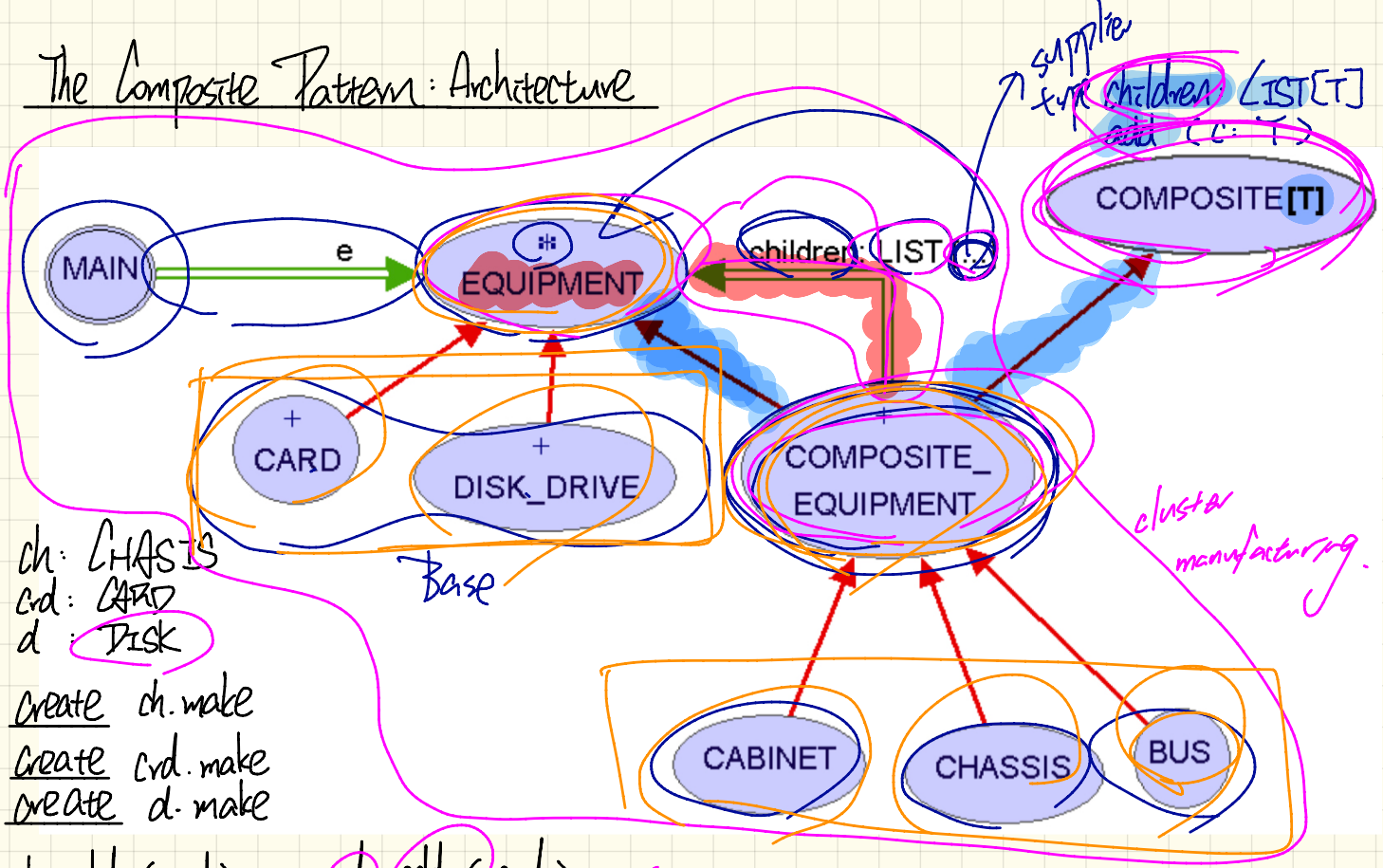
Thursday Nov. 7

Lecture 15

The Composite Pattern: Architecture



ch: CHASIS
cd: CARD
d : DISK

create ch.make
create cd.make
create d.make

ch.add (cd)
ch.add (d)

d.add (cd) ×

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
```

attribute

```
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
```

children : LL [EQUIP]

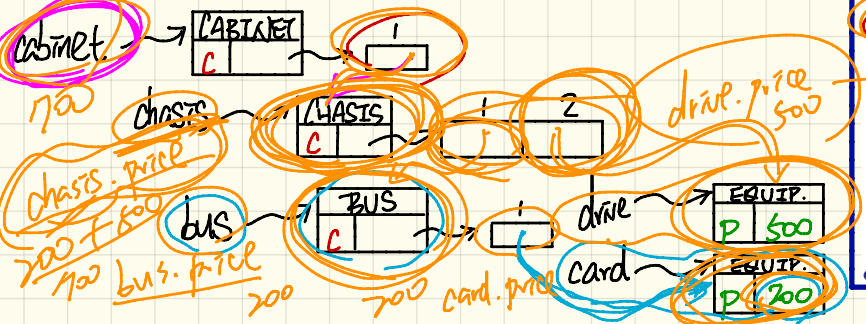
UAP: if the current child is basic ⇒ att.
otherwise ⇒ comput.

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
```

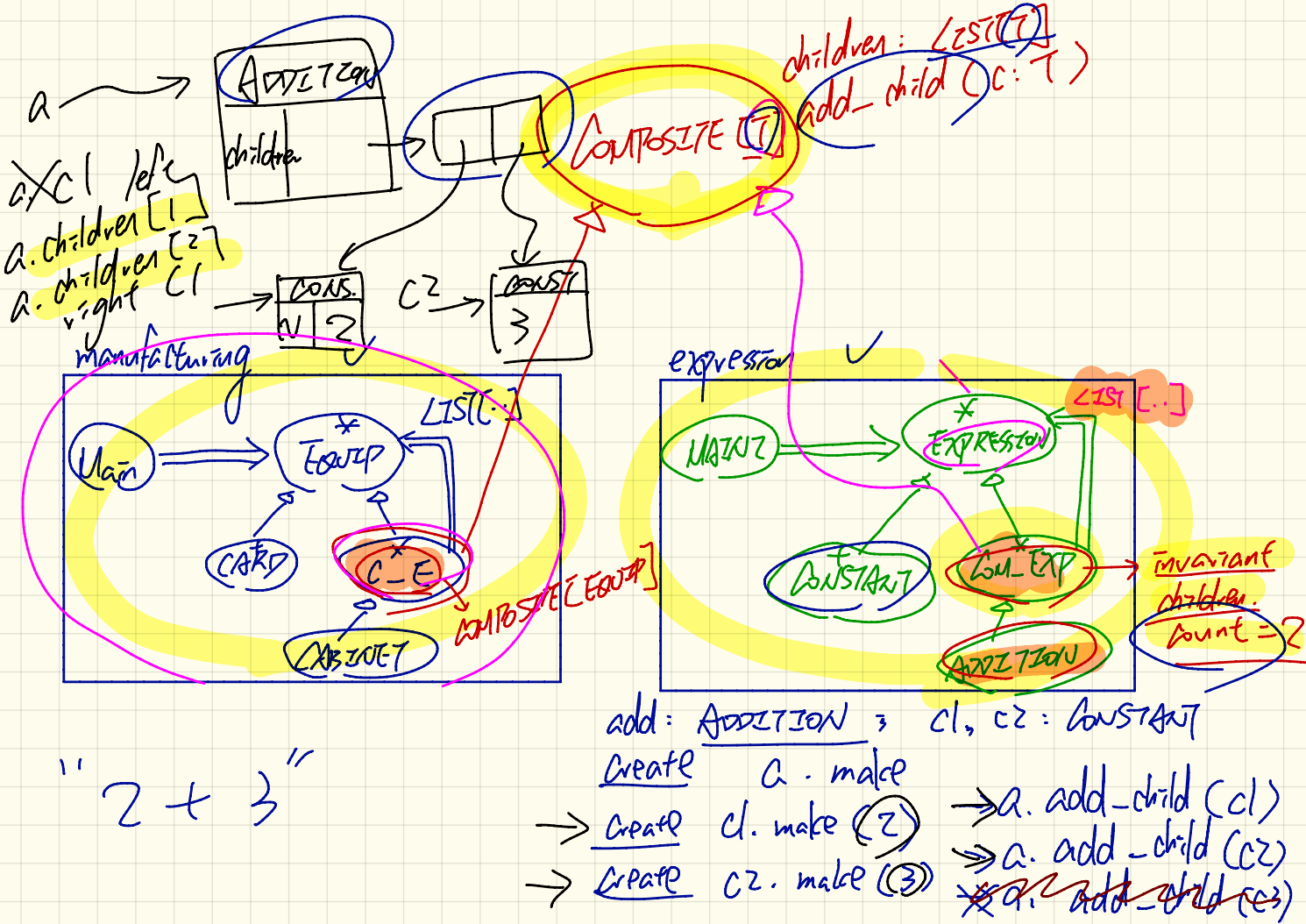
DT: CABINET



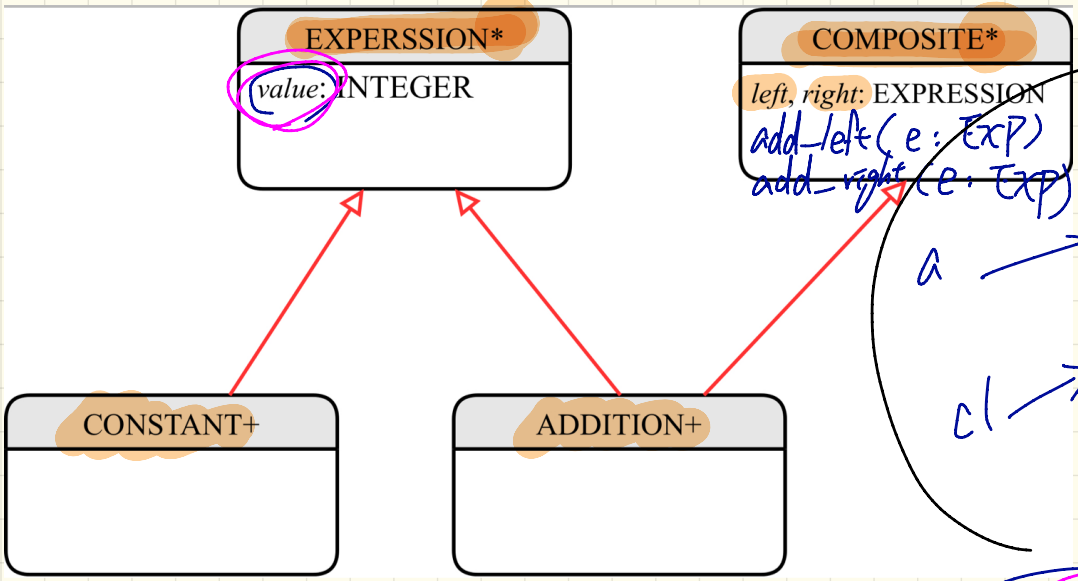

```
class
  CARD
inherit
  EQUIPMENT
feature
  make (n: STRING; p: REAL)
    do
      name := n
      price := p -- price is
    end
end
```

```
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
```

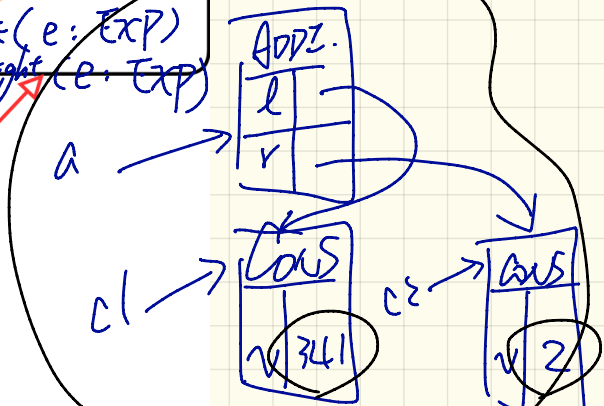
$$\frac{341}{(2+3)} + \frac{\quad}{((4+7)+9)}$$


a
ADDITION
children
CONSTANT
CONSTANT
COMPOSITE[T]
children: LIST[T]
add_child(c:T)
a.children[1]
a.children[2]
c1 left
c2 right
manufacturing
Main
EQUIP
LIST[..]
CARD
C_E
ABSOLUTE
COMPOSITE[EQUIP]
expression
MAIN
EXPRESSION
LIST[..]
CONSTANT
CON_EXP
ADDITION
invariant
children.
count=2
"2+3"
add: ADDITION ; c1, c2 : CONSTANT
create a.make
create c1.make(2)
create c2.make(3)
a.add_child(c1)
a.add_child(c2)
a.add_child(c3)

Design of Language Structure : Composite Pattern



a.value
a.left,
a.right



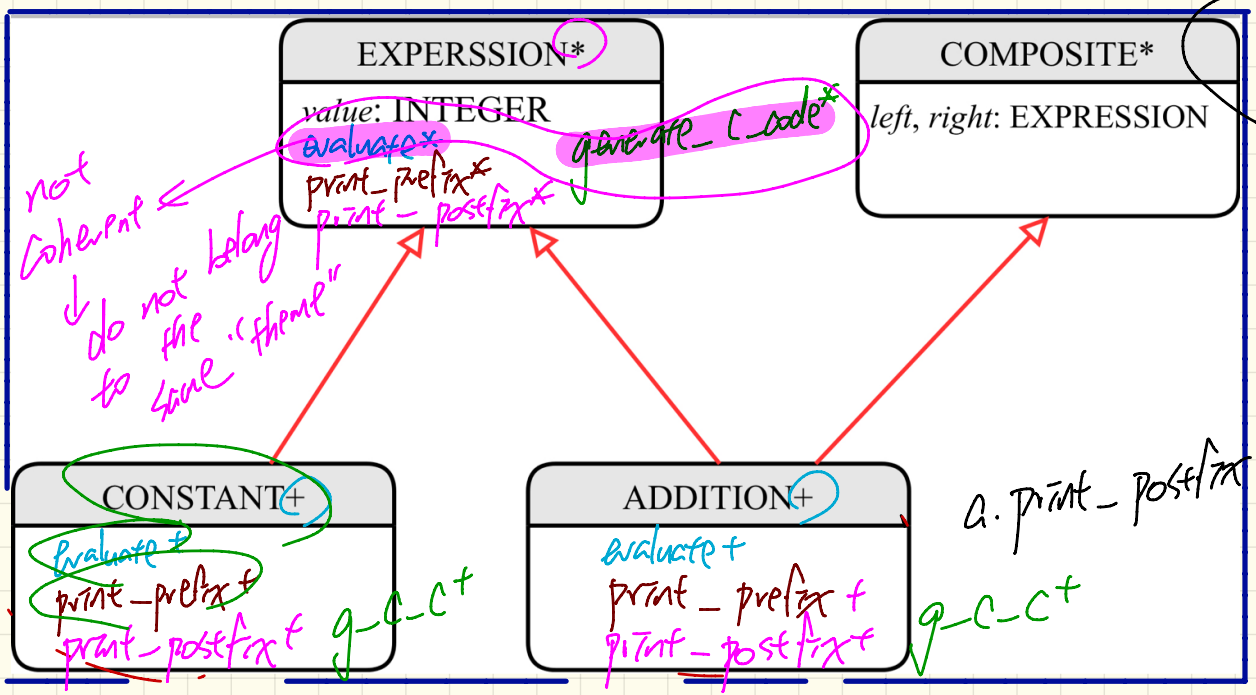
Q: How do you construct an object representing "341 + (2)"?

c1, c2: CONSTANT
a: ADDITION

Create c1.make (341)
Create c2.make (2)

Create @.make
a.add_left (c1)
a.add_right (c2)

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



not Coherent ↓ do not belong to the "theme"

Structure

a. print_postfix 341 2 +

Operations: → evaluate
print_prefix
print_postfix
type_check

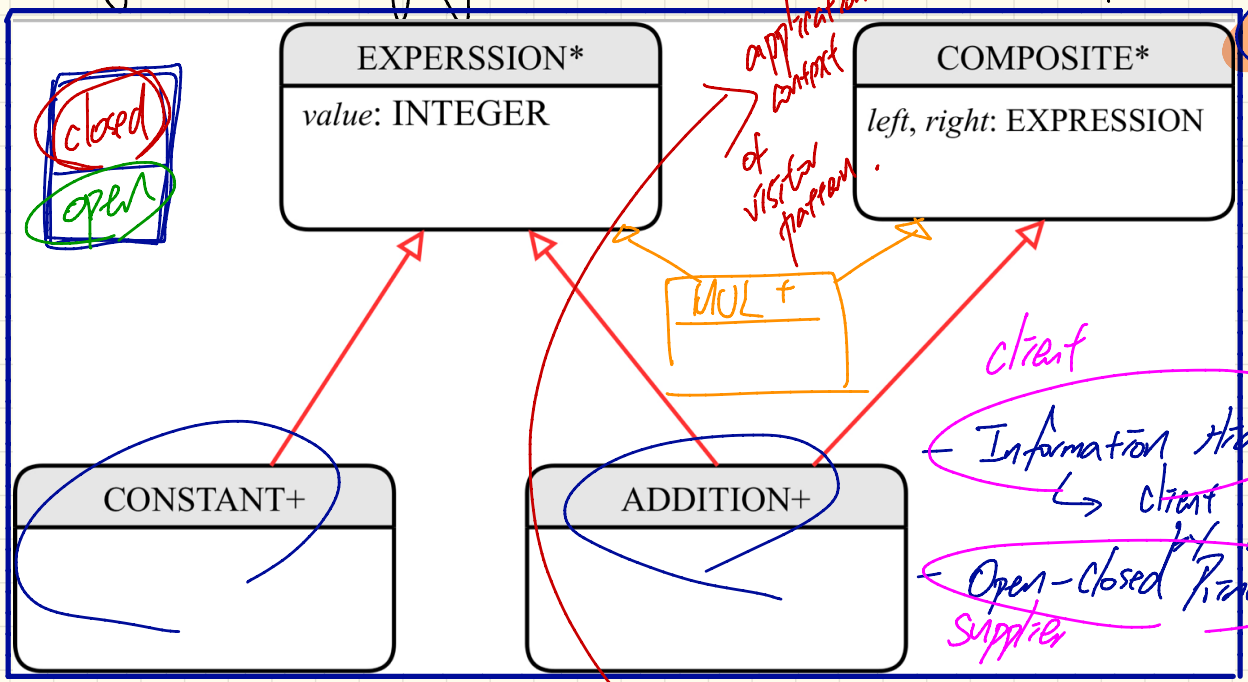
Operations

a 341 + 2

a. evaluate 343

a. print_prefix + 341 2

Design of a Language Application : Open-Closed Principle



Structure of the expression language

client
- Information Hiding
  - ↳ client not affected consciously chosen design decision
- Open-closed Principle
  - supplier

Operations: evaluate, print_prefix, print_postfix, type_check

Operations

generate_java_code

	Structure	Operations
Alt. 1	open	closed
Alt. 2	closed	open