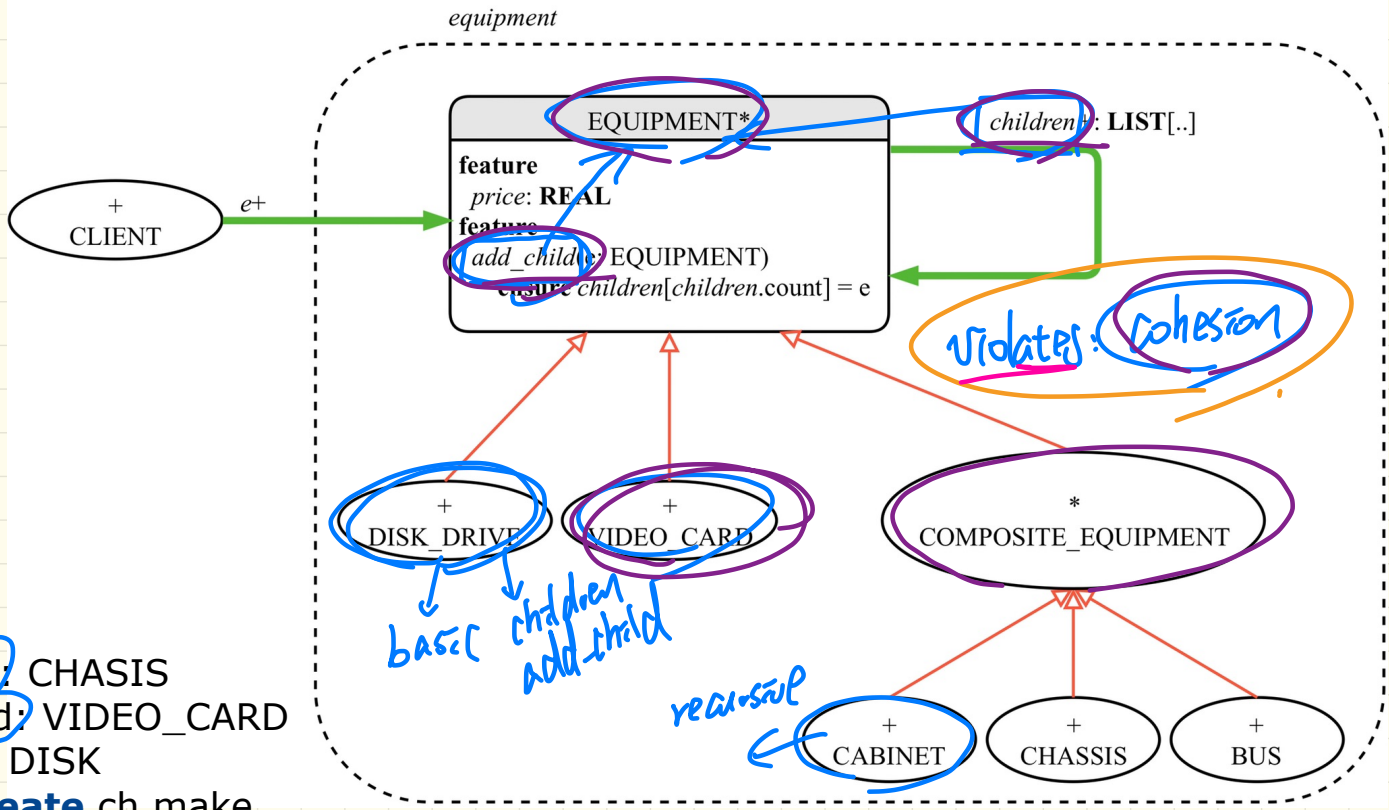
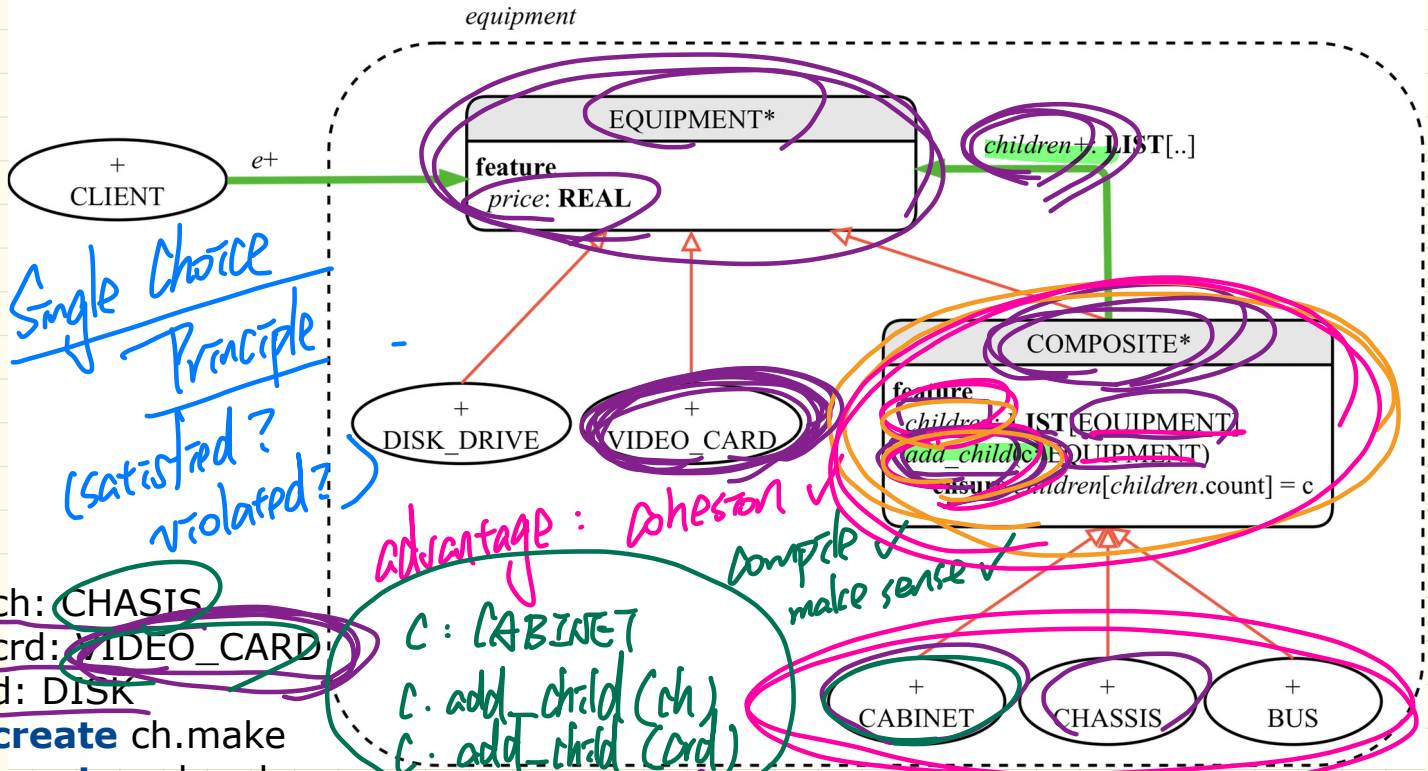
Lecture 20
Wednesday March 18



ch: CHASIS
crd: VIDEO_CARD
d: DISK
create ch.make
create crd.make
create d.make
ch.add_child(crd)
ch.add_child(d)

crd.add_child(d)

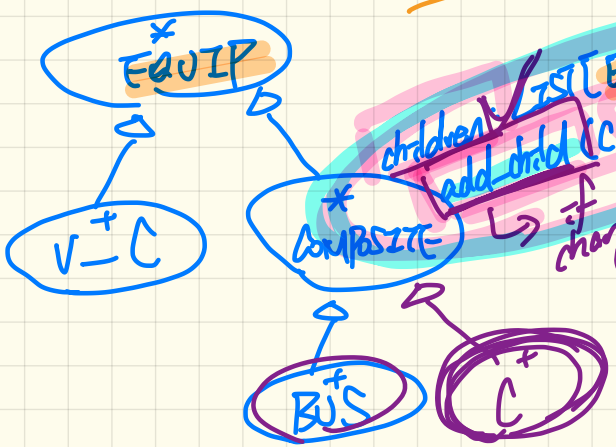
First Design Attempt


equipment
EQUIPMENT*
feature
price: REAL
children+: LIST[..]
+
CLIENT
e+
COMPOSITE*
+
DISK_DRIVE
+
VIDEO_CARD
+
CABINET
+
CHASSIS
+
BUS
Single Choice Principle
(satisfied? violated?)
advantage : cohesion ✓
compile ✓
make sense ✓
c : CABINET
c.add_child(ch)
c.add_child(crd)
ch: CHASIS
crd: VIDEO_CARD
d: DISK
create ch.make
create crd.make
create d.make
ch.add_child(crd)
ch.add_child(d)
crd.add_child(d)
not compiling
:(add_child is a feature applicable to descendants of COMPOSITE
Second Design Attempt

EQUIPMENT

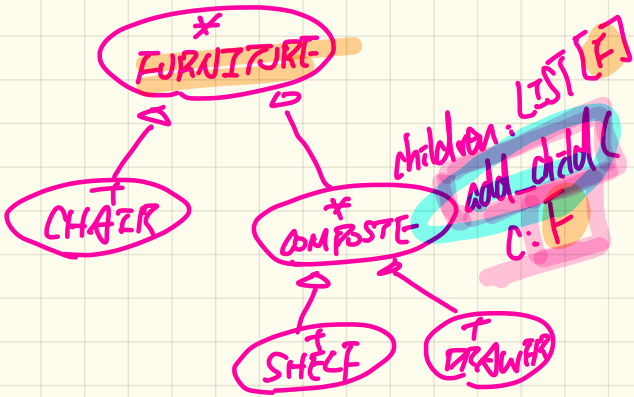
Single Choice Principle

violation ⇒ changes take multiple places to undo.

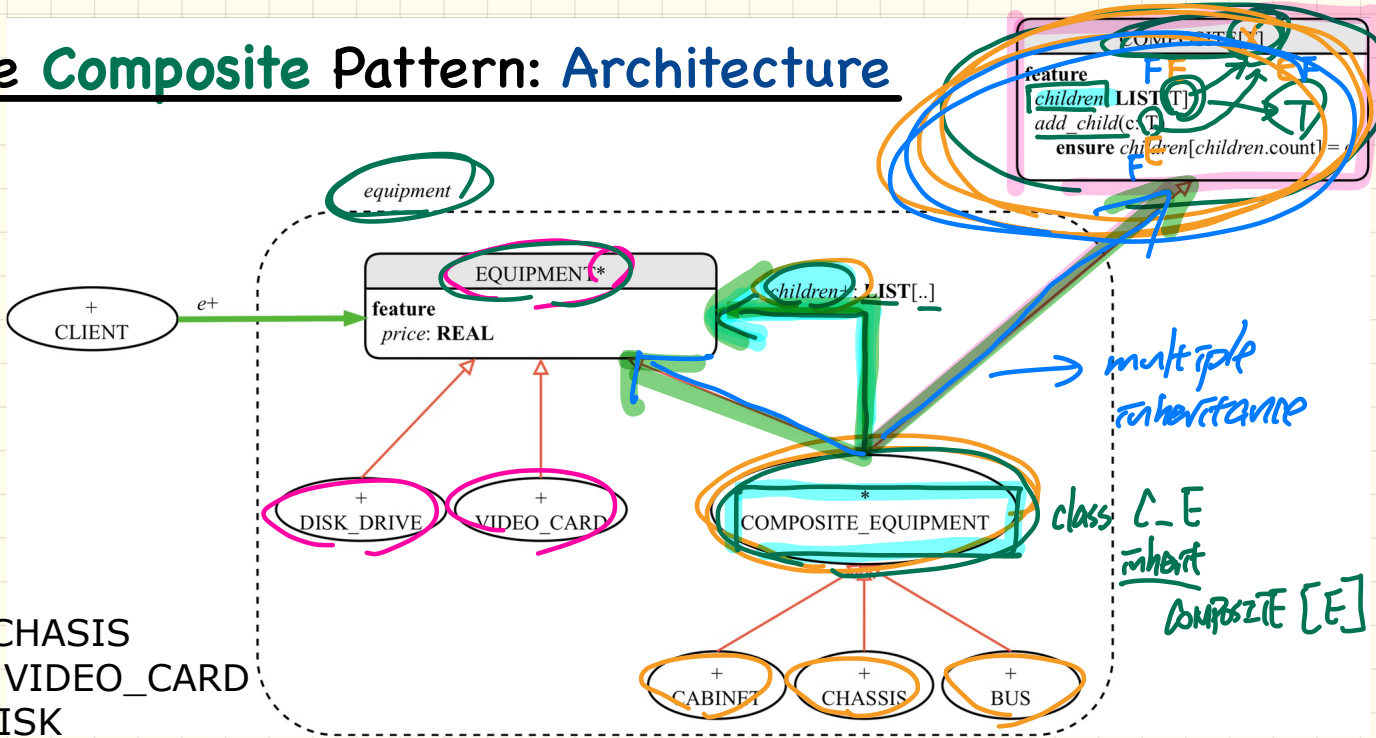


change:
change add_child
s.t. insert it to the end of the list.

FURNITURE



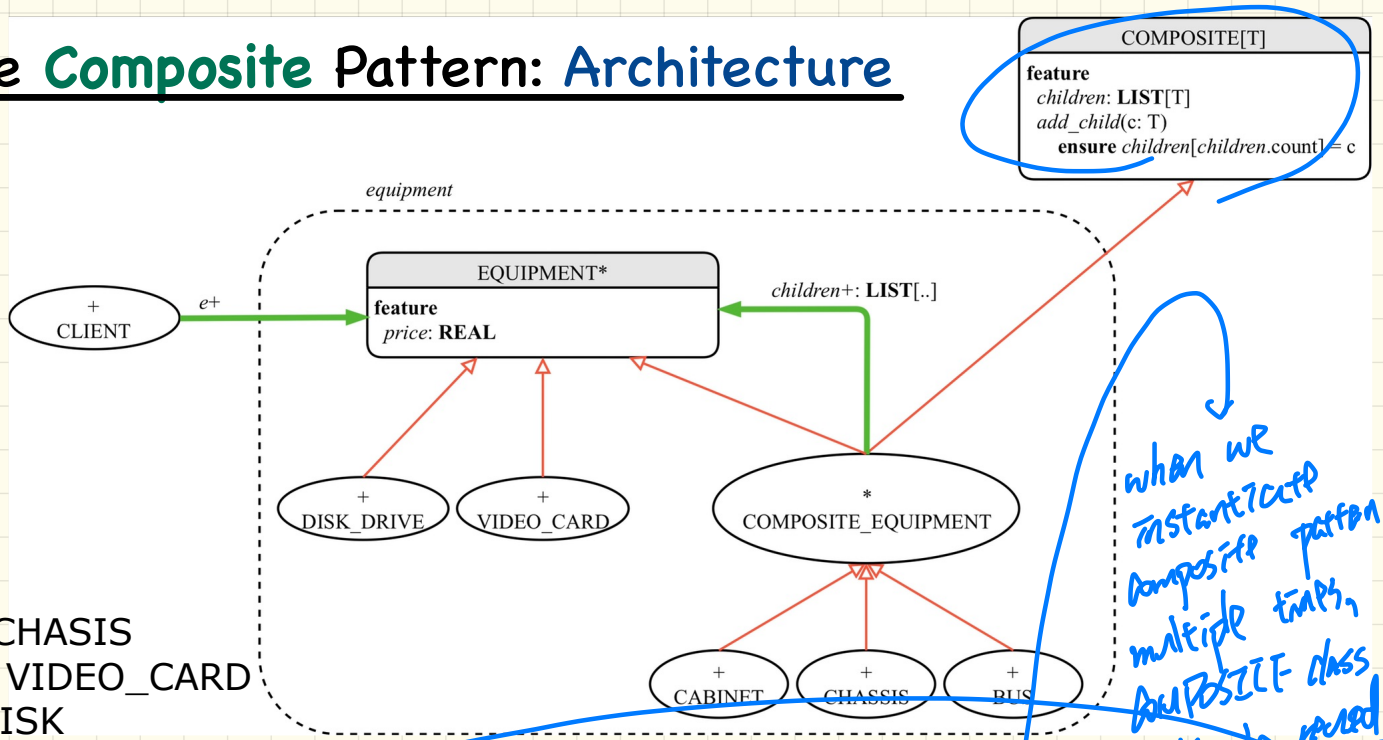
The Composite Pattern: Architecture



ch: CHASIS
crd: VIDEO_CARD
d: DISK
create ch.make
create crd.make
create d.make
ch.add_child(crd)
ch.add_child(d)
crd.add_child(d)

exercise:
should this compile?

The Composite Pattern: Architecture

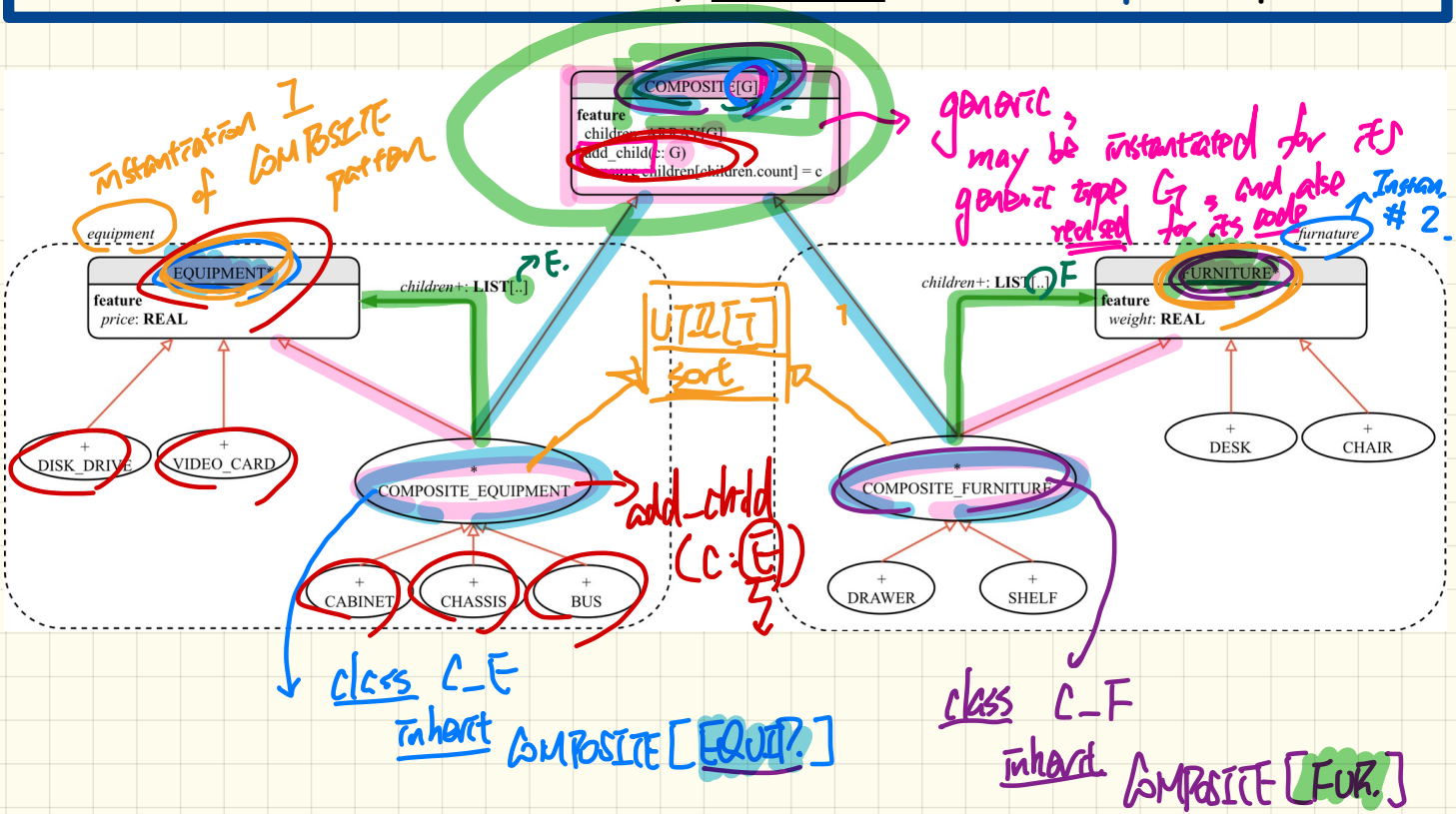


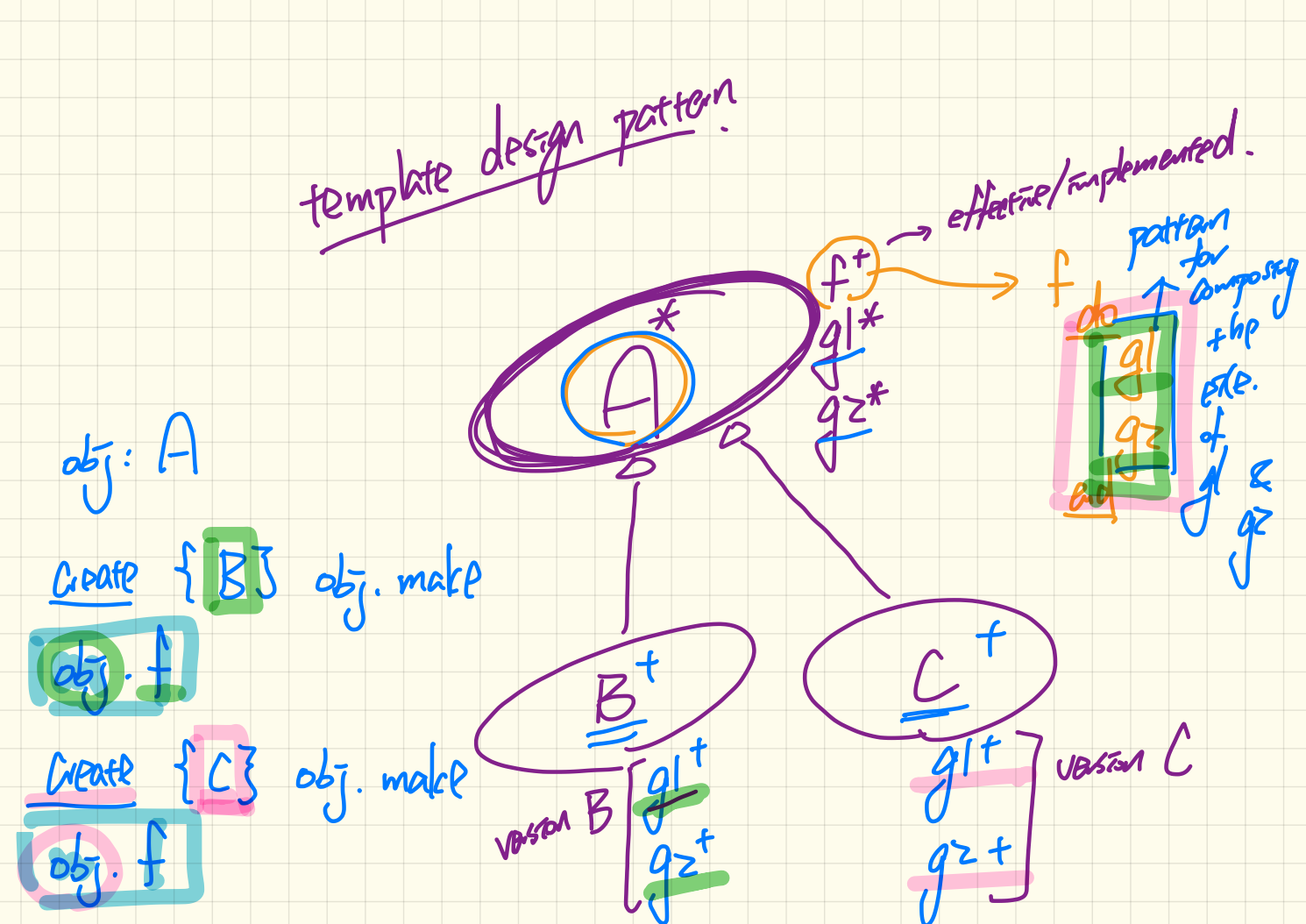
ch: CHASIS
crd: VIDEO_CARD
d: DISK
create ch.make
create crd.make
create d.make
ch.add_child(crd)
ch.add_child(d)
crd.add_child(d)

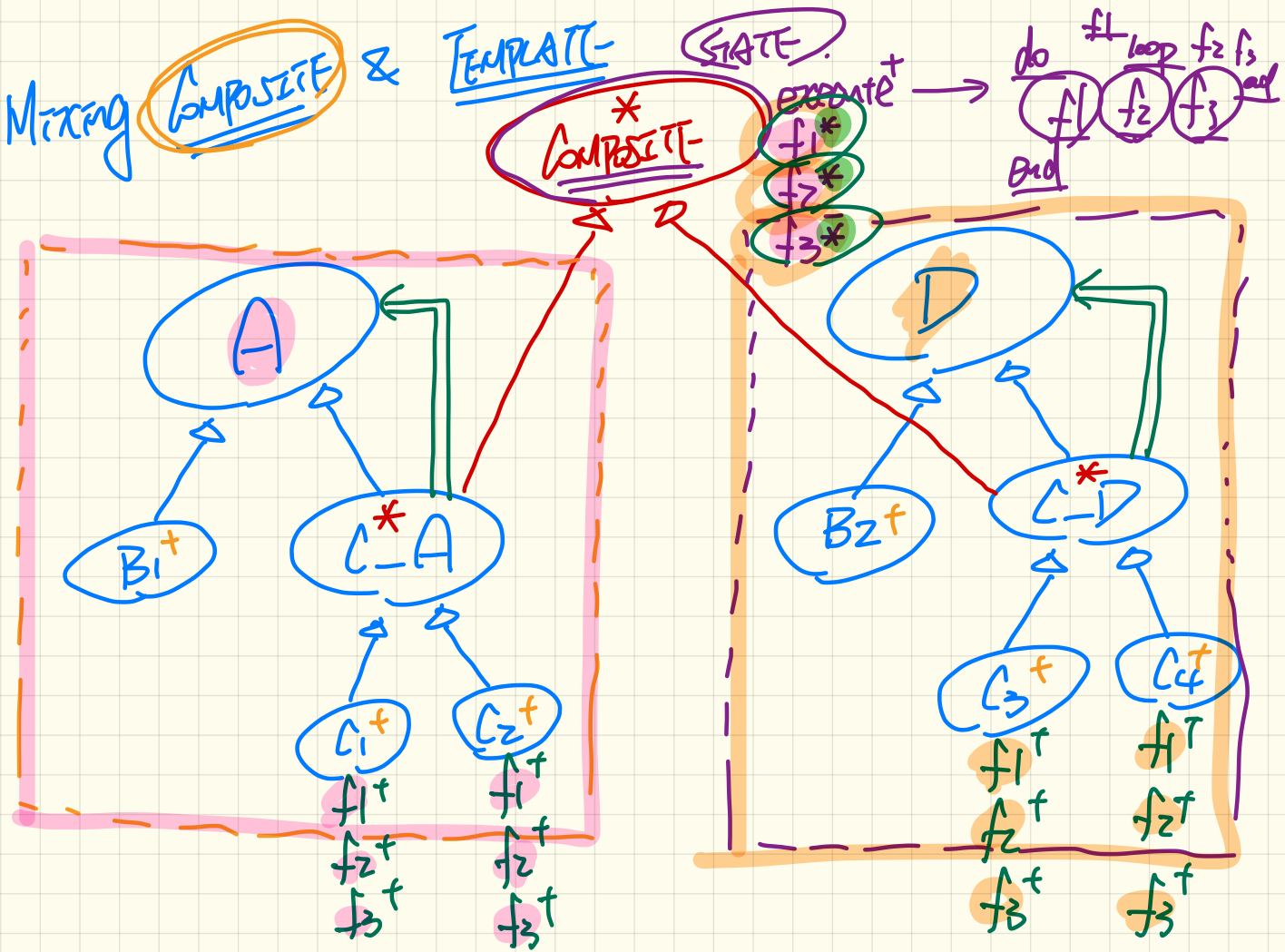
Why is COMPOSITE a separate class?

The Composite Pattern: Architecture

COMPOSITE class is reusable by instances of the composite pattern.




template design pattern.
f+
effective/implemented.
f
pattern for composing the exe. of g1 & g2
do
g1
g2
end
A
*
g1*
g2*
obj: A
Create {B} obj. make
obj. f
Create {C} obj. make
obj. f
B+
C+
version B
g1+
g2+
version C
g1+
g2+


Mixing
COMPOSITE
&
TEMPLATE-
STATE
COMPOSITE
execute
do
loop
end
A
B1
C_A
C1
C2
D
B2
C_D
C3
C4

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
```

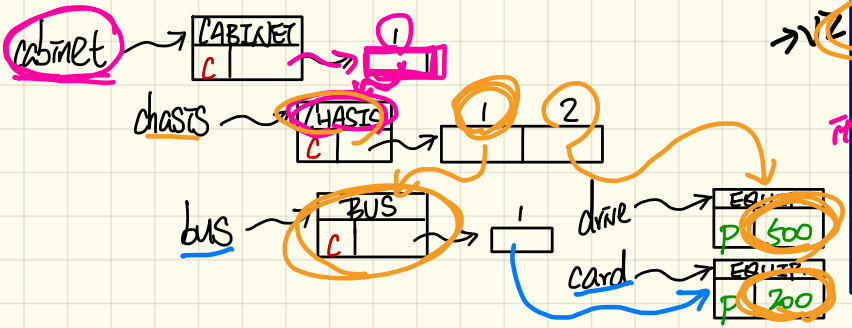
```
class
  COMPOSITE_EQUIPMENT
inherit
  EQUIPMENT
  COMPOSITE [EQUIPMENT]
create
  make
feature
  make (n: STRING)
    do name := n ; create children.make end
  price: REAL -- price is a query
    -- Sum the net prices of all sub-equipments
    do
      across
        children as cursor
      loop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




```
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 BUS sub-equip
    do
      across
        children as cursor
      loop
        Result := Result + cursor.item.price
      end
    end
end
```

DT? BUS

DT? CHASSIS

{CHASSIS}.price