Tuesday Oct. 30

Lecture 14

Interactive System: non-OO vs. OO

current_state : STATE

current_state . execute_session

C.O.

create {INITIAL} cs.make

cs . display

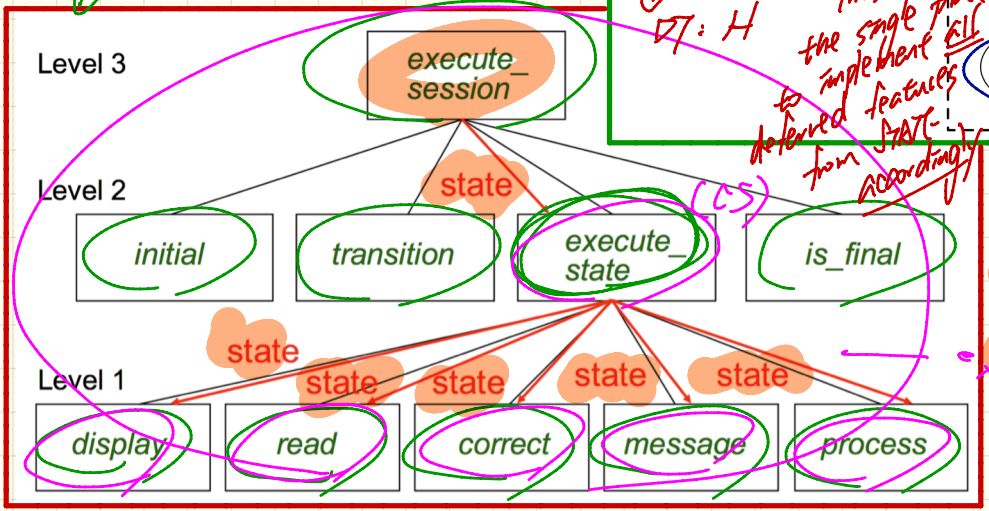
DT: I

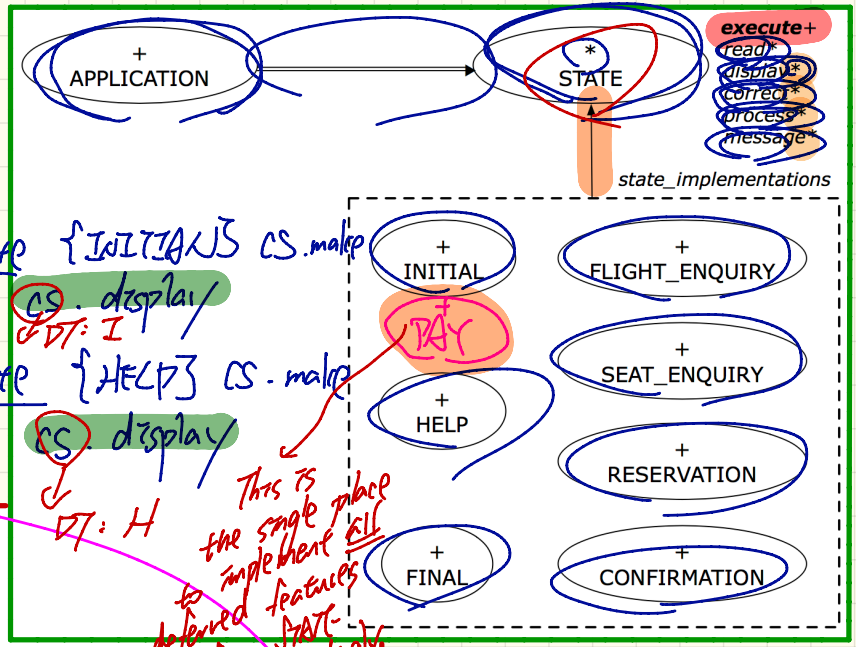
create {HELP} cs . make

cs . display

DT: H

This is the single place to implement all deferred features from STATE - according to





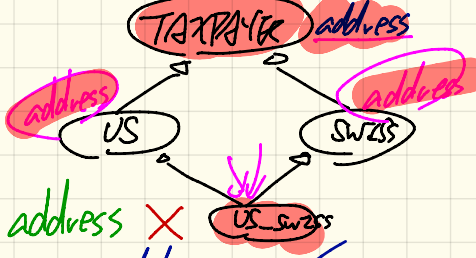
current_state : INTEGER

execute_session (current_state)

Multiple Inheritance : Example (1)

tpl : US_TAXPAYER

tpl : SWISS_US_TAXPAYER



tpl

tpl. address ✗

tpl. us_address ✓

tpl. pay_taxes ✗

tpl. pay_us_taxes ✓

tpl. swiss_address

tpl. swiss_tax_id

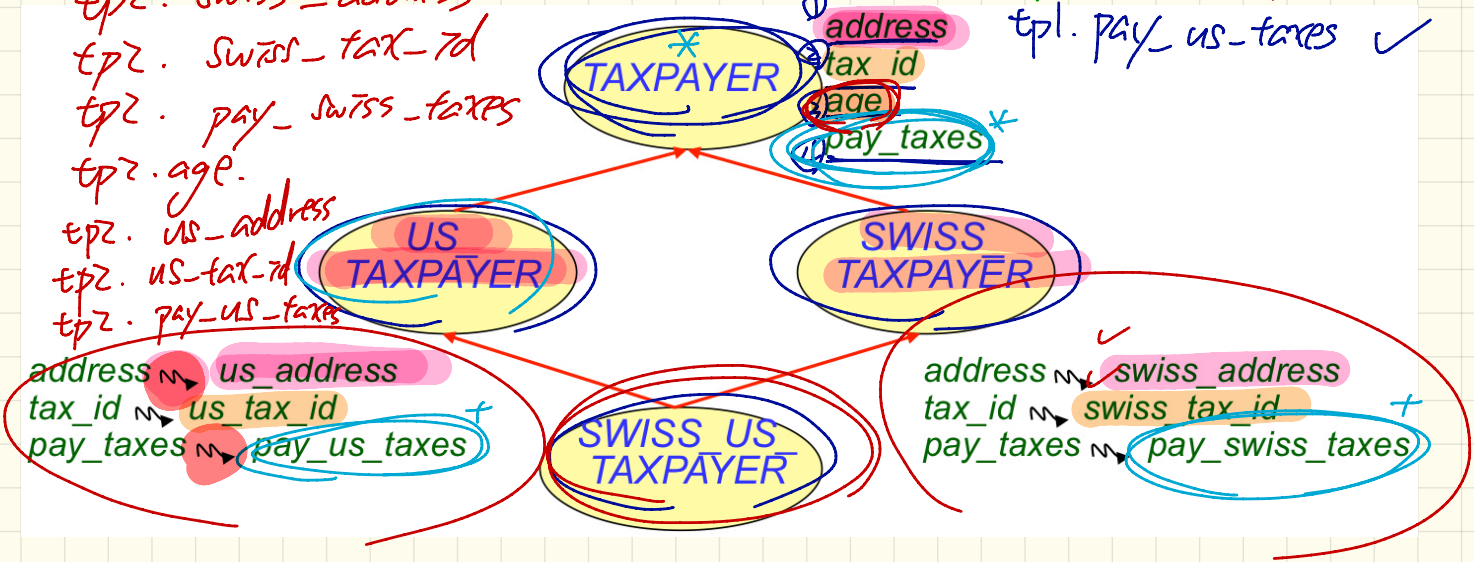
tpl. pay_swiss_taxes

tpl. age.

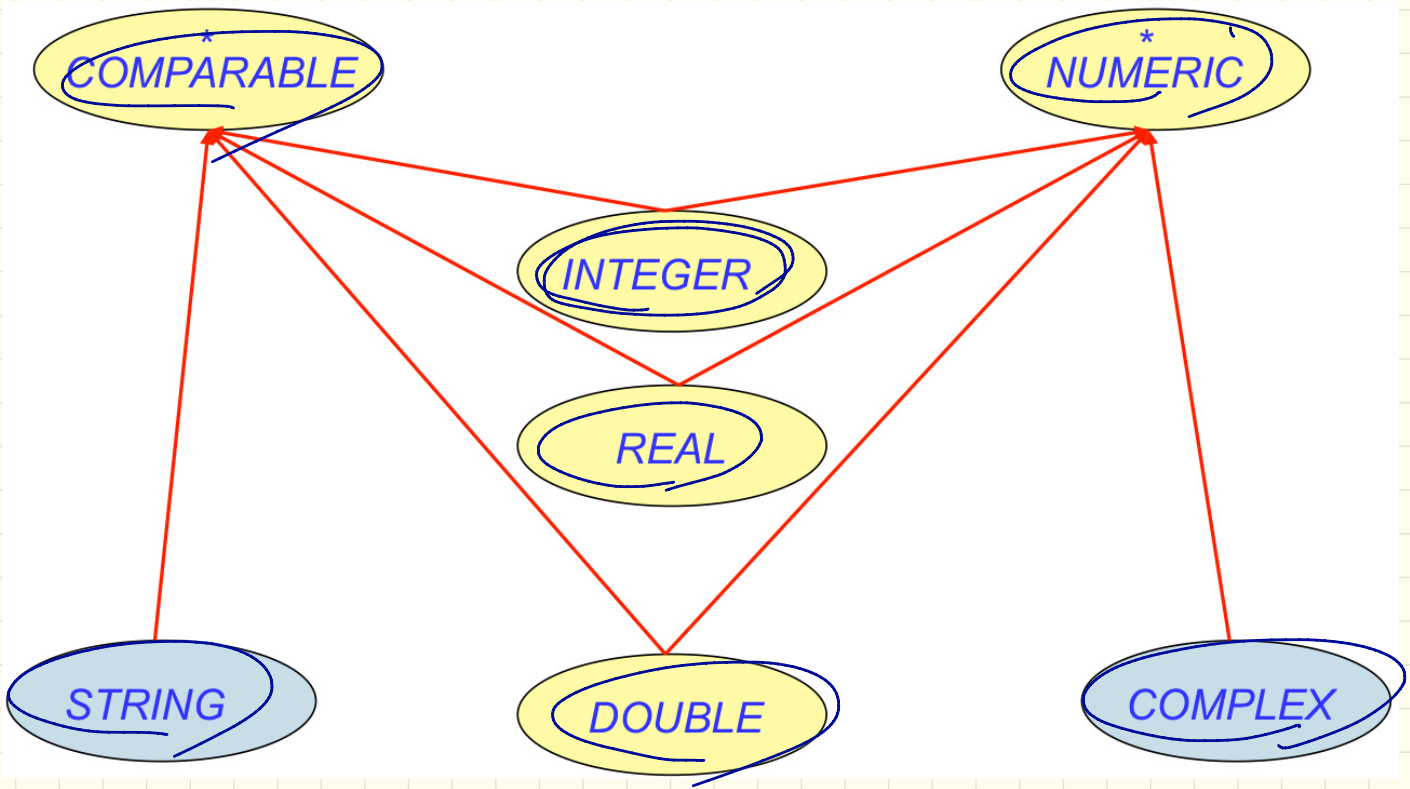
tpl. us_address

tpl. us_tax_id

tpl. pay_us_taxes



Multiple Inheritance : Example (2)



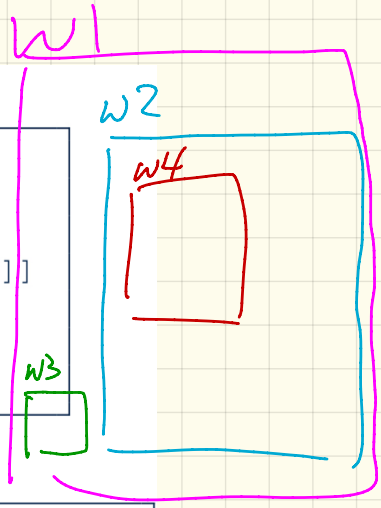
Multiple Inheritance : Exercise

w2.add(w4)

ST: Window

```
class RECTANGLE
  feature -- Queries
    width, height: REAL
    xpos, ypos: REAL
  feature -- Commands
    make (w, h: REAL)
    change_width
    change_height
    move
end
```

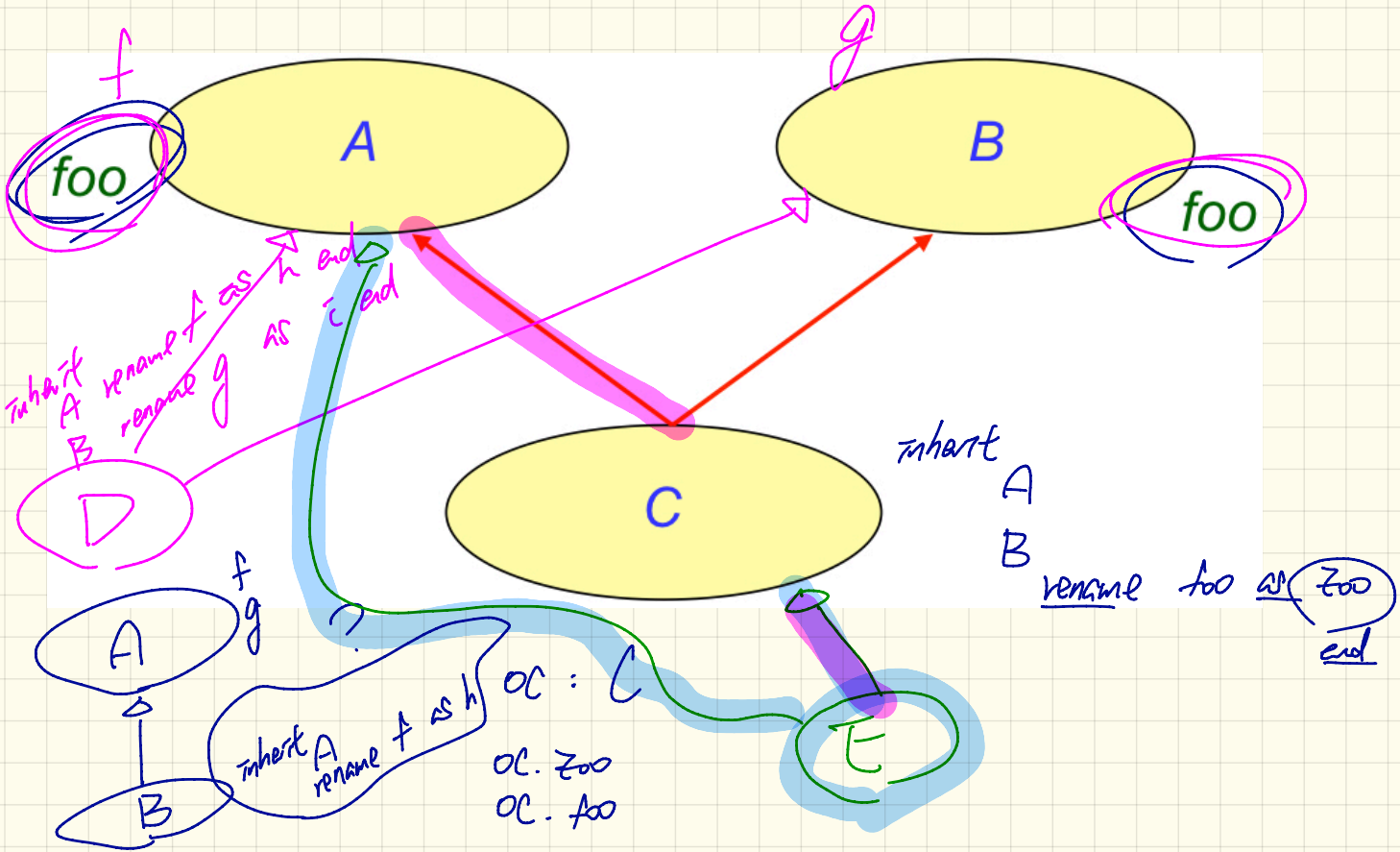
```
class TREE[G]
  feature -- Queries
    parent: TREE[G]
    descendants: LIST[TREE[G]]
  feature -- Commands
    add_child (c: TREE[G])
end
```



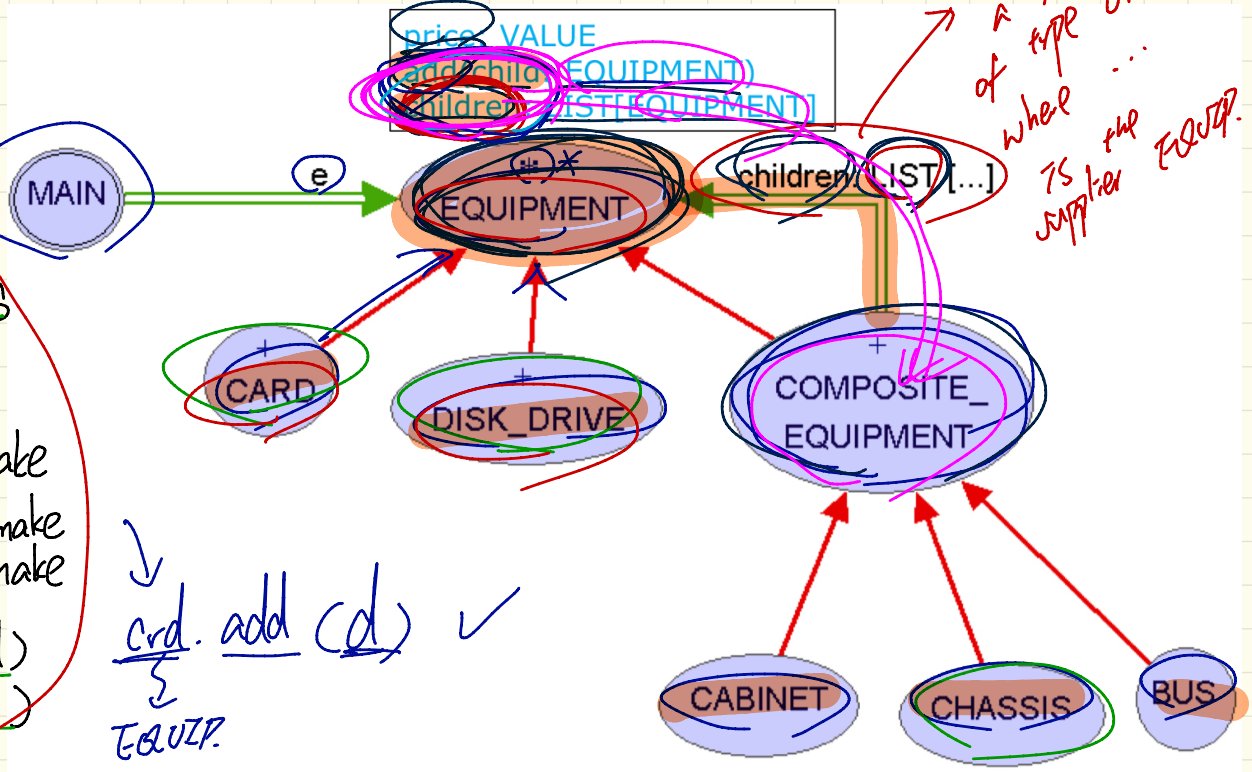

```
class WINDOW
  inherit
    RECTANGLE
    TREE[WINDOW]
feature
  add (w: WINDOW)
end
```

```
test_window: BOOLEAN
  local w1, w2, w3, w4: WINDOW
  do
    create w1.make(8, 6) ; create w2.make(4, 3)
    create w3.make(1, 1) ; create w4.make(1, 1)
    w2.add(w4) ; w1.add(w2) ; w1.add(w3)
    Result := w1.descendants.count = 2
  end
```

Multiple Inheritance: Name Clashes



First Design Attempt



C_E being the client declares feature of type LIST[...] where ... is the supplier EQUIP.

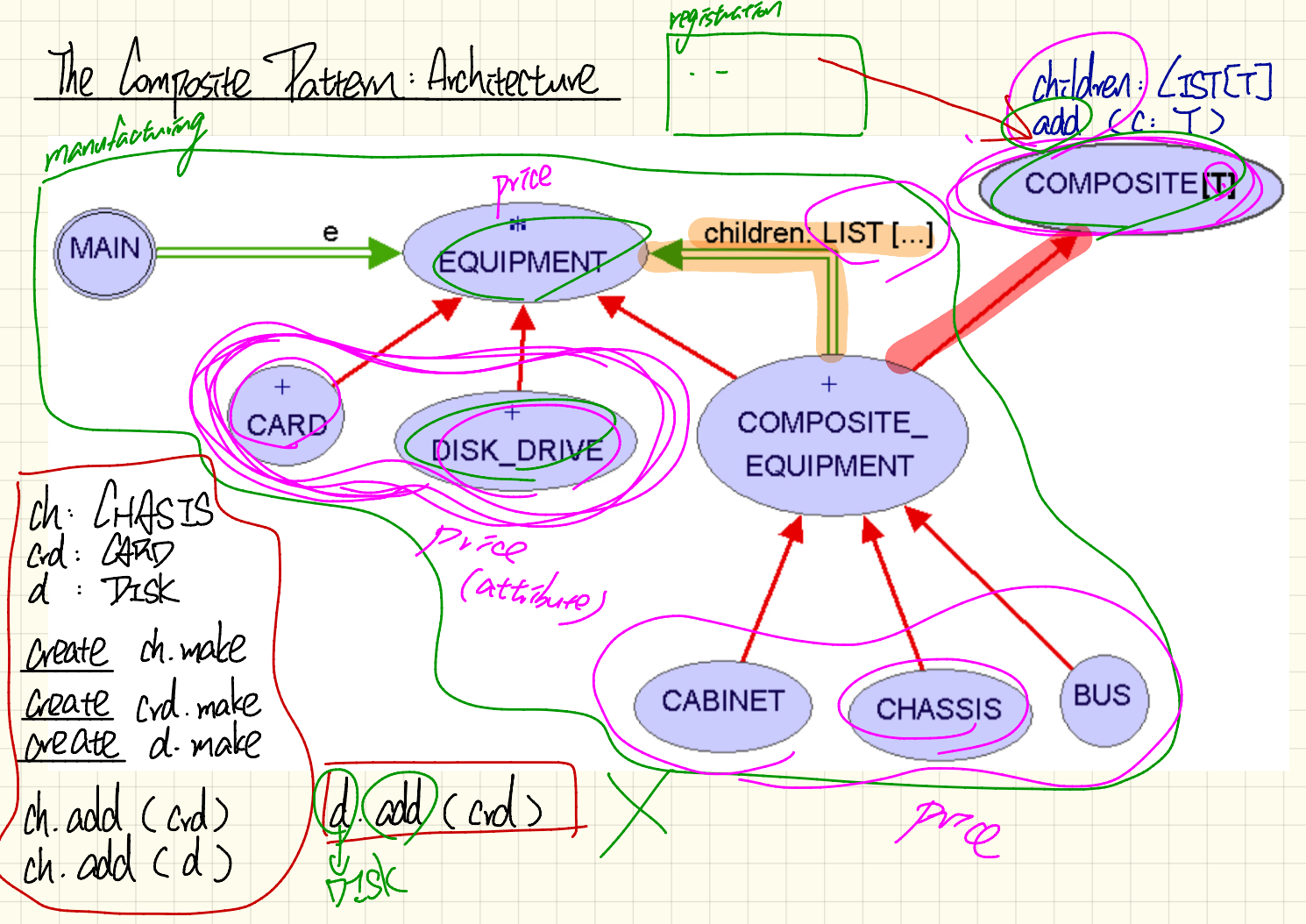
ch: CHASIS
crd: CARD
d : DISK

create ch.make
create crd.make
create d.make

ch.add (crd)
ch.add (d)

crd.add (d) ✓
↳ EQUIP.

The Composite Pattern : Architecture



The Composite Pattern: Implementation

```
deferred class
  EQUIPMENT
feature
  name: STRING
  price: REAL -- uniform access principle
end
```

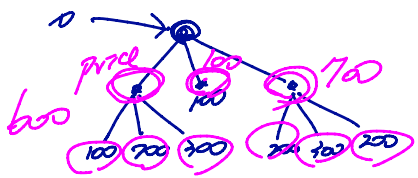
```
deferred class
  COMPOSITE[T]
feature
  children: LINKED_LIST[T]

  add_child (c: T)
    do
      children.extend (c) -- Polymorphism
    end
end
```

```
class
  CARD
inherit
  EQUIPMENT
feature
  make (n: STRING; p: REAL)
    do
      name := n
      price := p -- price is an attribute
    end
end
```

Storage

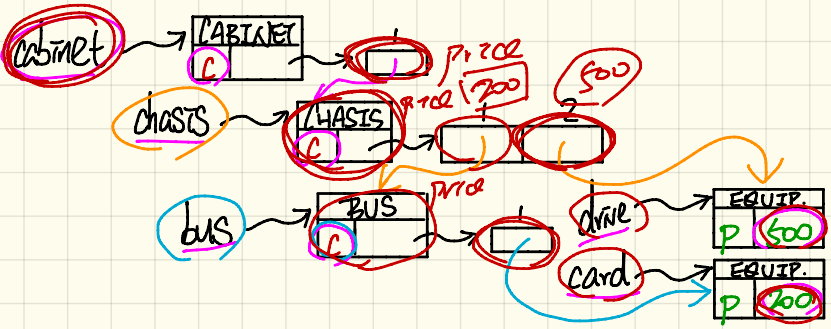
```
class
  COMPOSITE_EQUIPMENT
inherit
  EQUIPMENT
  COMPOSITE [EQUIPMENT]
create
  make
feature
  make (n: STRING)
    do name := n ; create children.make end
  price : REAL -- price is a query
      -- Sum the net prices of all sub-equipments
    do
      across
        children as cursor
      loop
        Result := Result + cursor.item.price -- dynamic binding
      end
    end
end
```



Testing the Composite Pattern

```
test_composite_equipment: BOOLEAN
  local
    card, drive: EQUIPMENT
    cabinet: CABINET -- holds a CHASSIS
    chassis: CHASSIS -- contains a BUS and a DISK_DRIVE
    bus: BUS -- holds a CARD
  do
    create {CARD} card.make("16Mbs Token Ring", 200)
    create {DISK_DRIVE} drive.make("500 GB harddrive", 500)
    create bus.make("MCA Bus")
    create chassis.make("PC Chassis")
    create cabinet.make("PC Cabinet")

    bus.add(card)
    chassis.add(bus)
    chassis.add(drive)
    cabinet.add(chassis)
    Result := cabinet.price = 700
  end
```




```
class
  CARD
inherit
  EQUIPMENT
feature
  make (n: STRING; p: REAL)
    do
      name := n
      price := p -- price is
    end
end
```

```
class
  COMPOSITE_EQUIPMENT
inherit
  EQUIPMENT
  COMPOSITE [EQUIPMENT]
create
  make
feature
  make (n: STRING)
    do name := n ; create children.make end
  price : REAL -- price is a query
      -- Sum the net prices of all sub-equip
    do
      across
        children as cursor
      loop
        Result := Result + cursor.item.price
      end
    end
end
```