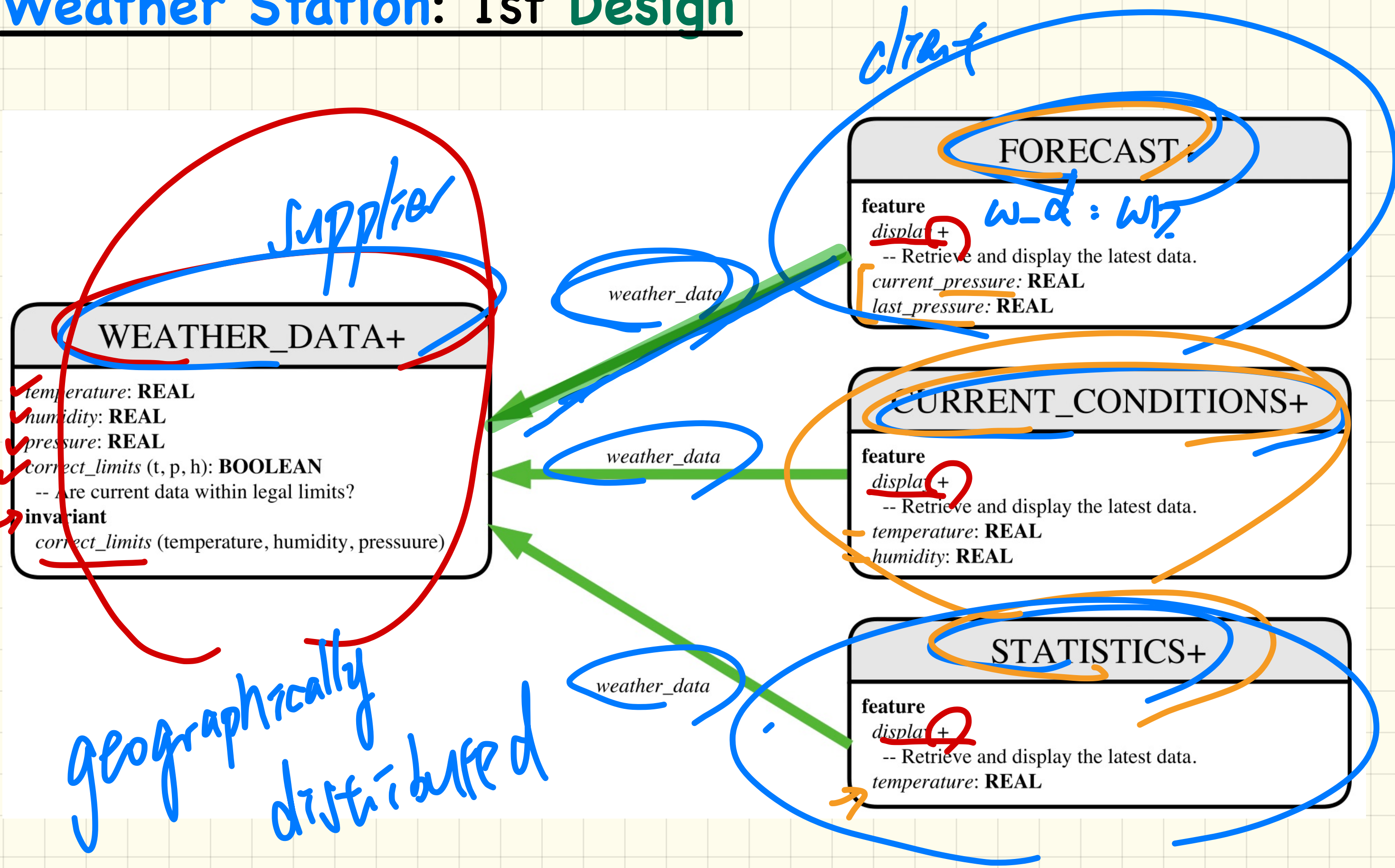
↳ ∵ no supplier can satisfy false

- if there's $\underline{\underline{B}}$

  s.t. B should be the postcond. of every Routine (Cmd or query)

  ↳ B should be a class <u>invariant</u>.

Weather Station: 1st Design



Weather Station:

1st Implementation

```
class WEATHER_DATA create make
feature -- Data
  temperature: REAL
  humidity: REAL
  pressure: REAL
feature -- Queries
  correct_limits(t,p,h: REAL): BOOLEAN
    ensure
      Result implies -36 <=t and t <= 60
      Result implies 50 <= p and p <= 110
      Result implies 0.8 <= h and h <= 100
feature -- Commands
  make (t, p, h: REAL)
    require
      correct_limits(temperature, pressure, humidity)
    ensure
      temperature = t and pressure = p and humidity = h
invariant
  correct_limits(temperature, pressure, humidity)
end
```

```
class FORECAST create make
feature -- Attributes
  current_pressure: REAL
  last_pressure: REAL
  weather_data: WEATHER_DATA
feature -- Commands
  make(wd: WEATHER_DATA)
    ensure weather_data = a_weather_data
  update
    do last_pressure := current_pressure
       current_pressure := weather_data.pressure
    end
  display
    do update
```

```
class CURRENT_CONDITIONS create make
feature -- Attributes
  temperature: REAL
  humidity: REAL
  weather_data: WEATHER_DATA
feature -- Commands
  make(wd: WEATHER_DATA)
    ensure weather_data = wd
  update
    do temperature := weather_data.temperature
       humidity := weather_data.humidity
    end
  display
    do update
```

```
class STATISTICS create make
feature -- Attributes
  weather_data: WEATHER_DATA
  current_temp: REAL
  max, min, sum_so_far: REAL
  num_readings: INTEGER
feature -- Commands
  make(wd: WEATHER_DATA)
    ensure weather_data = a_weather_data
  update
    do current_temp := weather_data.temperature
       -- Update min, max if necessary.
    end
  display
    do update
```

Weather Station:

Testing 1st Design

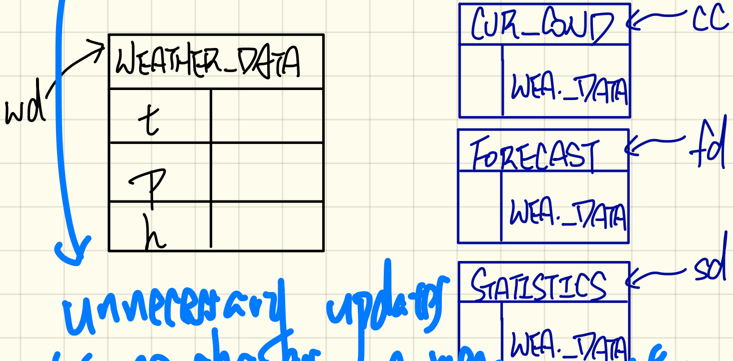
```
class WEATHER_STATION create make
feature -- Attributes
  cc: CURRENT_CONDITIONS ; fd: FORECAST ; sd: STATISTICS
  wd: WEATHER_DATA
feature -- Commands
  make
    do create wd.make (9, 75, 25)
       create cc.make (wd) ; create fd.make (wd) ; create sd.make(wd)

       wd.set_measurements (15, 60, 30.4)
       cc.display ; fd.display ; sd.display
       cc.display ; fd.display ; sd.display

       wd.set_measurements (11, 90, 20)
       cc.display ; fd.display ; sd.display
  end
end
```

after this, app must update

```
class FORECAST create make
feature -- Attributes
  current_pressure: REAL
  last_pressure: REAL
  weather_data: WEATHER_DATA
feature -- Commands
  make(wd: WEATHER_DATA)
    ensure weather_data = a_weather_data
  update
    do last_pressure := current_pressure
       current_pressure := weather_data.pressure
    end
  display
    do update
```

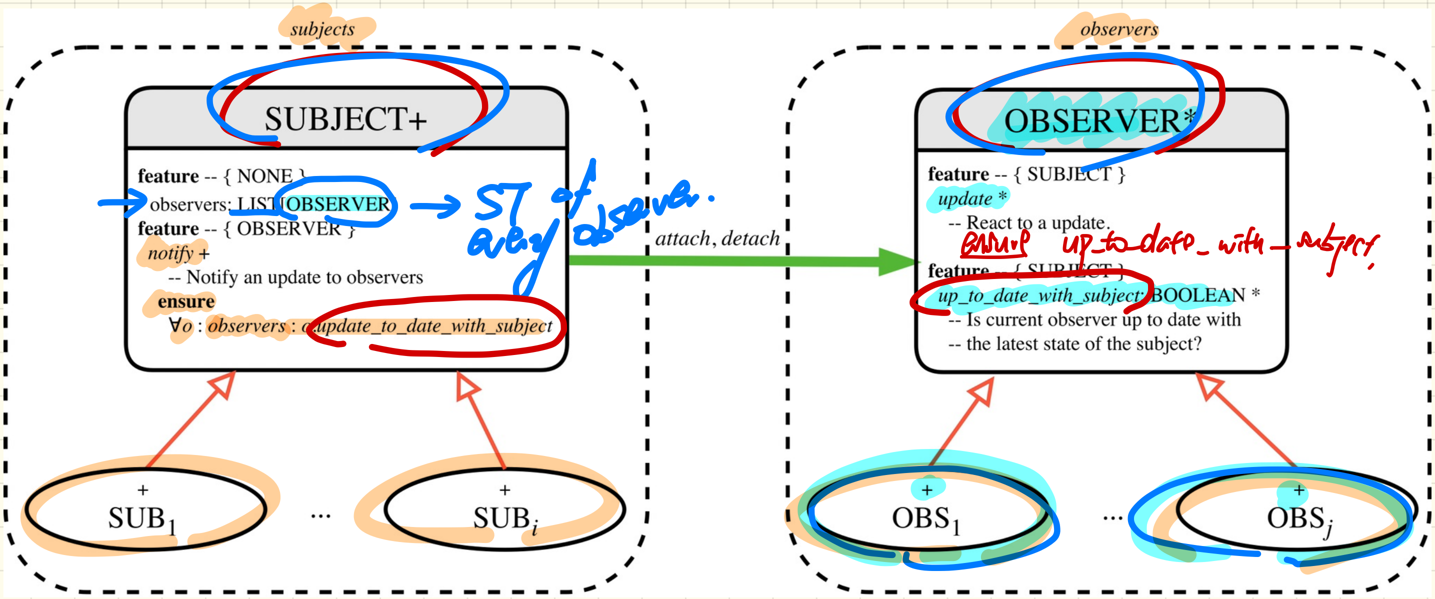
```
class CURRENT_CONDITIONS create make
feature -- Attributes
  temperature: REAL
  humidity: REAL
  weather_data: WEATHER_DATA
feature -- Commands
  make(wd: WEATHER_DATA)
    ensure weather_data = wd
  update
    do temperature := weather_data.temperature
       humidity := weather_data.humidity
    end
  display
    do update
```

```
class STATISTICS create make
feature -- Attributes
  weather_data: WEATHER_DATA
  current_temp: REAL
  max, min, sum_so_far: REAL
  num_readings: INTEGER
feature -- Commands
  make(wd: WEATHER_DATA)
    ensure weather_data = a_weather_data
  update
    do current_temp := weather_data.temperature
       -- Update min, max if necessary.
    end
  display
    do update
```

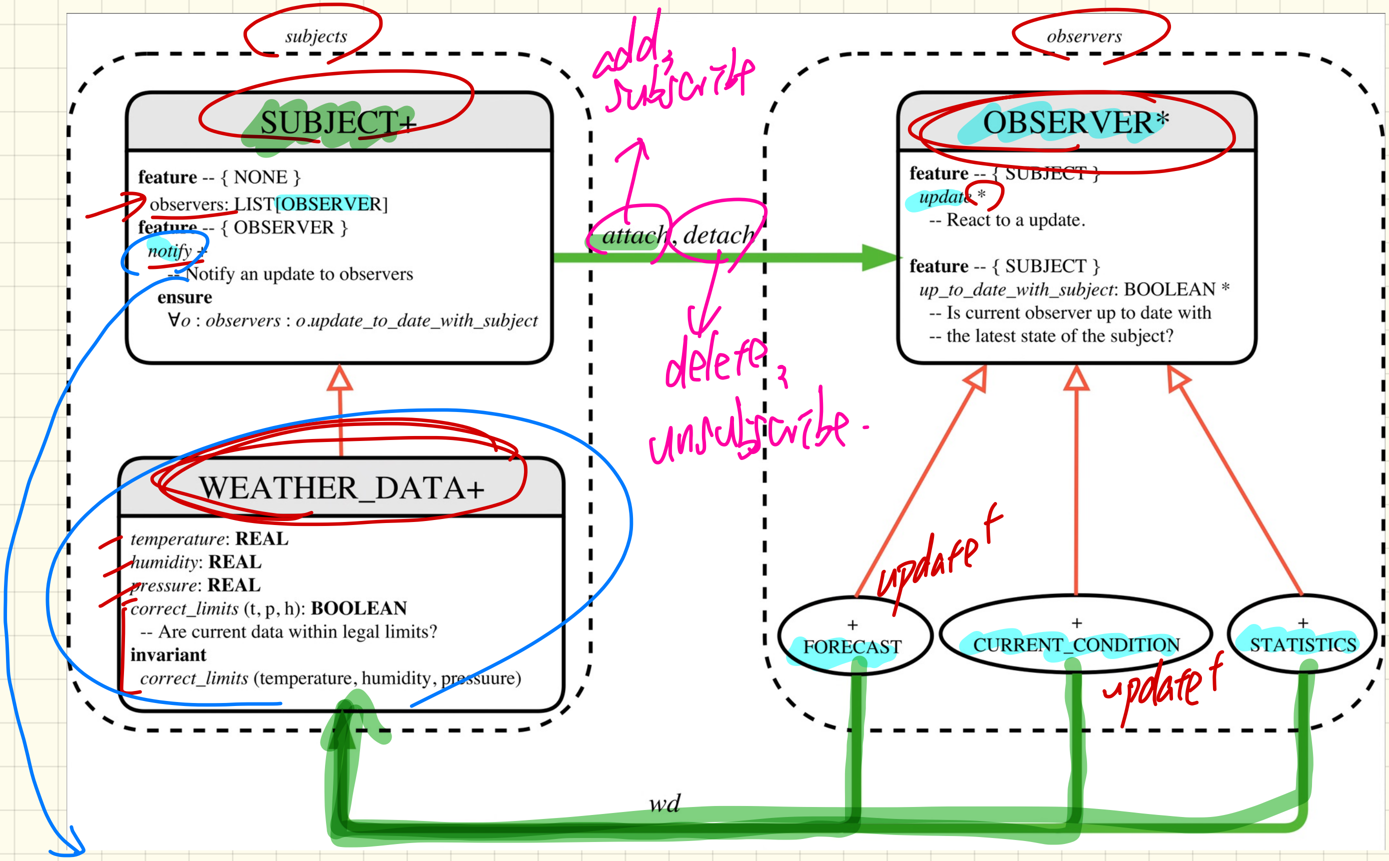


unnecessary updates ∵ no changes on measurement.

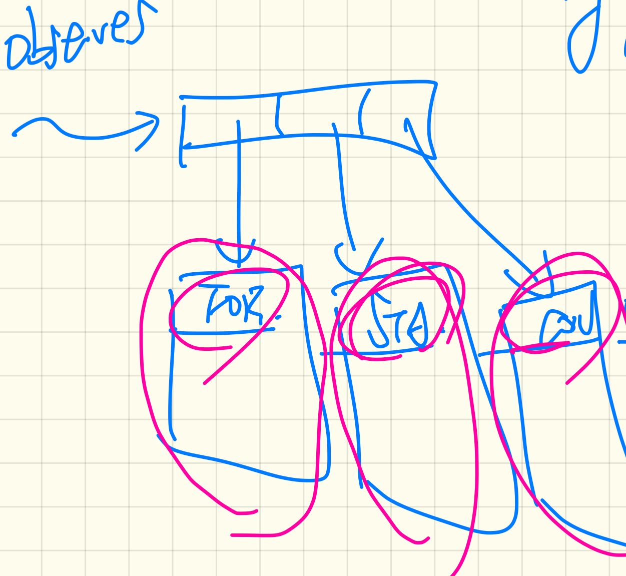
The Observer Pattern


subjects
SUBJECT+
feature -- { NONE }
observers: LIST[OBSERVER]
ST of every observer.
feature -- { OBSERVER }
notify +
-- Notify an update to observers
ensure
∀o : observers : o.update_to_date_with_subject
attach, detach
observers
OBSERVER*
feature -- { SUBJECT }
update *
-- React to a update.
ensure up_to_date_with_subject
feature -- { SUBJECT }
up_to_date_with_subject: BOOLEAN *
-- Is current observer up to date with
-- the latest state of the subject?
+
SUB₁
…
+
SUBᵢ
+
OBS₁
…
+
OBSⱼ

The Observer Pattern: Application to Weather Station



```
class SUBJECT
    observers : LIST[OBSERVER]

    notify do
        across observers is observer
        loop
            observer.update
        end
    end
```


observers
FOR
STA
CU