Wednesday January 16

Lecture 4

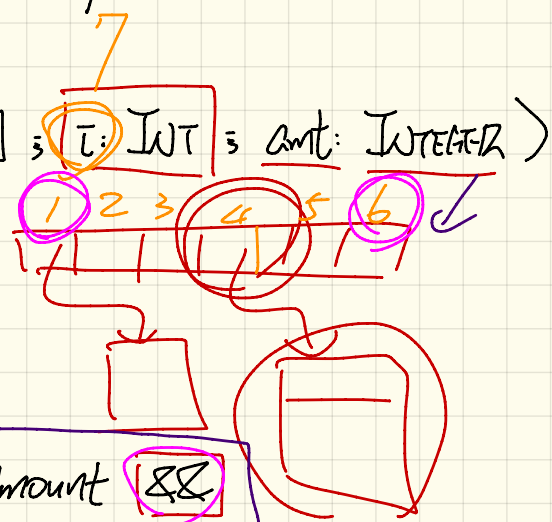
- Lab 2 posted
- Lab 1

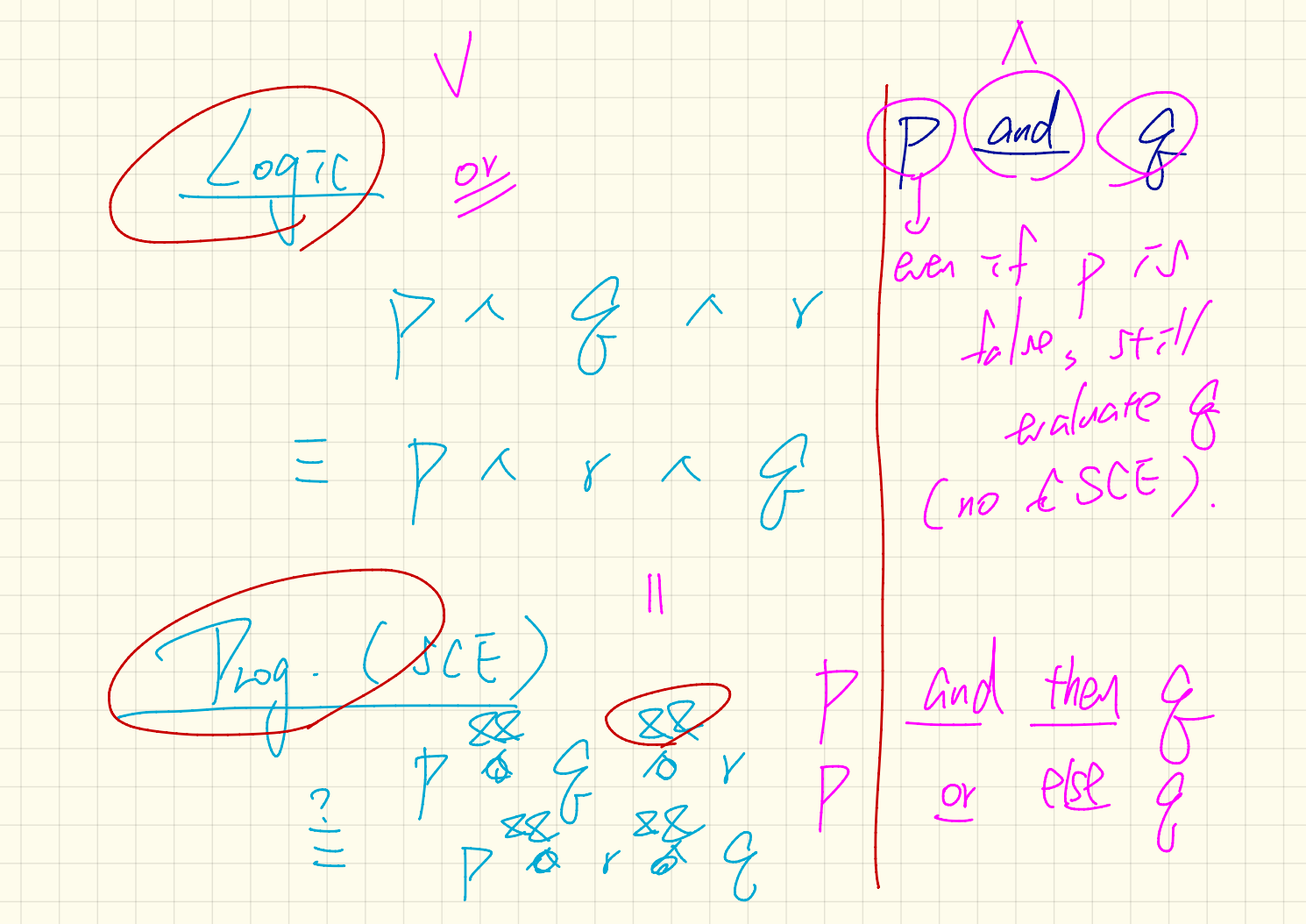
Office Hours

W/F 3pm ~ 5pm

# Short-Circuit Operators vs. Logical Operator

withdraw ( accounts : ARRAY[ACCOUNT] ; i: INT ; amt: INTEGER )




Logic
∨
or
p ∧ q ∧ r
≡ p ∧ r ∧ q
Prog. (SCE)
&&
||
p && q || r
≡ ?
p || r && q
∧
p and q
even if p is false, still evaluate q (no ∧ SCE).
p and then q
p or else q

a : ARRAY[STRING]

rebase

Create a.make_empty

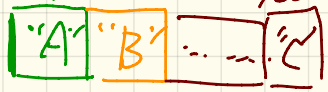
a.lower 1
a.upper 0

a.force("A", a.upper + 1)

a.lower 1
a.upper 1

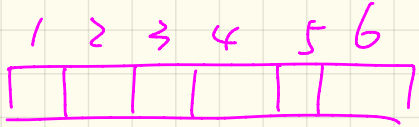
a.force("B", a.upper + 1)

a.lower 1
a.upper 2




[a.lower, a.upper]

a.upper - a.lower + 1

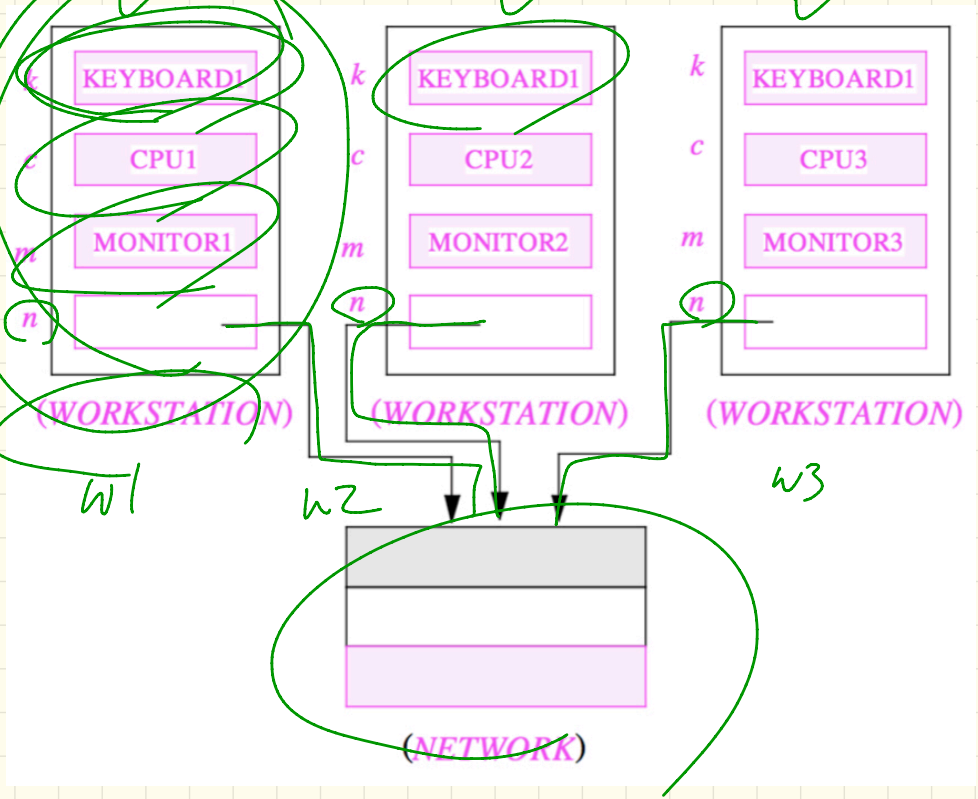



[1, 6]

6 - 1 + 1

a.force("C", 100)

# Modelling: Aggregation vs. Composition



# Expanded Type for Composition

```
class KEYBOARD ... end class CPU ... end
class MONITOR ... end class NETWORK ... end
class WORKSTATION
  k: expanded KEYBOARD
  c: expanded CPU
  m: expanded MONITOR
  n: NETWORK
end
```

k cannot be shared.

```
expanded class KEYBOARD ... end
expanded class CPU ... end
expanded class MONITOR ... end
class NETWORK ... end
class WORKSTATION
  k: KEYBOARD
  c: CPU
  m: MONITOR
  n: NETWORK
end
```

# Use of Expanded Type

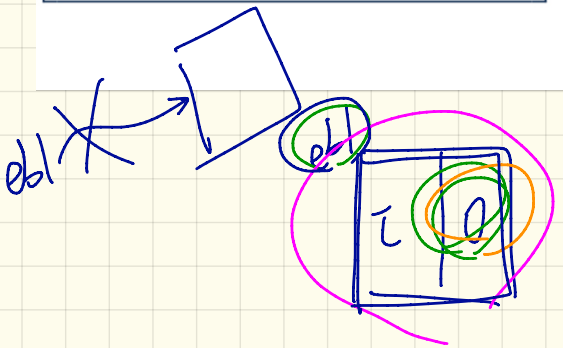
eb1 == eb2

```
expanded class
  B
feature
  change_i (ni: INTEGER)
    do
      i := ni
    end
feature
  i: INTEGER
end
```

```
1  test_expanded: BOOLEAN
2    local
3      eb1, eb2: B
4    do
5      Result := eb1.i = 0 and eb2.i = 0
6      check Result end
7      Result := eb1 = eb2
8      check Result end
9      eb2.change_i (15)
10     Result := eb1.i = 0 and eb2.i = 15
11     check Result end
12     Result := eb1 /= eb2
13     check Result end
14   end
```

comp. contents

eb1.i vs. eb2.i



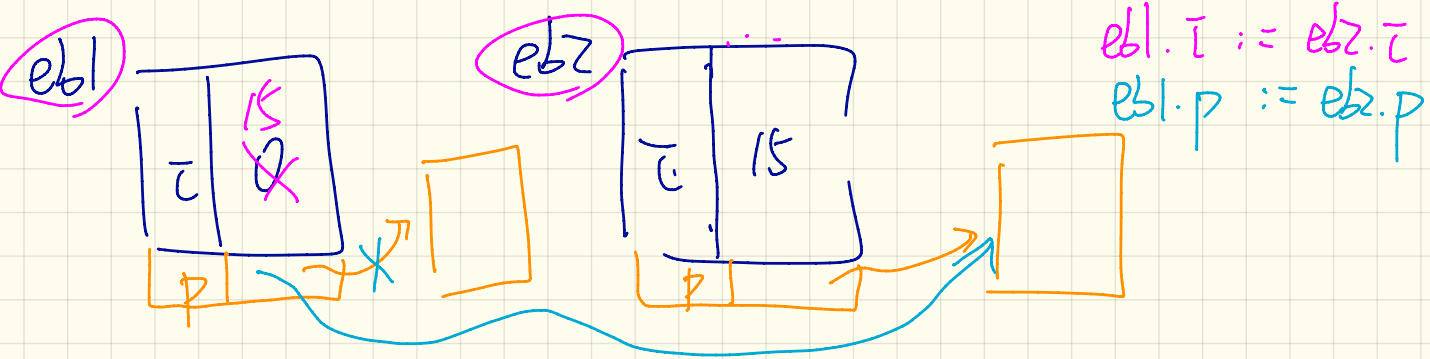

obj1 = obj2

↳ obj1, obj2 Ref. T. → compare addresses.

↳ obj1, obj2 Exp. T → compare contents

```
expanded class
  B
feature
  change_i (ni: INTEGER)
    do
      i := ni
    end
feature
  i: INTEGER
end
```

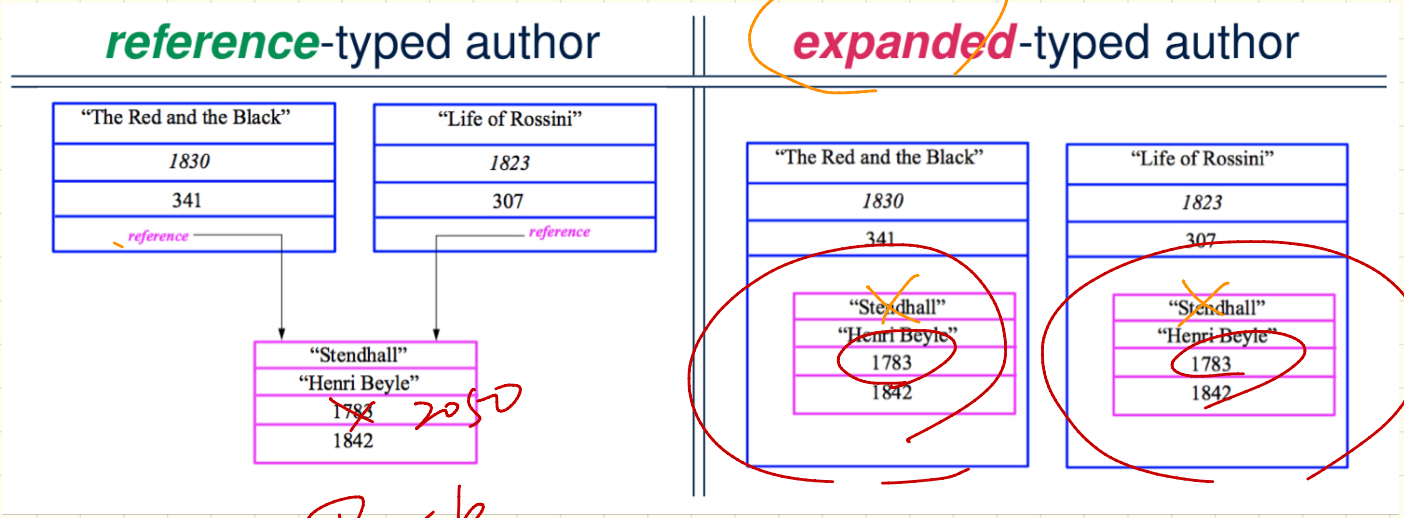
```
1   test_expanded: BOOLEAN
2    local
3     eb1, eb2: B
4    do
5     Result := eb1.i = 0 and eb2.i = 0
6     check Result end
7     Result := eb1 = eb2
8     check Result end
9     eb2.change_i (15)
10    Result := eb1.i = 0 and eb2.i = 15
11    check Result end
12    Result := eb1 /= eb2
13    check Result end
14   end
```



```
class B
  .  -
end
```

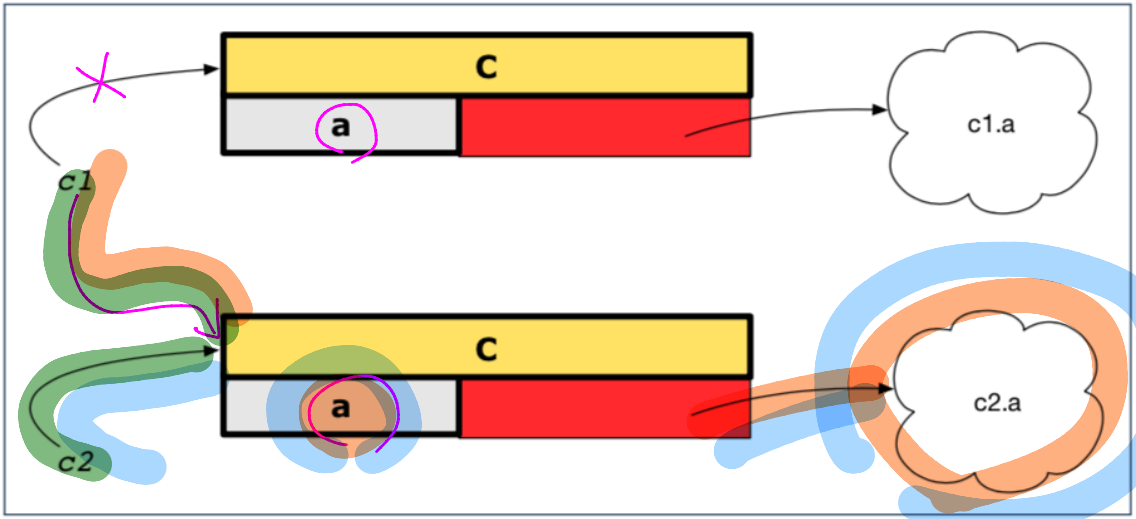
```
local
    v1: B
    v2: expanded B
do
    v1 (=) v2
```

# Reference or Expanded Type




Single Choice Principle

# Reference Copy : c1 := c2

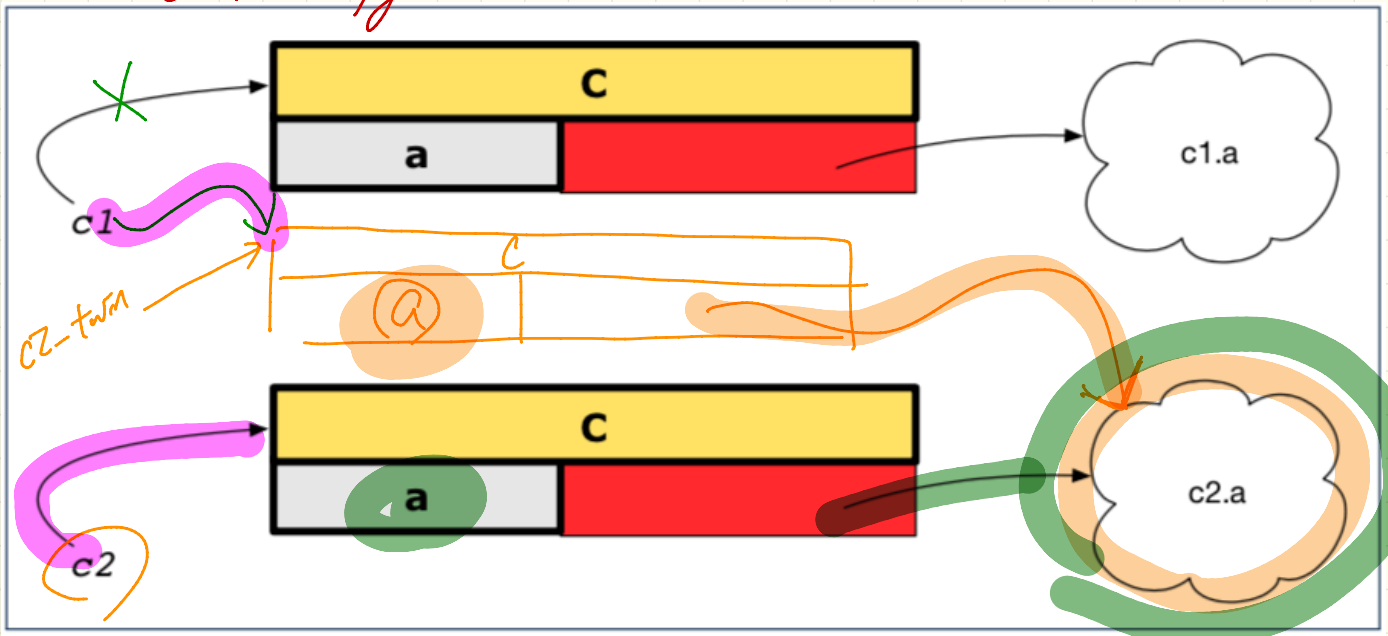



① c1 = c2 T

→ ② c1.a = c2.a T

Shallow Copy : c1 := c2.twin

1st-level copy

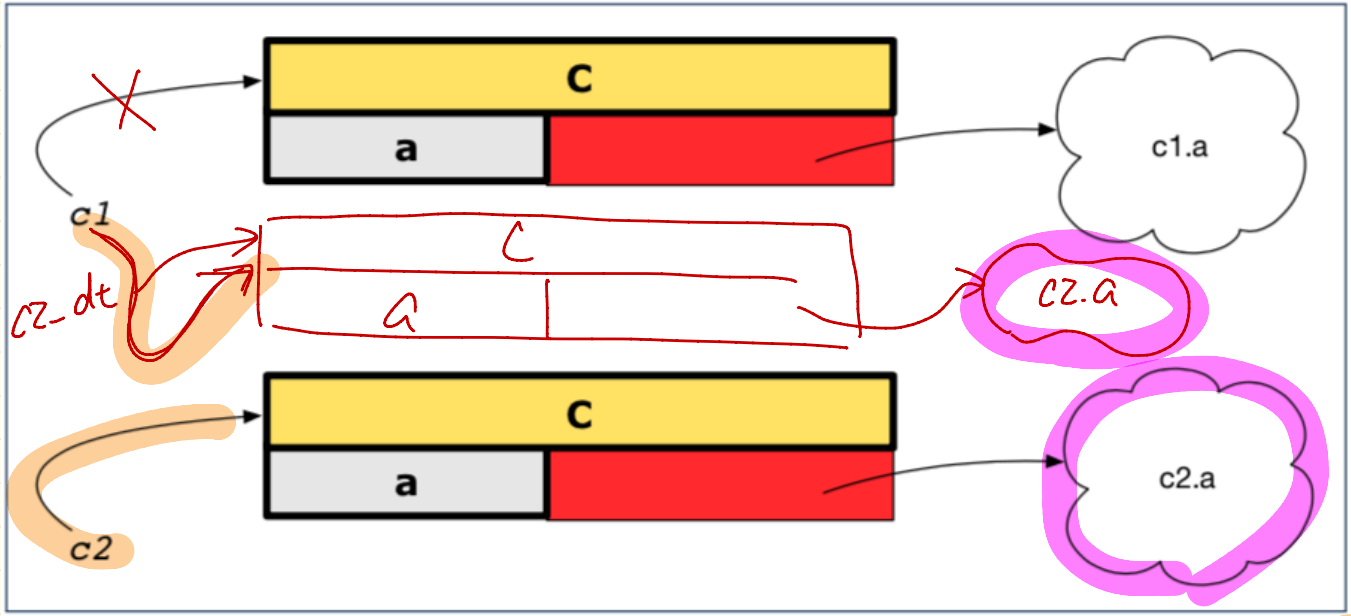



c2_twin.a := c2.a

① c1 = c2 F

② c1.a = c2.a T

Deep Copy : c1 := c2.deep_twin




c2_dt.a := c2.a.deep_twin

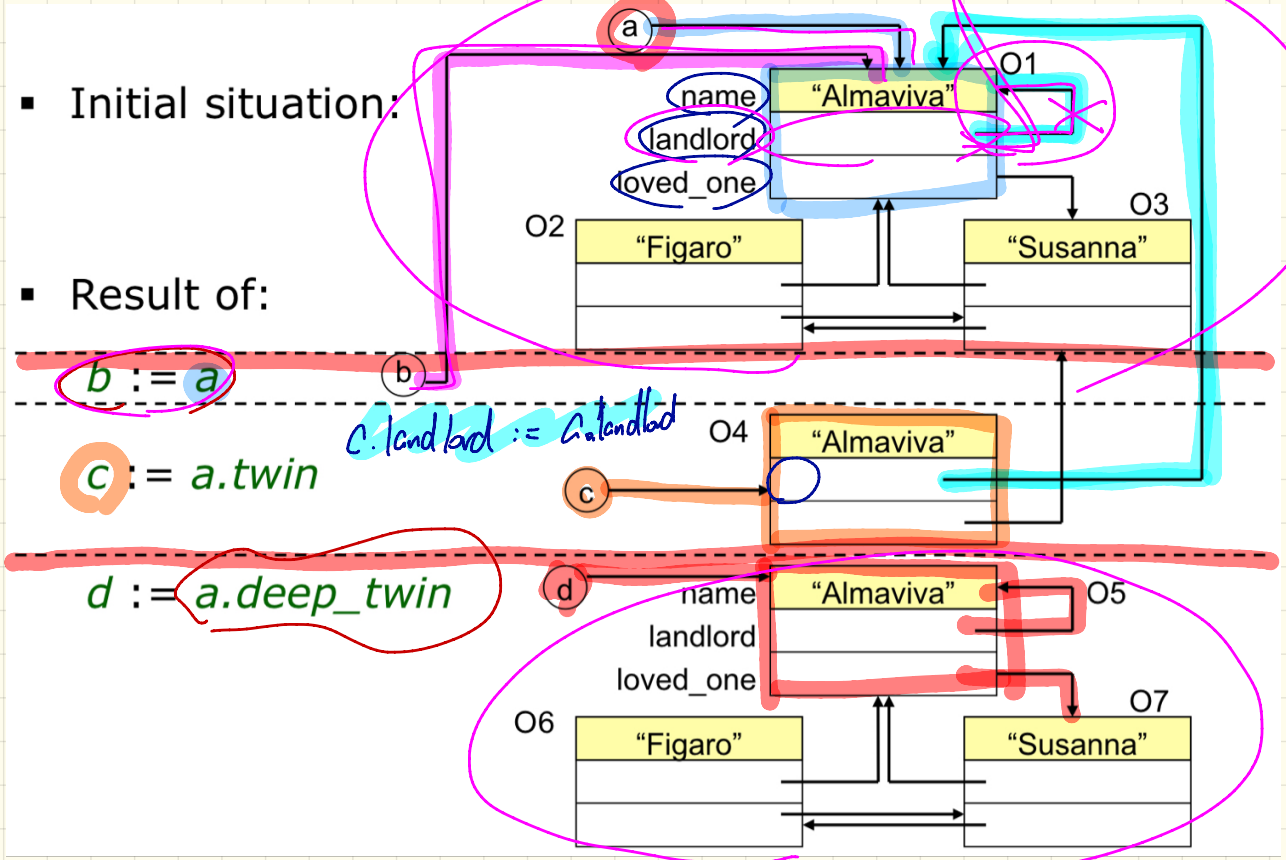
① c1 = c2 F

② c1.a = c2.a

# Ref. vs. Shallow vs. Deep Copies

- Initial situation:

- Result of:



# Copying Collection Objects : Reference Copy & Make Changes

```
1  old_imp := imp
2  Result := old_imp = imp  -- Result = true
3  imp[2] := "Jim"
4  Result :=
5    across 1 |..| imp.count as j
6    all imp [j.item] ~ old_imp [j.item]
7    end -- Result = true
```