Wednesday March 6

Lecture 15

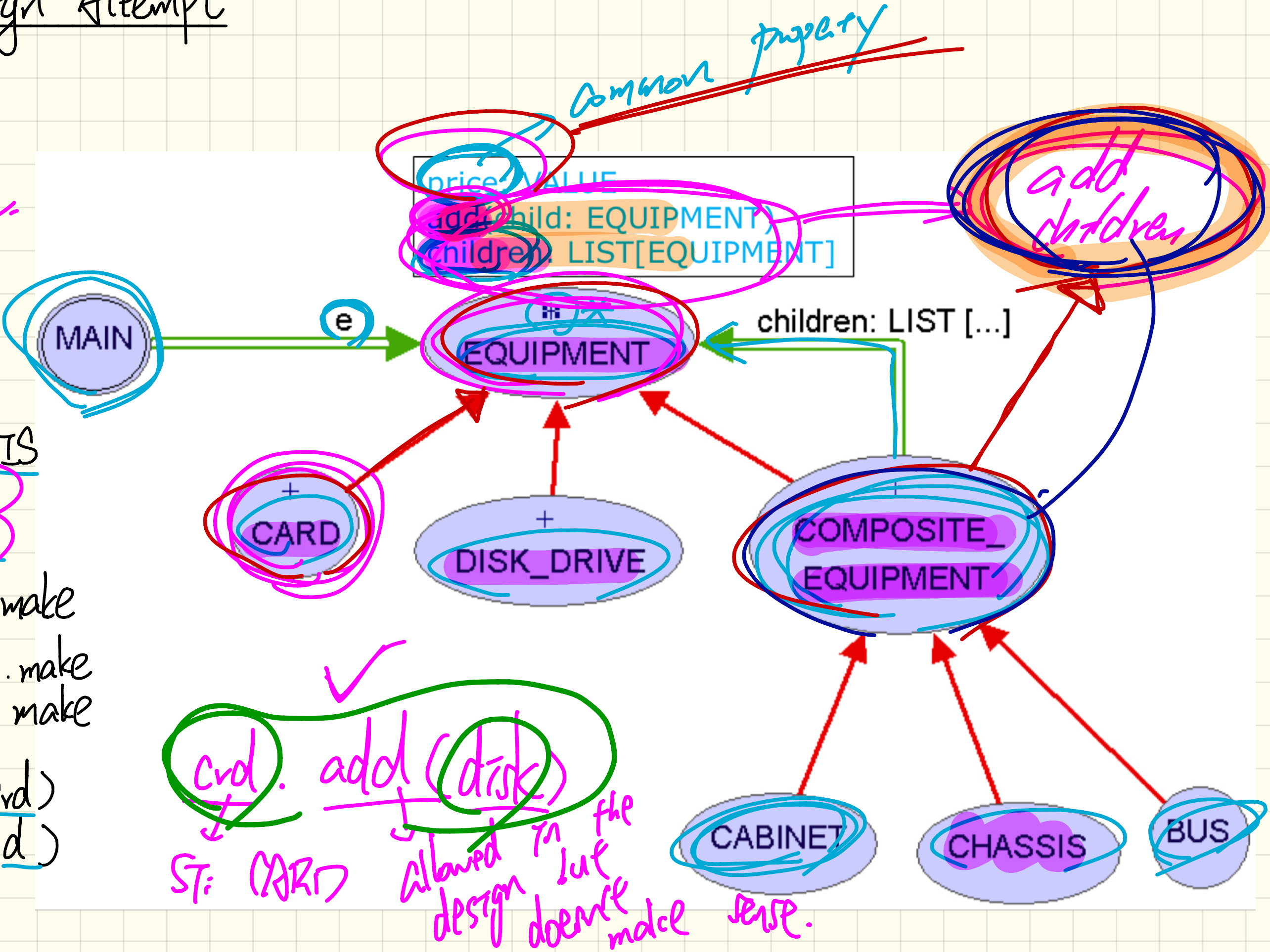
First Design Attempt

cohesion

ch : CHASIS
crd : CARD
d : DISK

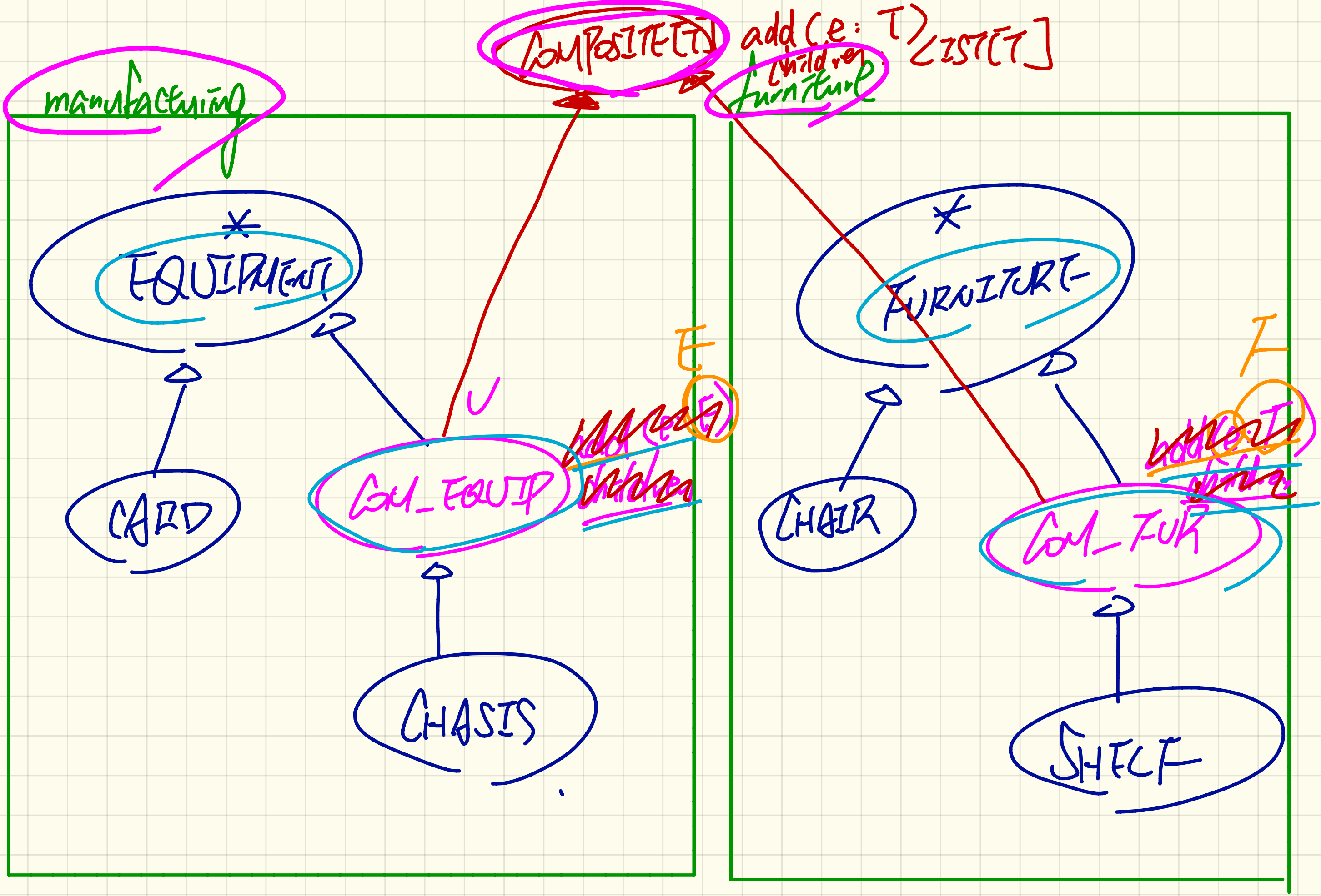
create ch.make
create crd.make
create d.make

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

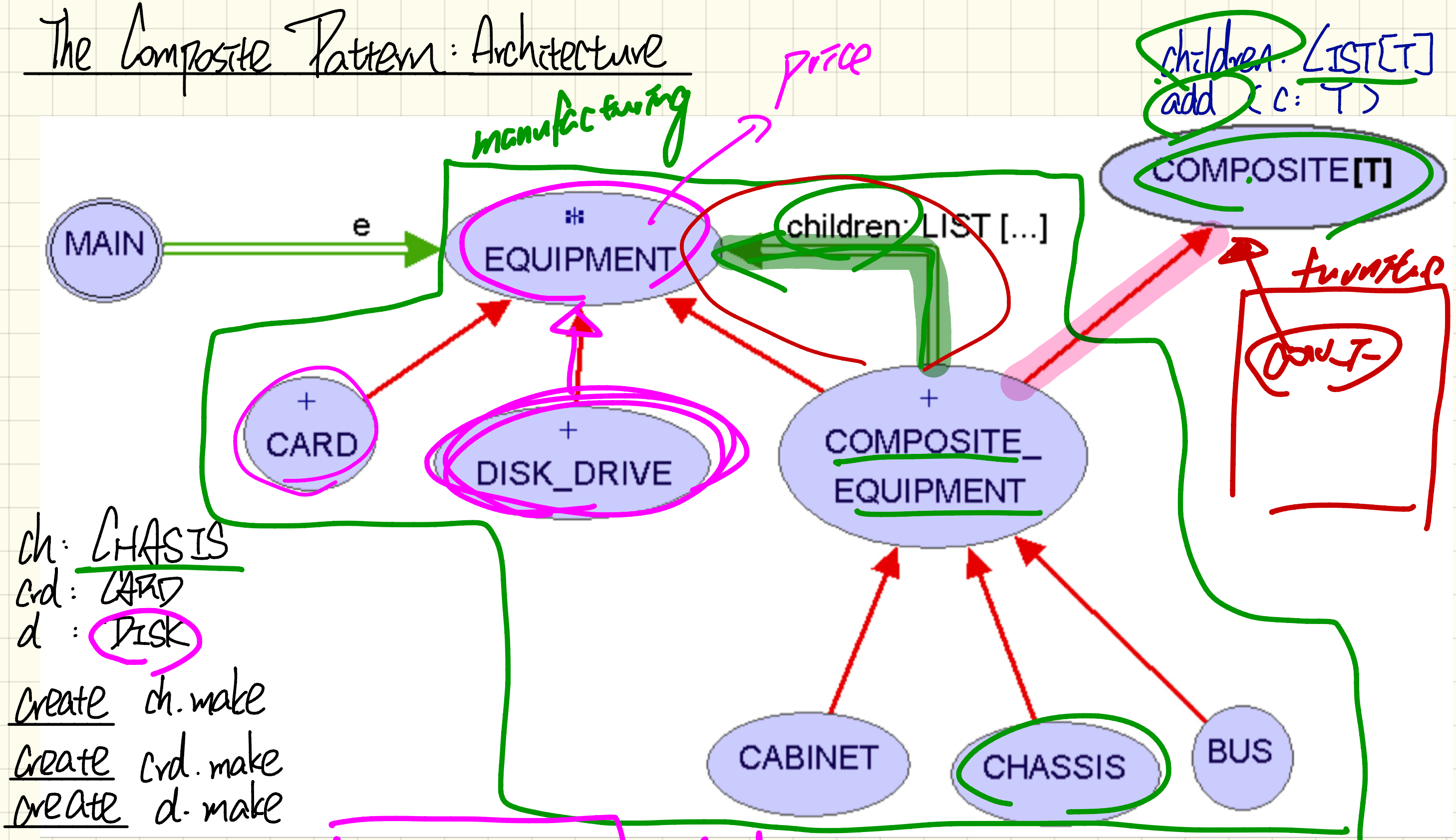


crd. add (disk)
ST: CARD

allowed in the design but doesn't make sense.


COMPOSITE[T]
add(e: T)
children: LIST[T]
furniture
manufacturing
*
EQUIPMENT
CARD
U
COM_EQUIP
E
~~add(e: T)~~
~~children~~
CHASIS
*
FURNITURE
CHAIR
COM_FUR
E
~~add(e: T)~~
~~children~~
SHELF

The Composite Pattern: Architecture



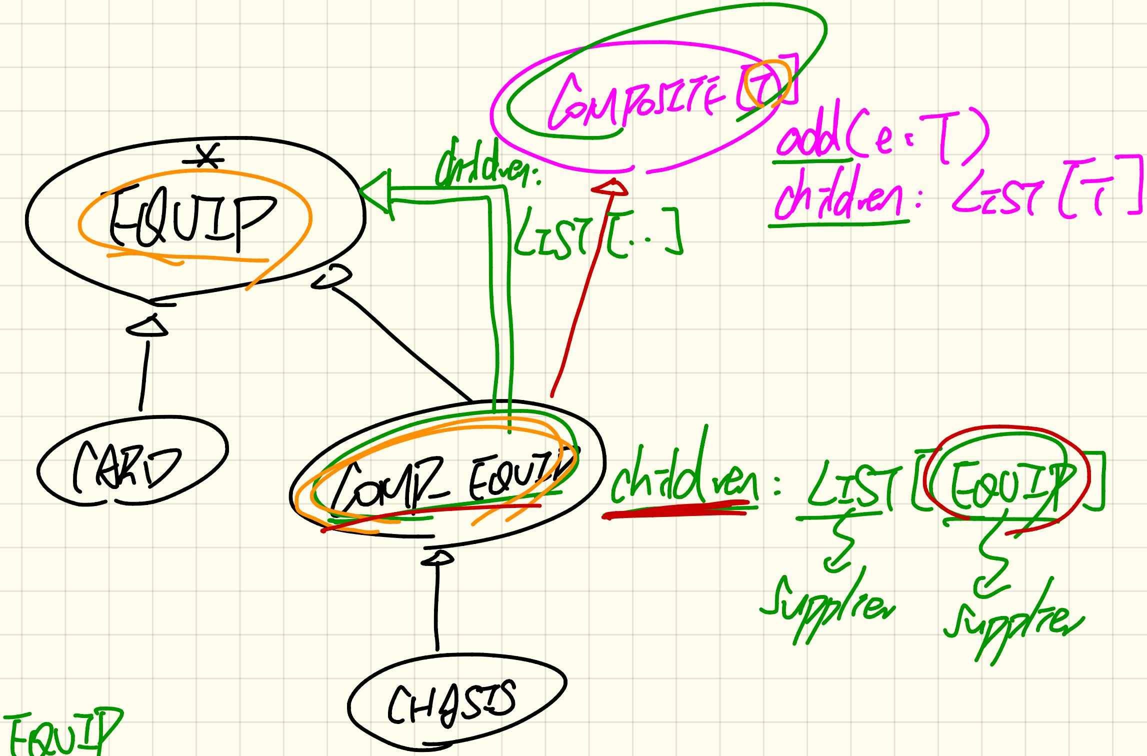
```
ch  : CHASIS
crd : CARD
d   : DISK

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

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

```
d.add (crd)
```

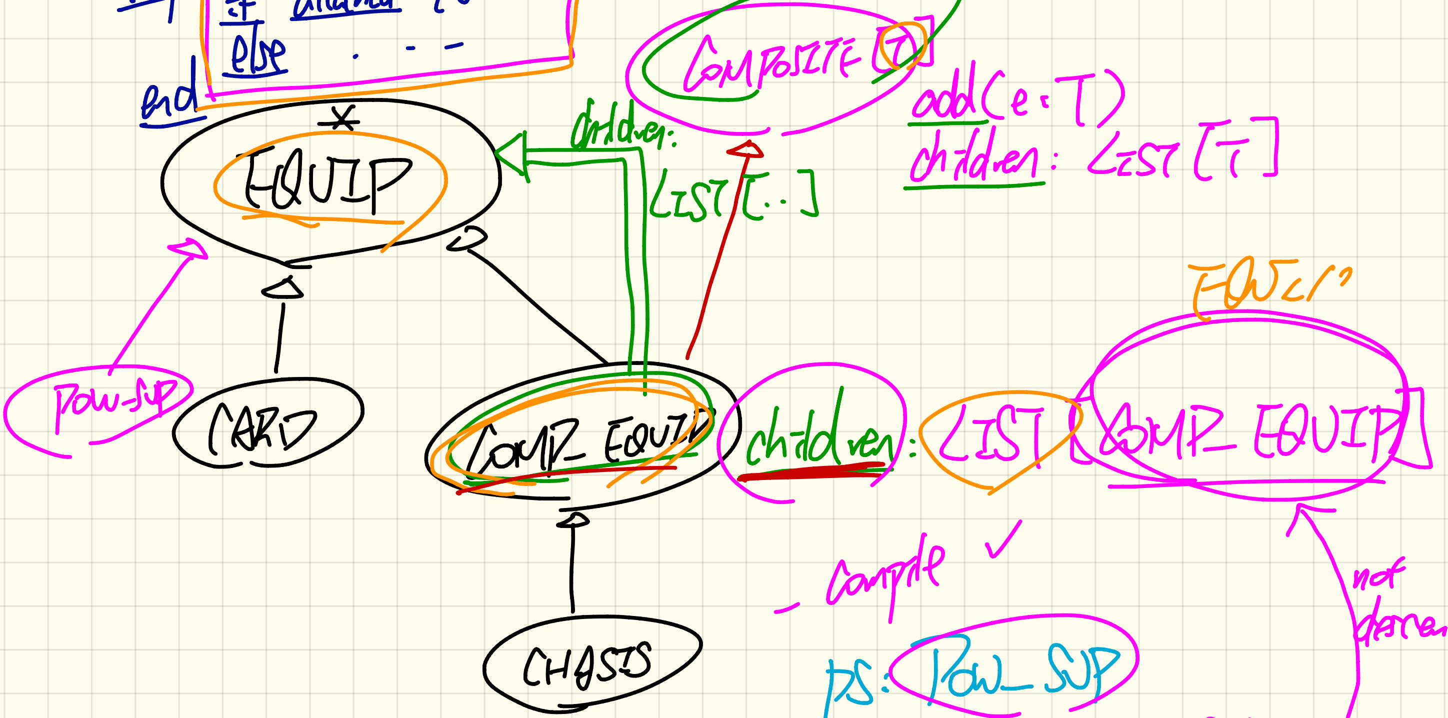
X doesn't even compile.



```
class COMP_EQUIP
  inherit
    COMPOSITE[EQUIP]
```

```
across children as c
loop
    if attached {EQUIP} c.item then - - end
    else . - -
end
```

→ EQUIP c.item.price

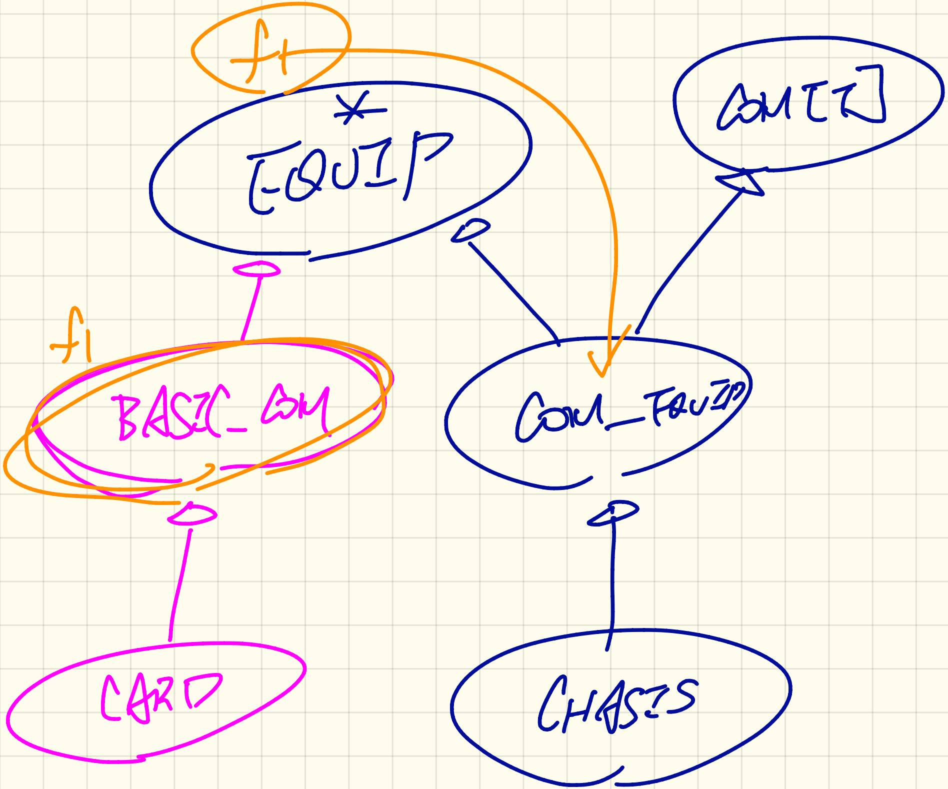


- compile ✓

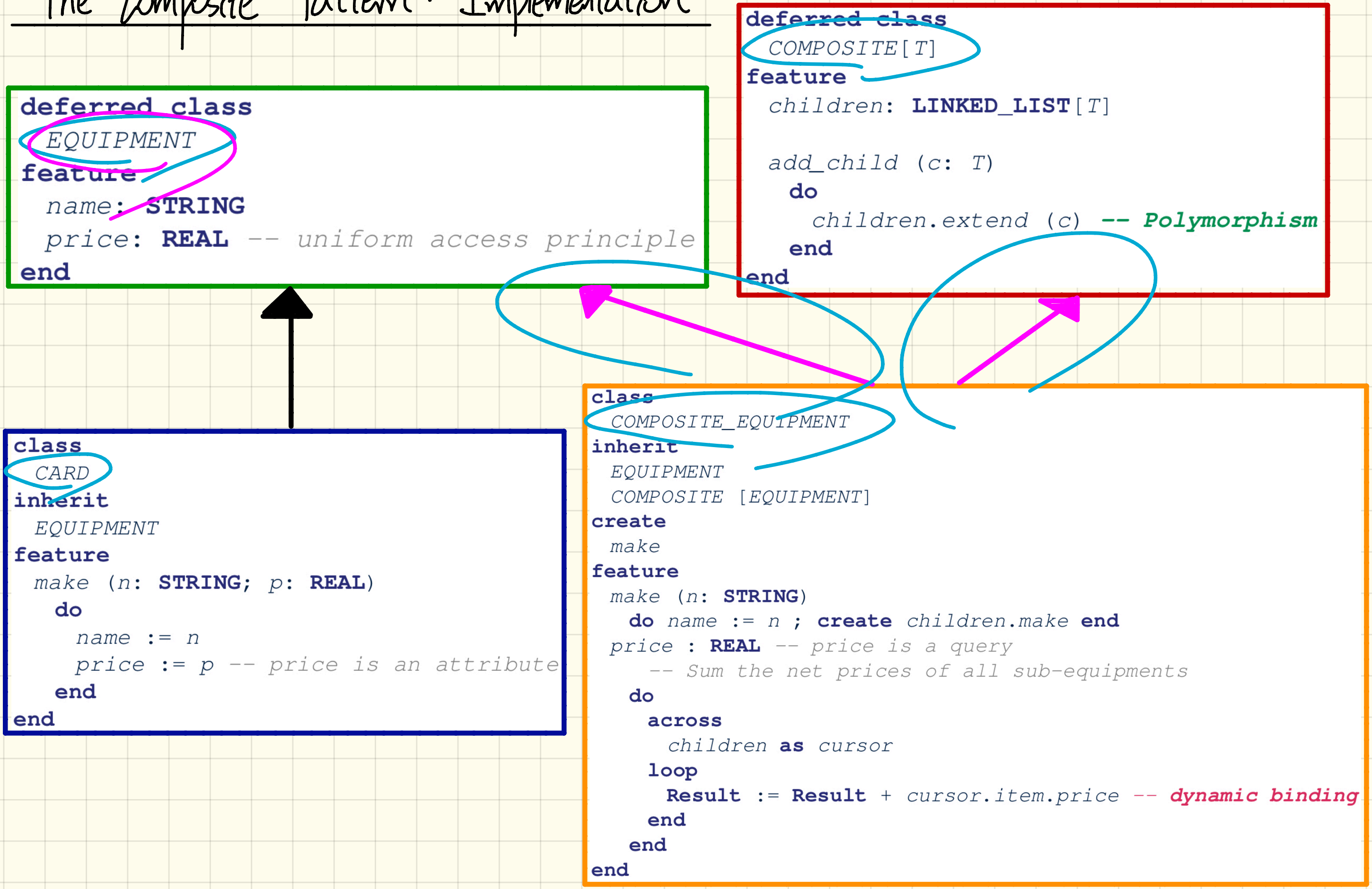
```
ps: POW_SUP
c : CABINET
c.add (ps)
```

ST: POW_SUP

not element


f1
*
EQUIP
COM[T]
f1
BASIC_COM
COM_EQUIP
CARD
CHASIS

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
```

```
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
```

cabinet. price

card. price

DT: CABNT

cabinet. price

↳ chassis. price

↳ bus. price

↳ card. price

↳ drive. price

200

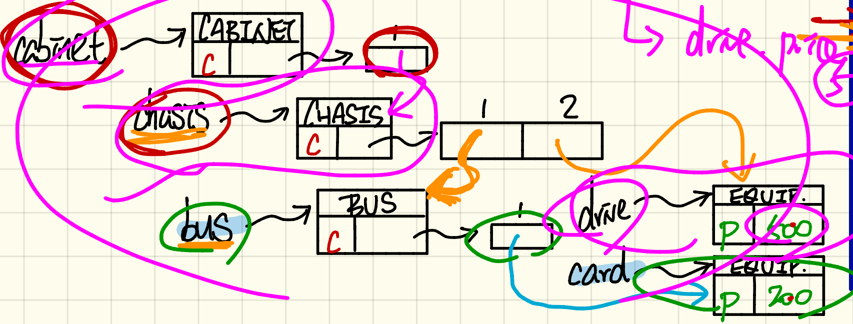
500

```
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
```

1. bus.price

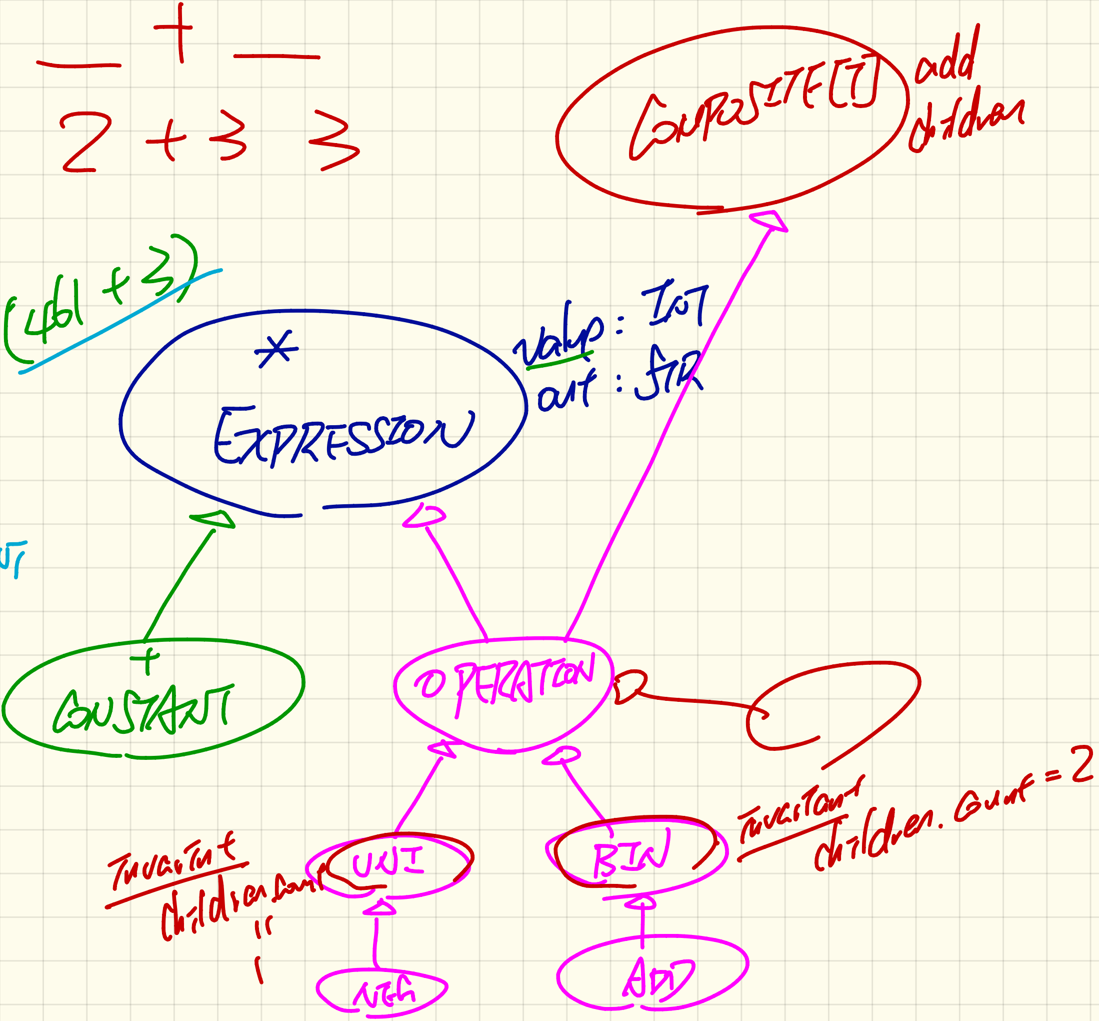
chassis



341
2 -2

341 + 2

(341 + 2) + (461 + 3)

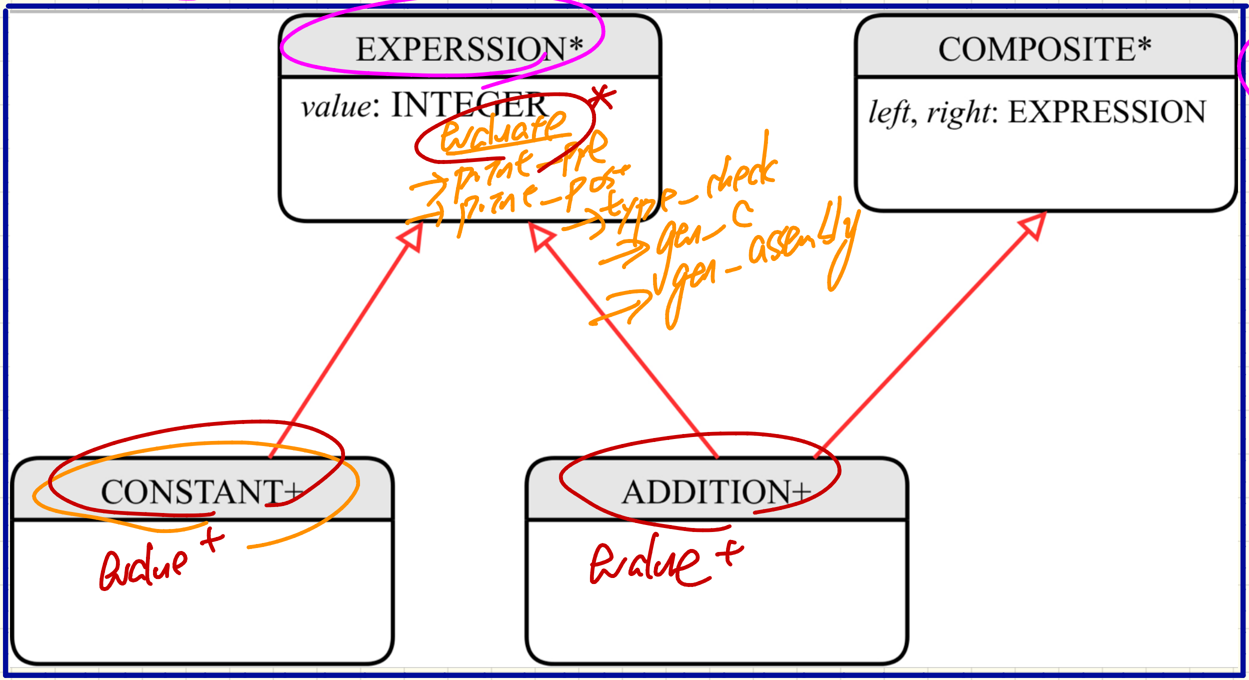


c1, c2, c3: CONSTANT

add: ADDITION

add.(c1)
add.(c2)
add.(c3)

Design of Language Operations : How to Extend the Composite Pattern?



EXPERSSION*

value: INTEGER

COMPOSITE*

left, *right*: EXPRESSION

CONSTANT+

ADDITION+

Structure

Composite

Operations: evaluate
print_prefix
print_postfix
type_check

Operations

3 + 4

7

3 4 +

+ 3 4