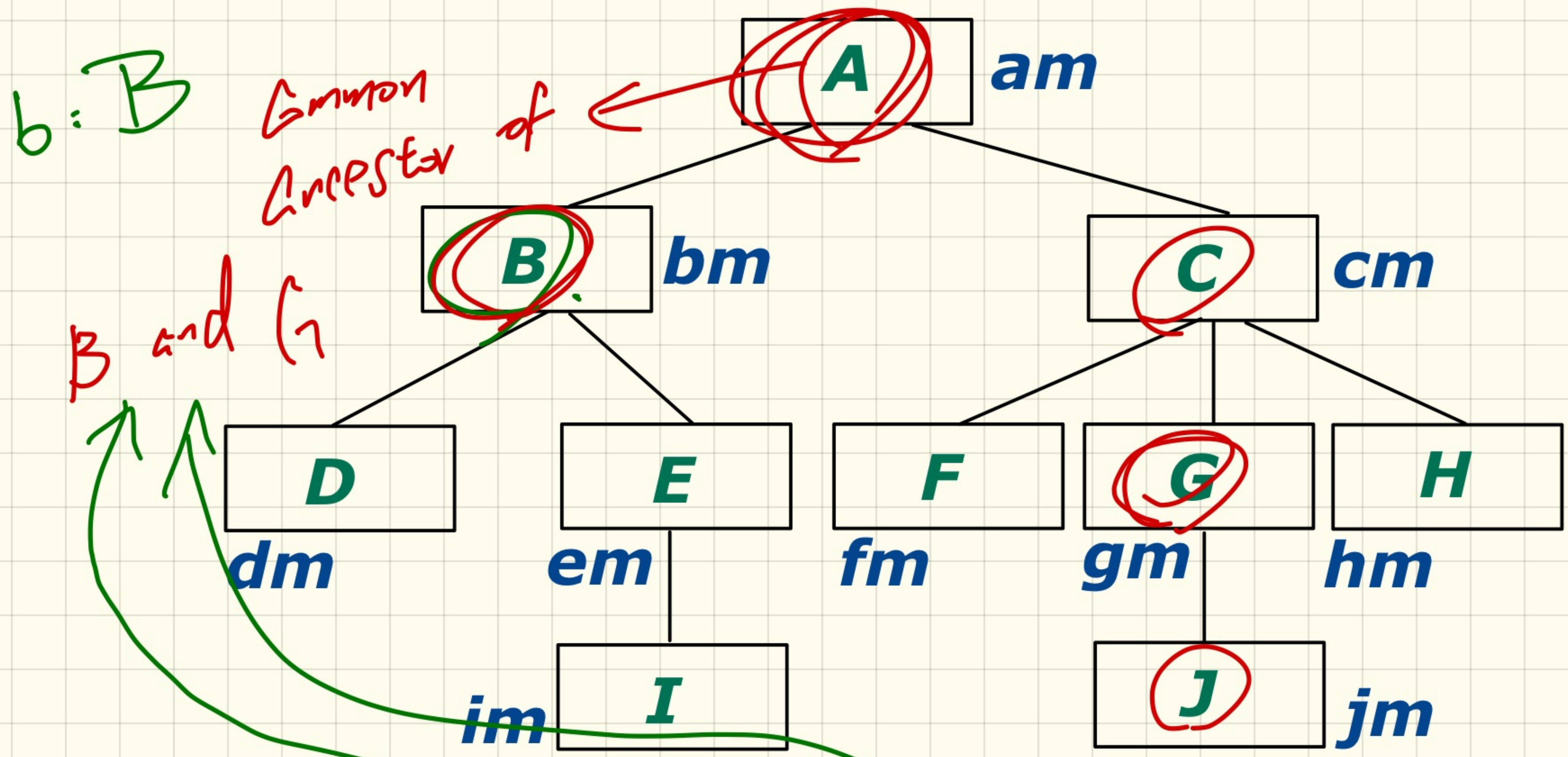


LECTURE 11

THURSDAY OCTOBER 10

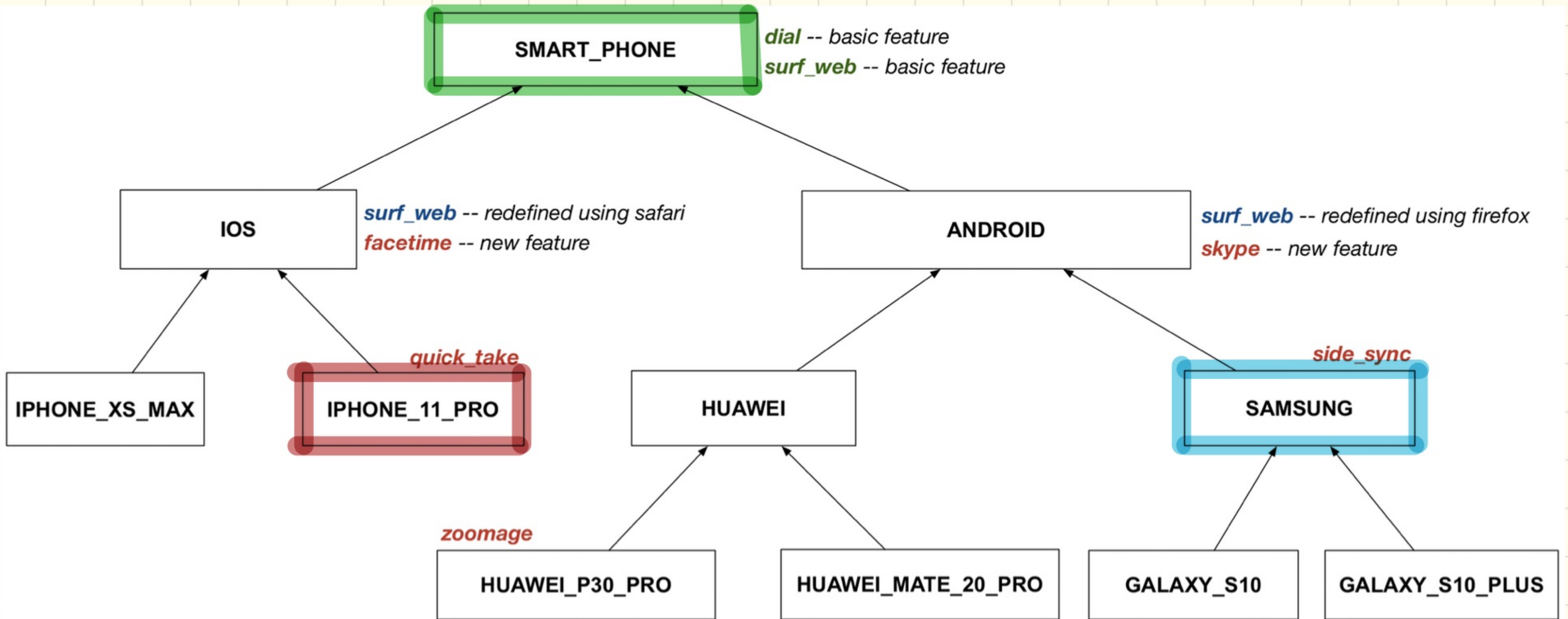
Inheritance Forms a Type Hierarchy (1)

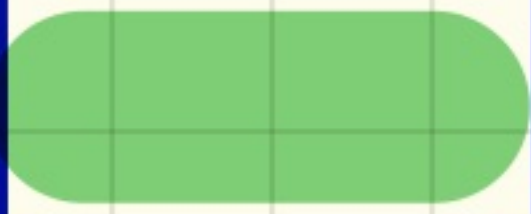
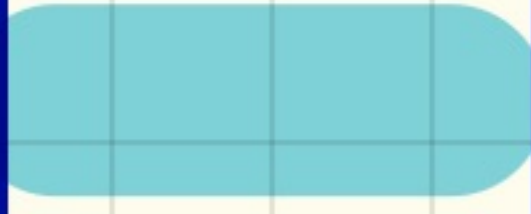
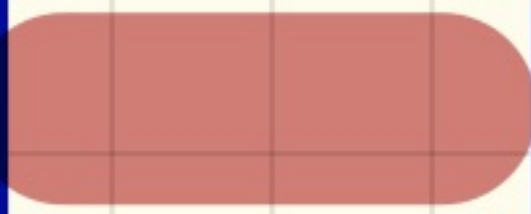


	ancestors	expectations	descendants
B	B, A	im, bm	B D E I
G	G, C, A	gm, cm, am	
J	J, G, C, A	jm, gm, cm, am	

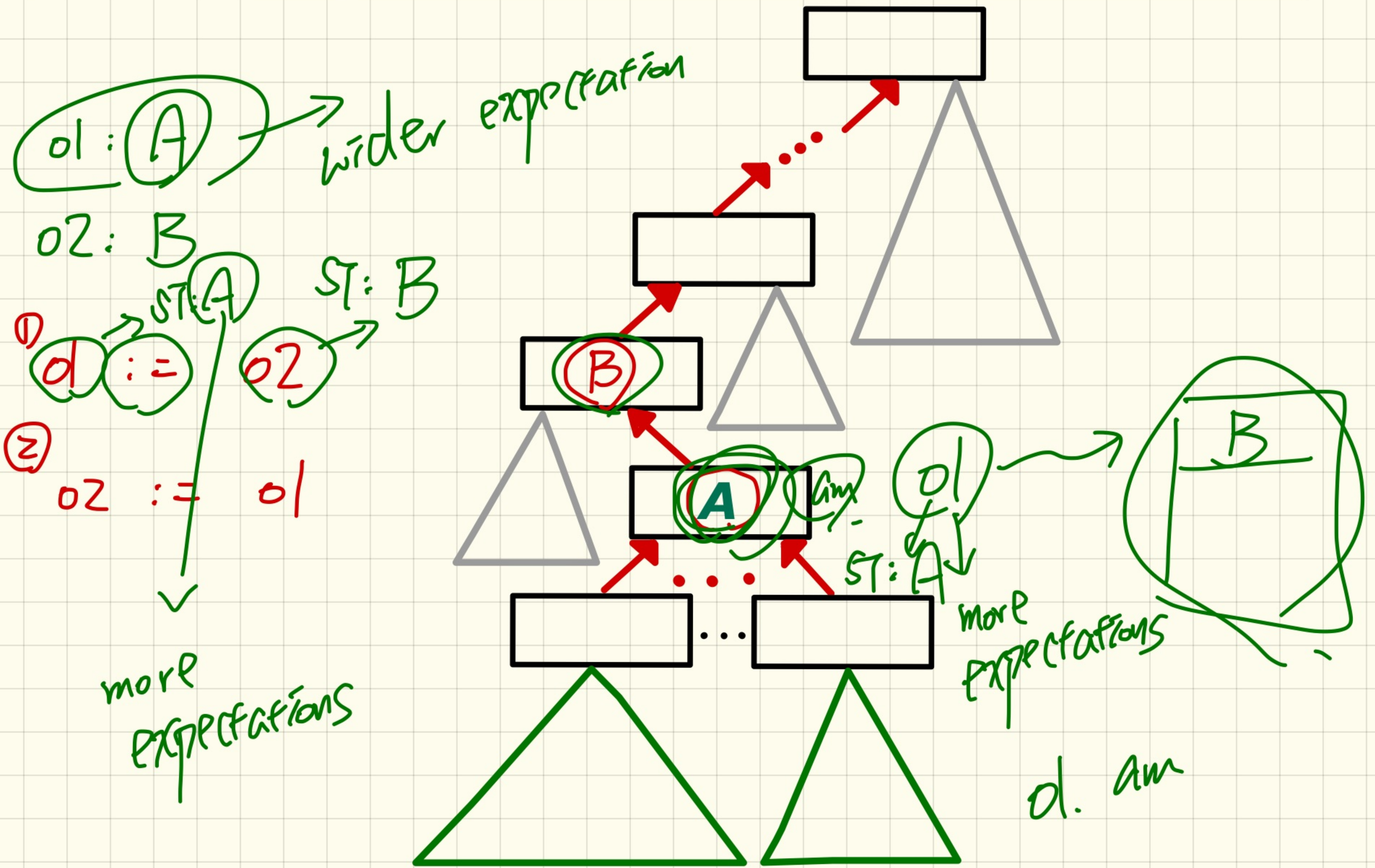
∵ G is an ancestor of J
 $E(G) \subseteq E(J)$

Inheritance Forms a Type Hierarchy (2)



	ancestors	expectations	descendants
			
			
			

Ancestors, Expectations, Descendants, and Code Reuse



ol: A → wider expectation

o2: B
ST: A →
ST: B →

① ol := o2

② o2 := ol

more expectations

more expectations

d. am



ol: B
create {B} ol. make

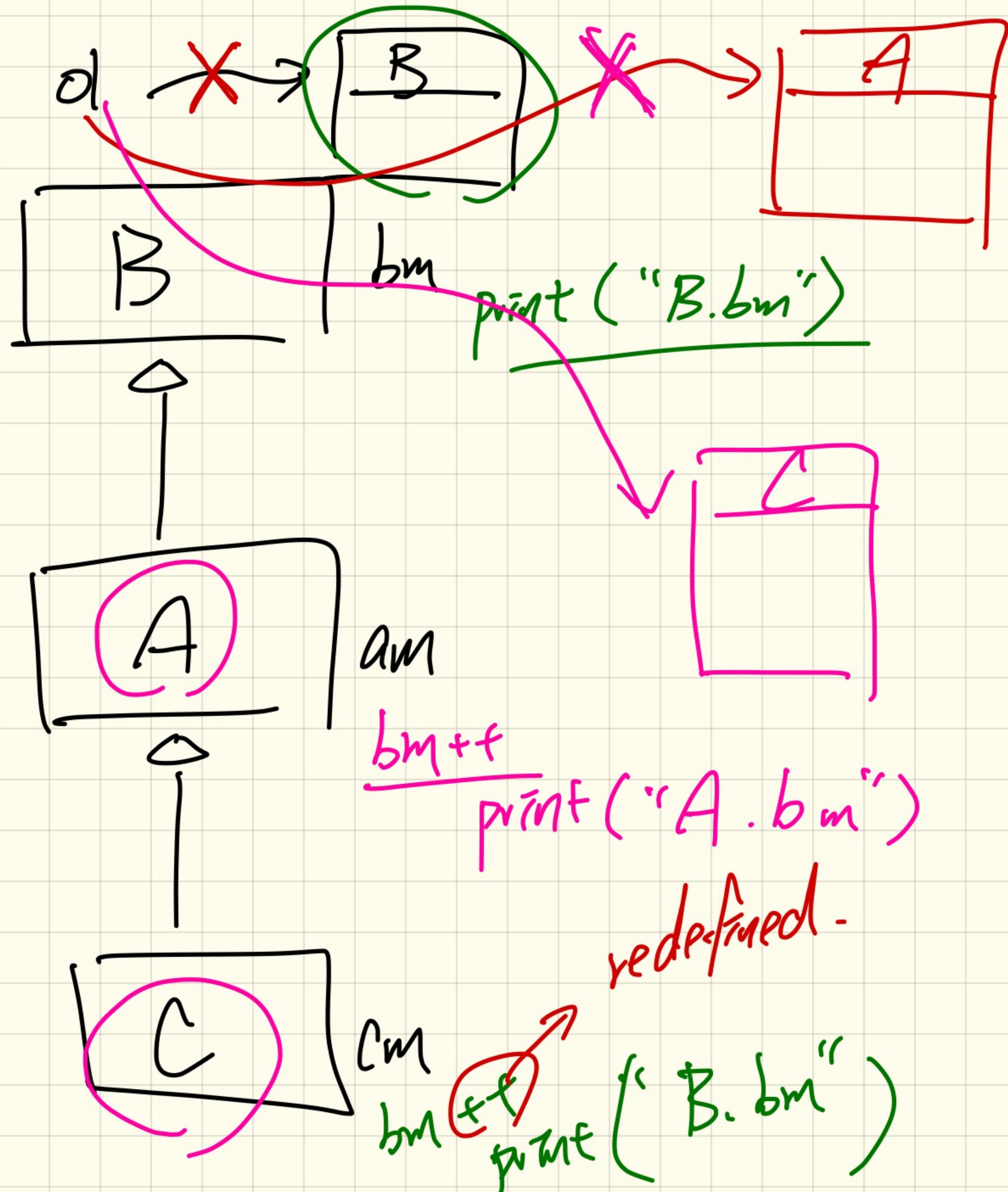
ol. bm → B.bm

create {A} ol. make

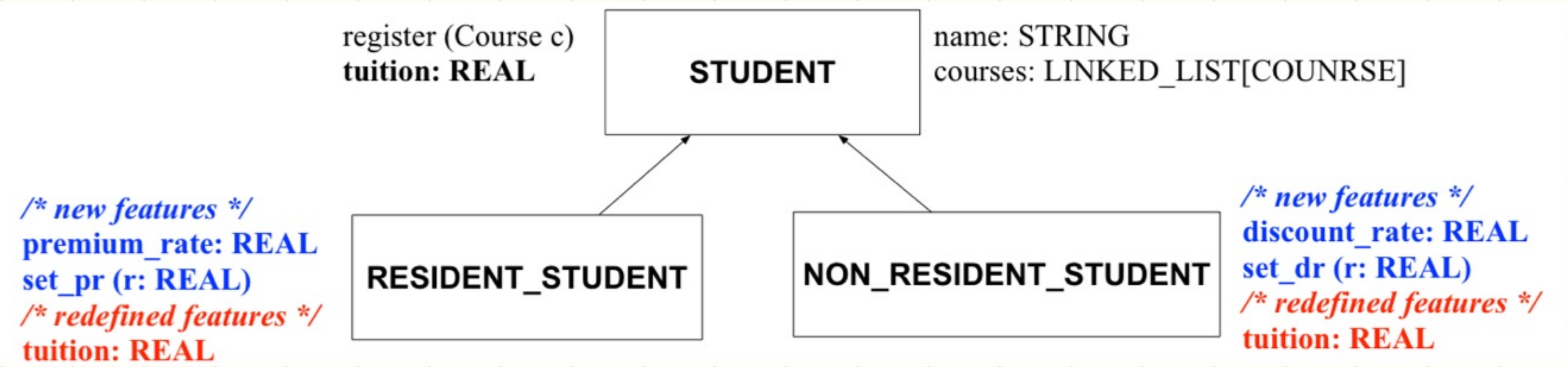
ol. bm → A.bm

create {C} ol. make

ol. bm → A.bm



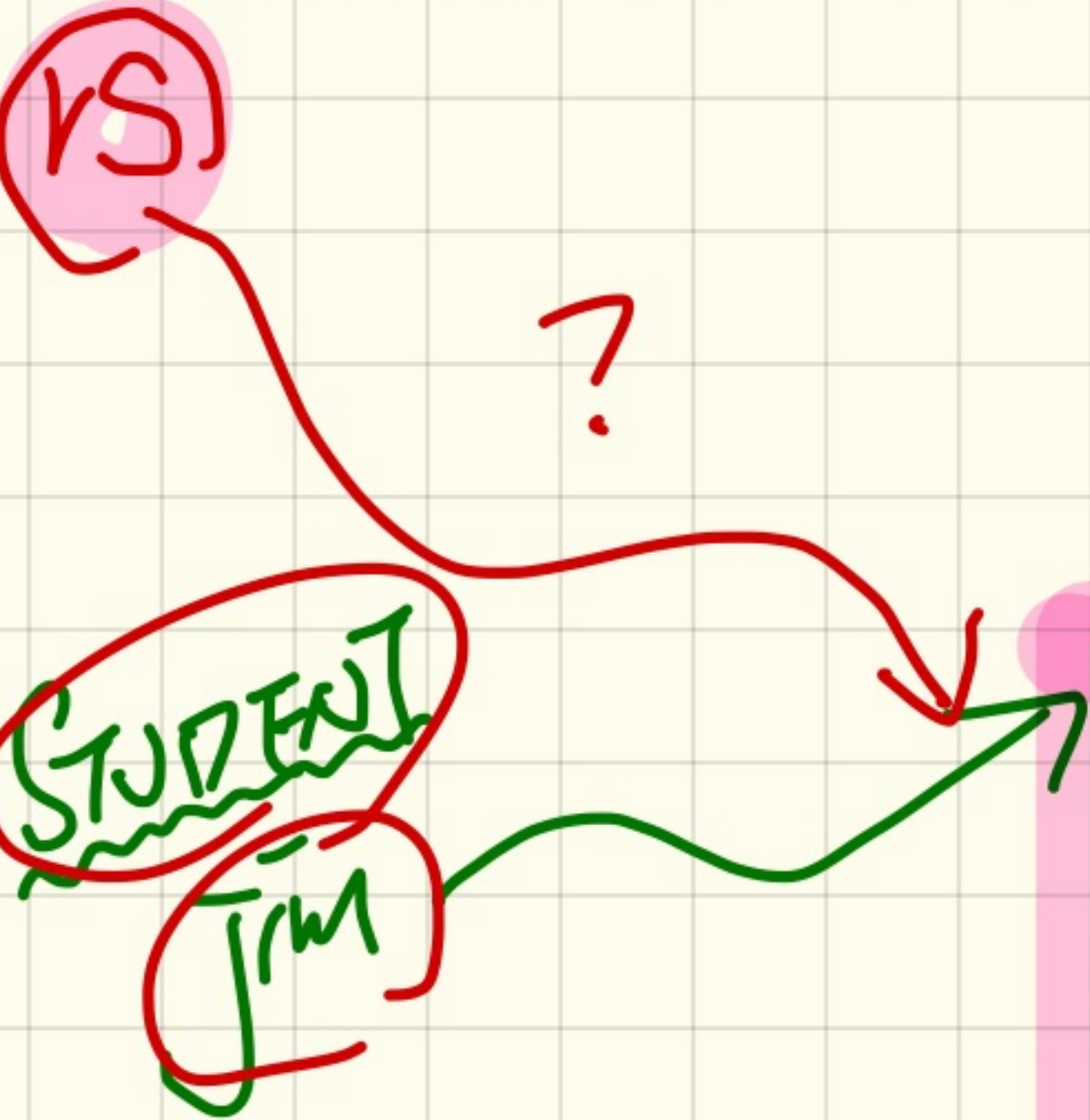
Type Cast: Motivation



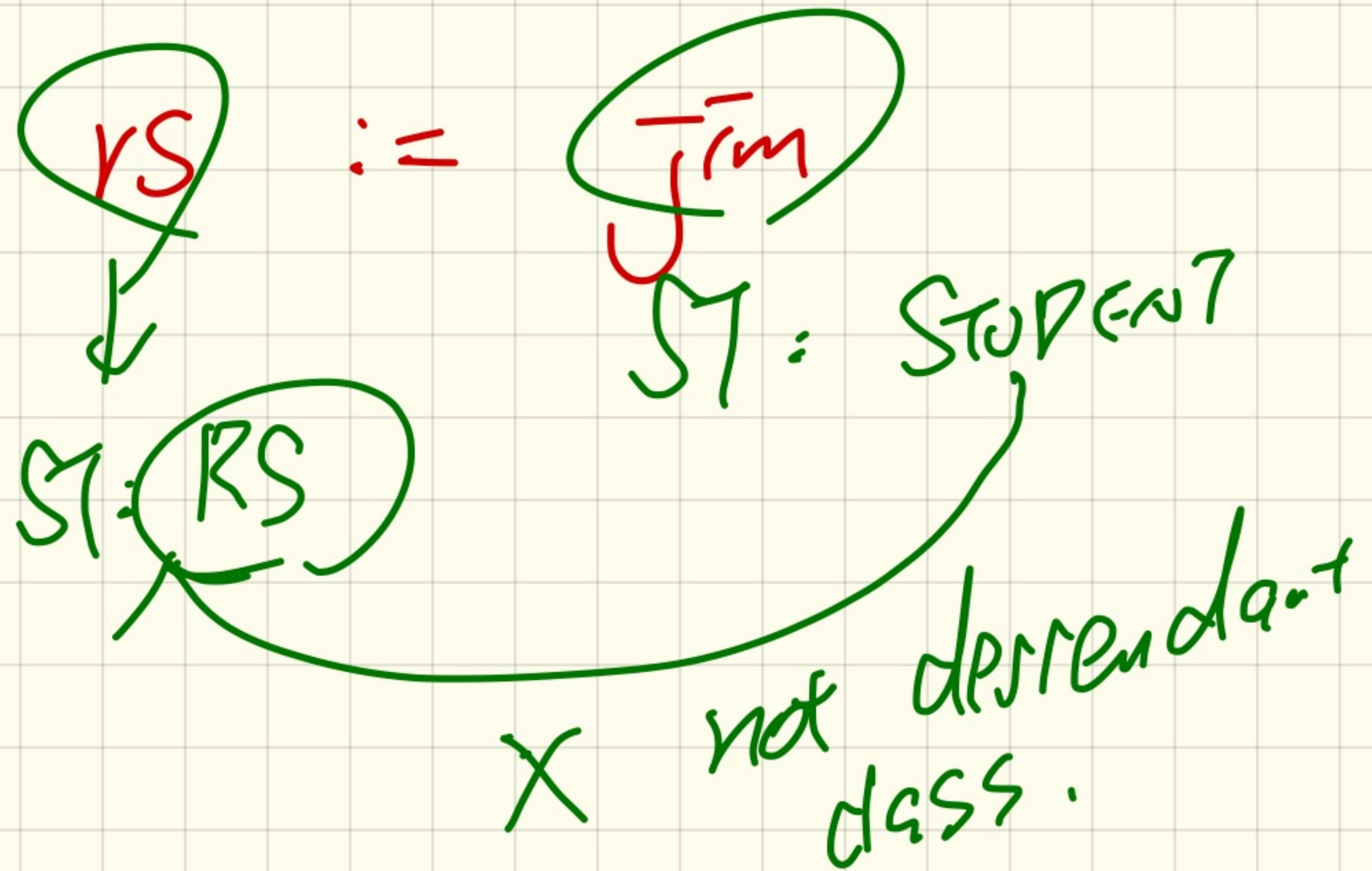
```

1 local jim: STUDENT, rs: RESIDENT_STUDENT
2 do create {RESIDENT_STUDENT} jim.make ("J. Davis")
3   rs := jim
4   rs.setPremiumRate(1.5)
  
```

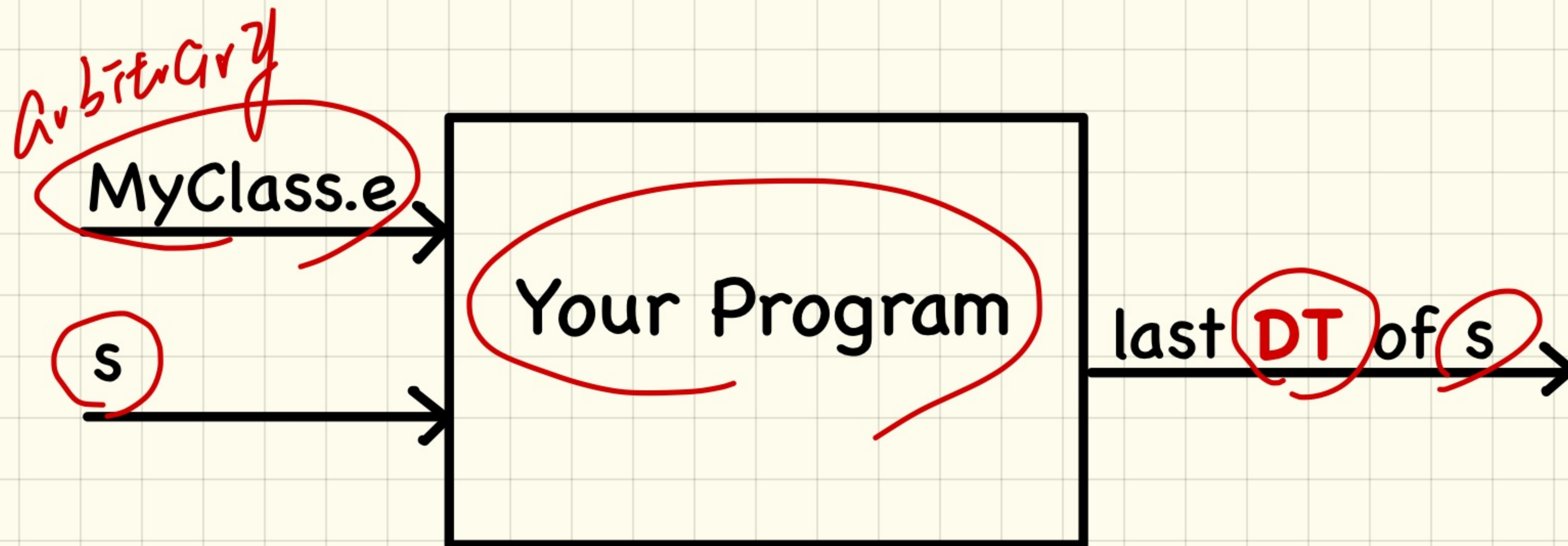
RESIDENT_STUDENT



RESIDENT_S.	
n.	
cs.	
pr.	



Inferring the DT of a Variable is Undecidable



```
class MyClass
  make
  local
    s: STUDENT
  do
    create {RESIDENT_STUDENT} s.make
  end
end
```

while (...)
 create {R-S} s.make
end

create {N-R-S} s.make

Type Cast: Syntax

```

1 check attached (RESIDENT_STUDENT) jim as rs_jim then
2   rs := rs_jim
3   rs.set_pr (1.5)
4 end
  
```

rs := rs_jim

RESIDENT STUDENT
rs_jim

STUDENT
jim

RESIDENT_S.	
n.	"J. Davies"
cs.	
pr.	1.5

1. NO NEW OBJECT
CREATED

2. ST of jim
WAS NOT MODIFIED

RS
rs



Boolean Exp.

check

attached {RS} from AS vs_jm

and

attached {NRS} alan AS nrs_alan

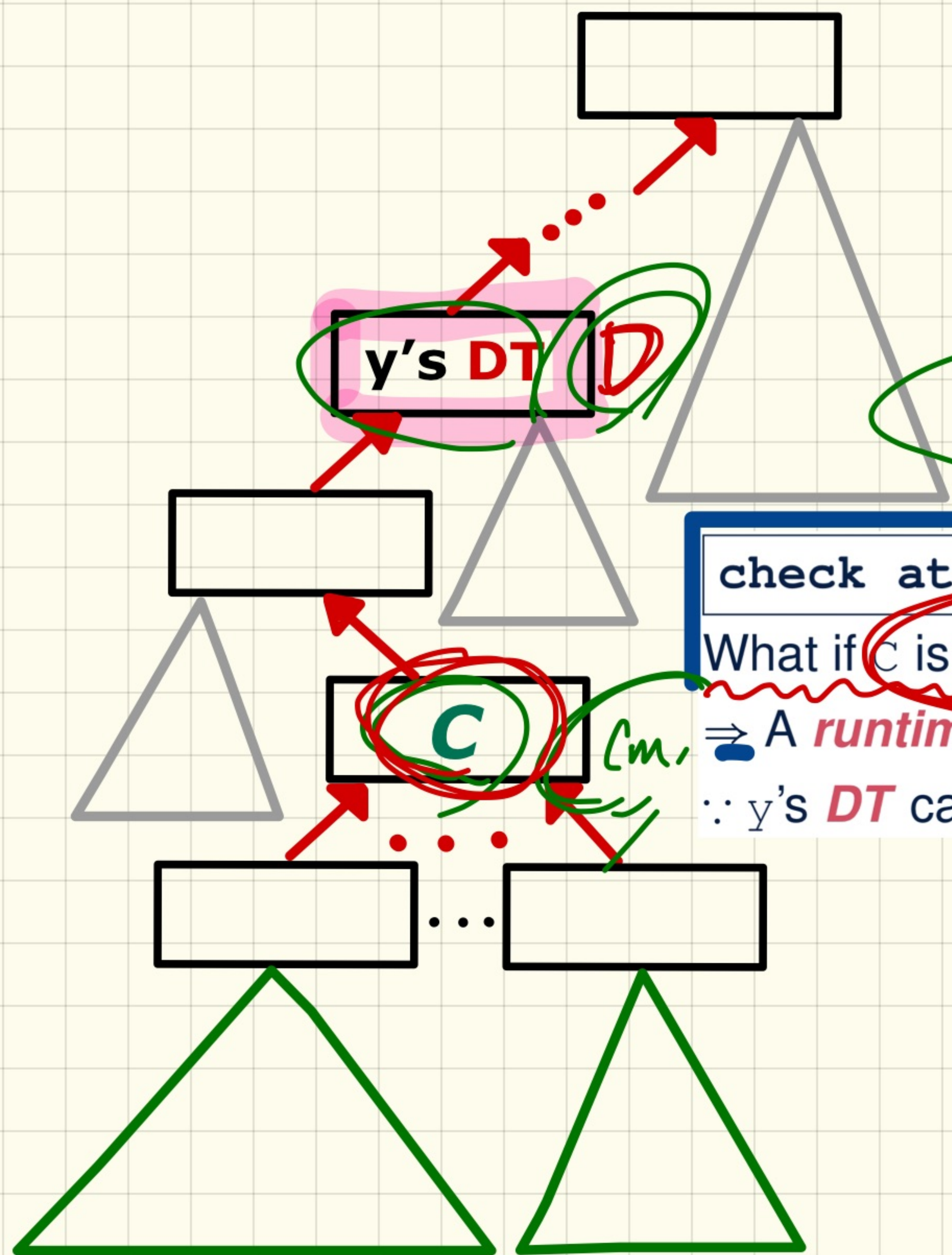
then

vs_jm ST: RS

nrs_alan ST: NRS

end

Ancestors, Expectations, Descendants, and Code Reuse



create $\{D\}$ y. make
 Assume we allowed this case
 \Rightarrow ST of $C-y$: C
 $C-y$ (C_m) as $C-y$.

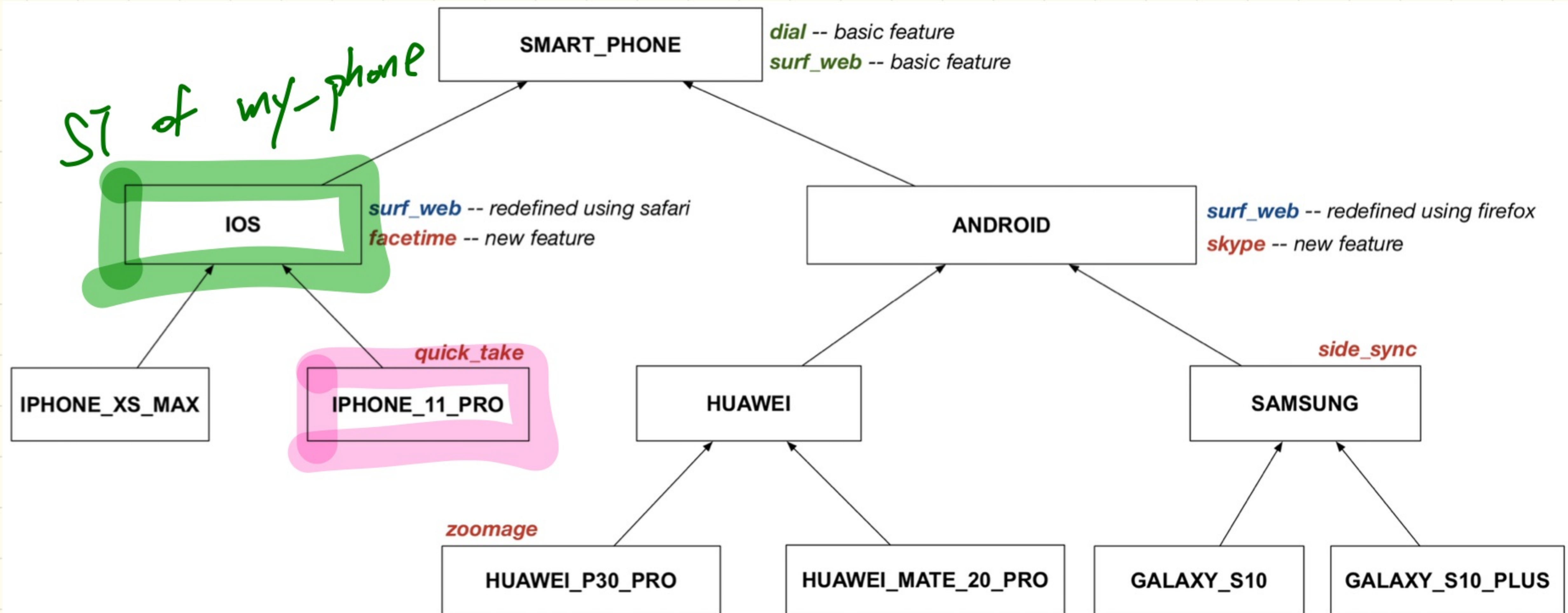
```
check attached  $C$  of  $y$  then ... end
```

always compiles

What if C is not an ancestor of y 's DT ?
 \Rightarrow A runtime assertion violation occurs!
 \therefore y 's DT cannot fulfill the expectation of C .

for a case to be successful \rightarrow "C" must be an ancestor of y 's DT .

Violation-Free Cast: Upwards or Downwards (1)



`my_phone: IOS`

```
create { IPHONE_11_PRO } my_phone.make
```

```
-- can only call features defined in IOS on myPhone
```

```
-- dial, surf_web, facetime ● quick_take, skype, side_sync, zoomage ●
```

```
check attached { SMART_PHONE } my_phone as sp then
```

```
-- can now call features defined in SMART_PHONE on sp
```

```
-- dial, surf_web ● facetime, quick_take, skype, side_sync, zoomage ●
```

```
end
```

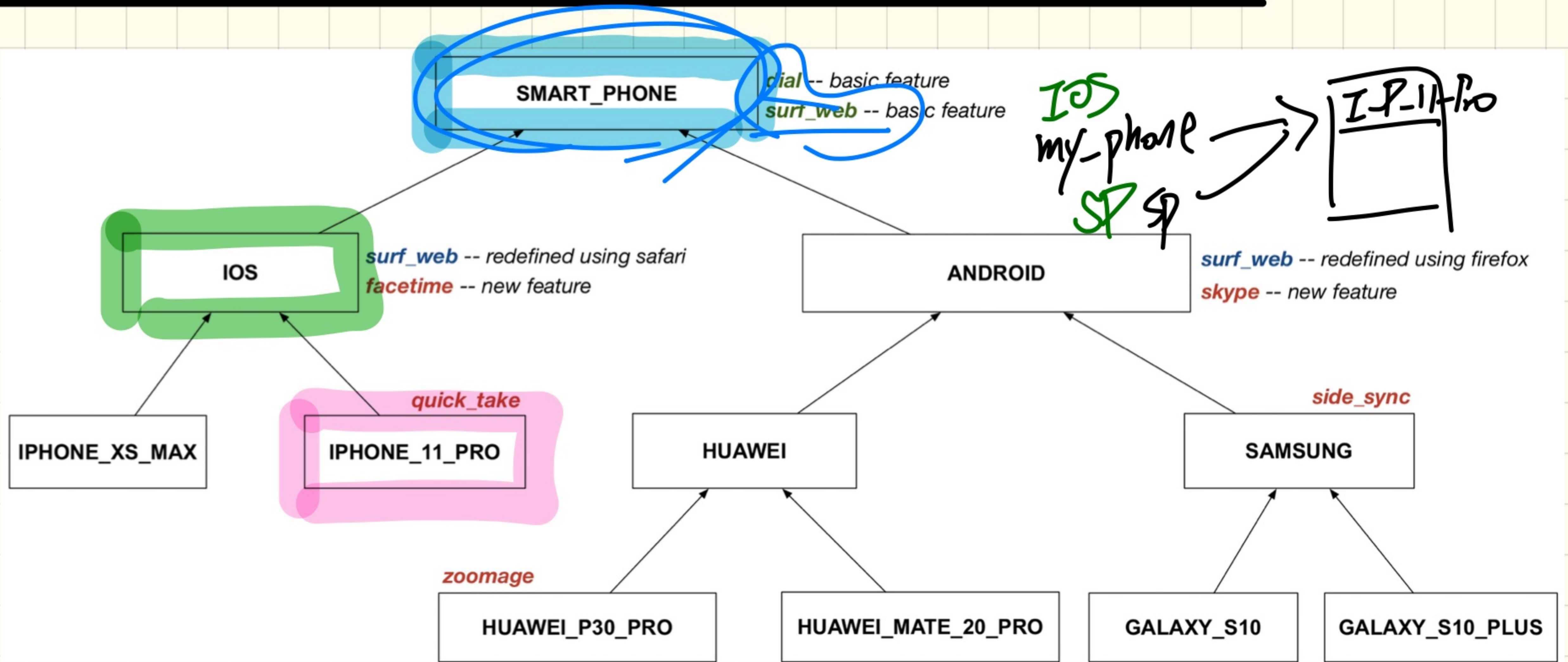
```
check attached { IPHONE_11_PRO } my_phone as ip11_pro then
```

```
-- can now call features defined in IPHONE_11_PRO on ip11_pro
```

```
-- dial, surf_web, facetime, quick_take ● skype, side_sync, zoomage ●
```

```
end
```

Violation-Free Cast: Upwards or Downwards (2)

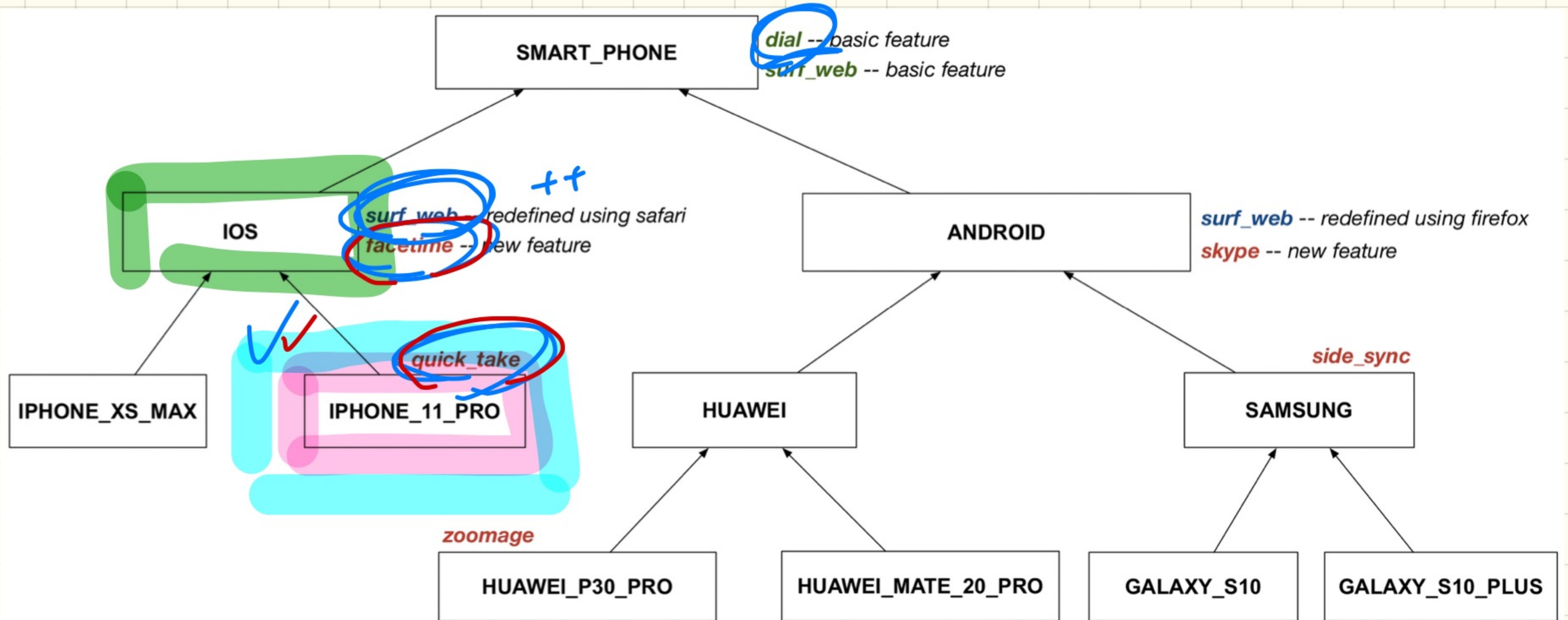


```

my_phone: IOS
create { IPHONE_11_PRO } my_phone.make
  -- can only call features defined in IOS on myPhone
  -- dial, surf_web, facetime, quick_take, skype, side_sync, zoomage
check attached { SMART_PHONE } my_phone as sp then
  -- can now call features defined in SMART_PHONE on sp
  -- dial, surf_web, facetime, quick_take, skype, side_sync, zoomage
end
check attached { IPHONE_11_PRO } my_phone as ip11_pro then
  -- can now call features defined in IPHONE_11_PRO on ip11_pro
  -- dial, surf_web, facetime, quick_take, skype, side_sync, zoomage
end
  
```

Handwritten annotations:
 - Blue circles around 'IOS' and 'SMART_PHONE' in the code.
 - Blue arrows pointing from 'ST: IOS' and 'ST: SMART_PHONE' to the corresponding code elements.
 - Blue circles around 'sp' and 'ip11_pro' in the code.
 - Blue underlines under 'skype', 'side_sync', and 'zoomage' in the code.
 - A red checkmark is next to the first 'check attached' block.
 - A box labeled 'IP-11 Pro' with an arrow pointing to 'IPHONE_11_PRO' in the diagram above.

Violation-Free Cast: Upwards or Downwards (3)

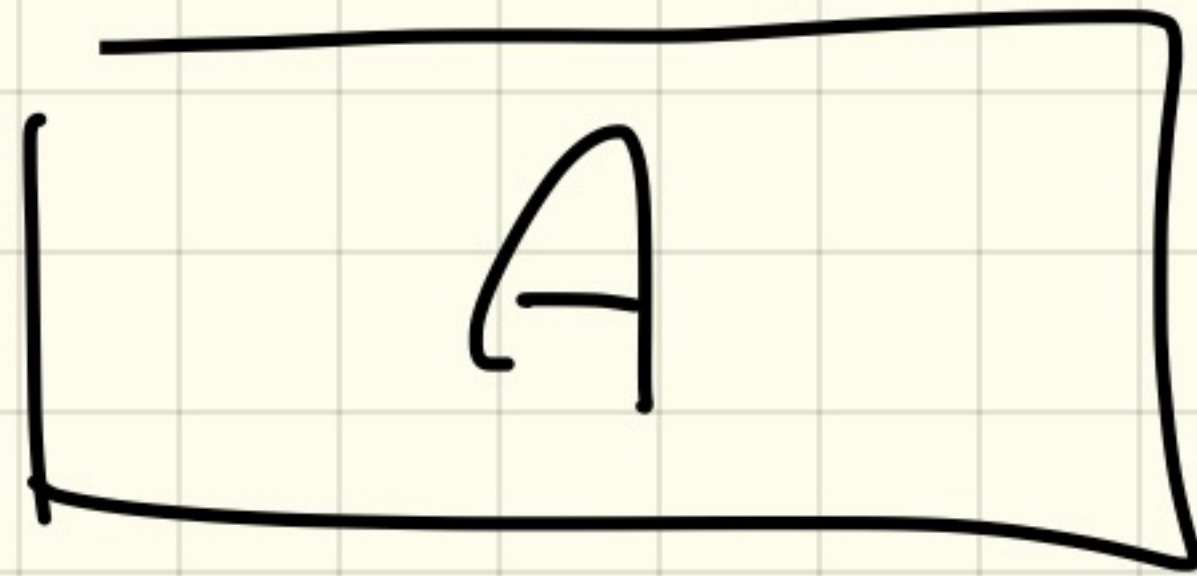


```

my_phone: IOS
create { IPHONE_11_PRO } my_phone.make
  -- can only call features defined in IOS on myPhone
  -- dial, surf_web, facetime ● quick_take, skype, side_sync, zoomage ●
check attached { SMART_PHONE } my_phone as sp then
  -- can now call features defined in SMART_PHONE on sp
  -- dial, surf_web ● facetime, quick_take, skype, side_sync, zoomage ●
end
check attached { IPHONE_11_PRO } my_phone as ip11_pro then
  -- can now call features defined in IPHONE_11_PRO on ip11_pro
  -- dial, surf_web, facetime, quick_take ● skype, side_sync, zoomage ●
end
    
```

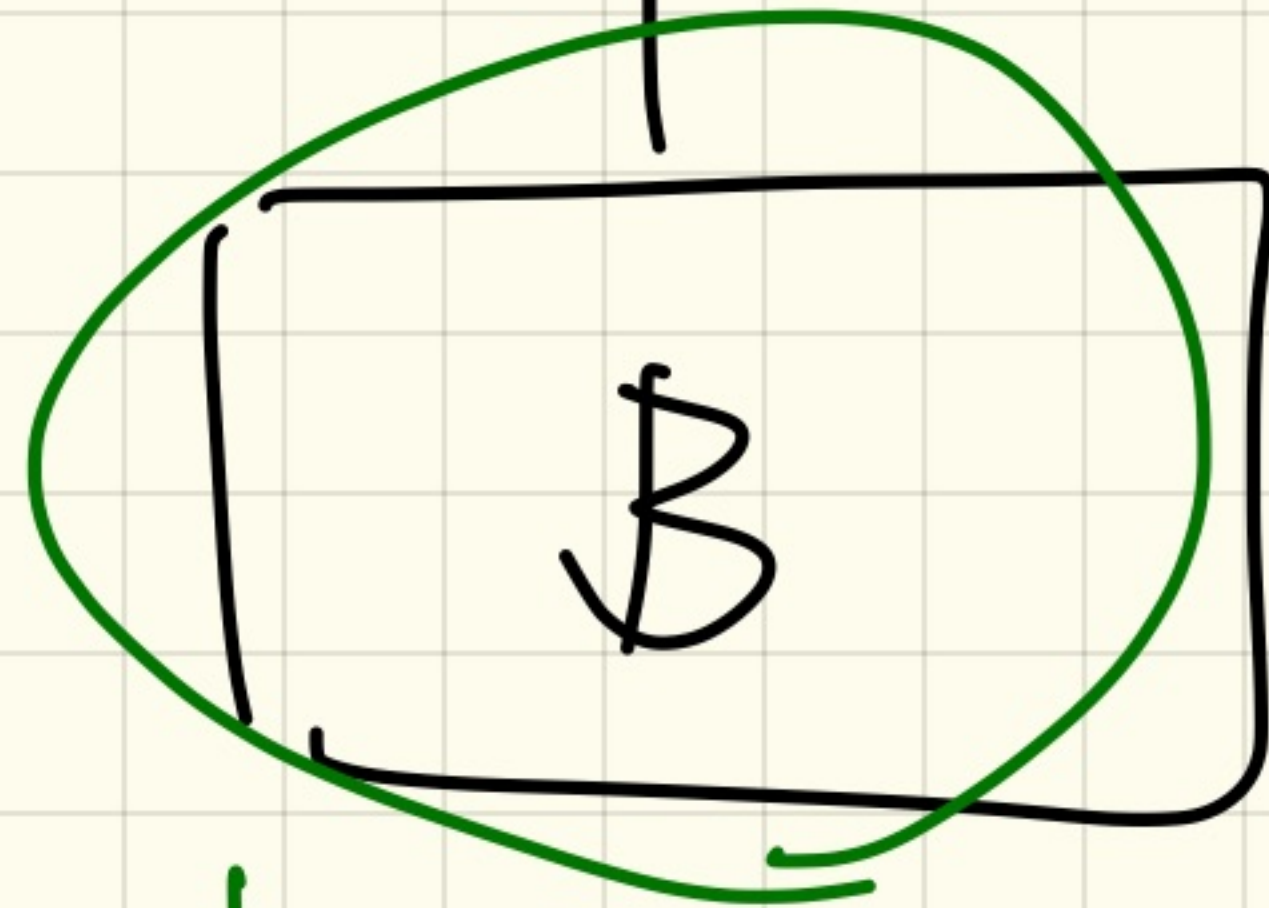
DT of m-p to IP-11-Pro

ip11-pro can be expected features of ancestors of it



am

print("A.am")



am + f

bm

print("B.am")

o: B

create {B} o. make

o. am → "B.am"

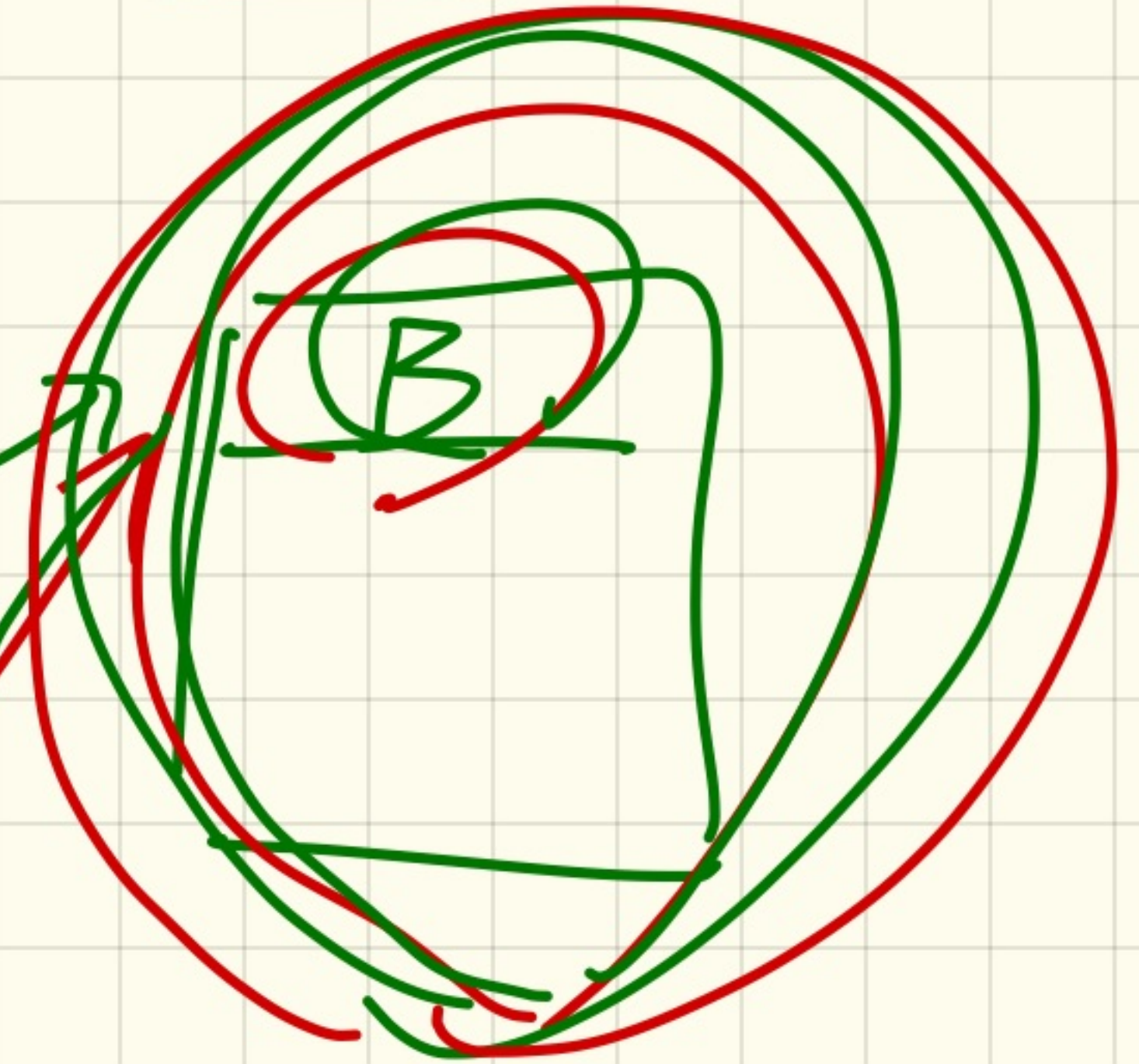
check {A} o as a-o then

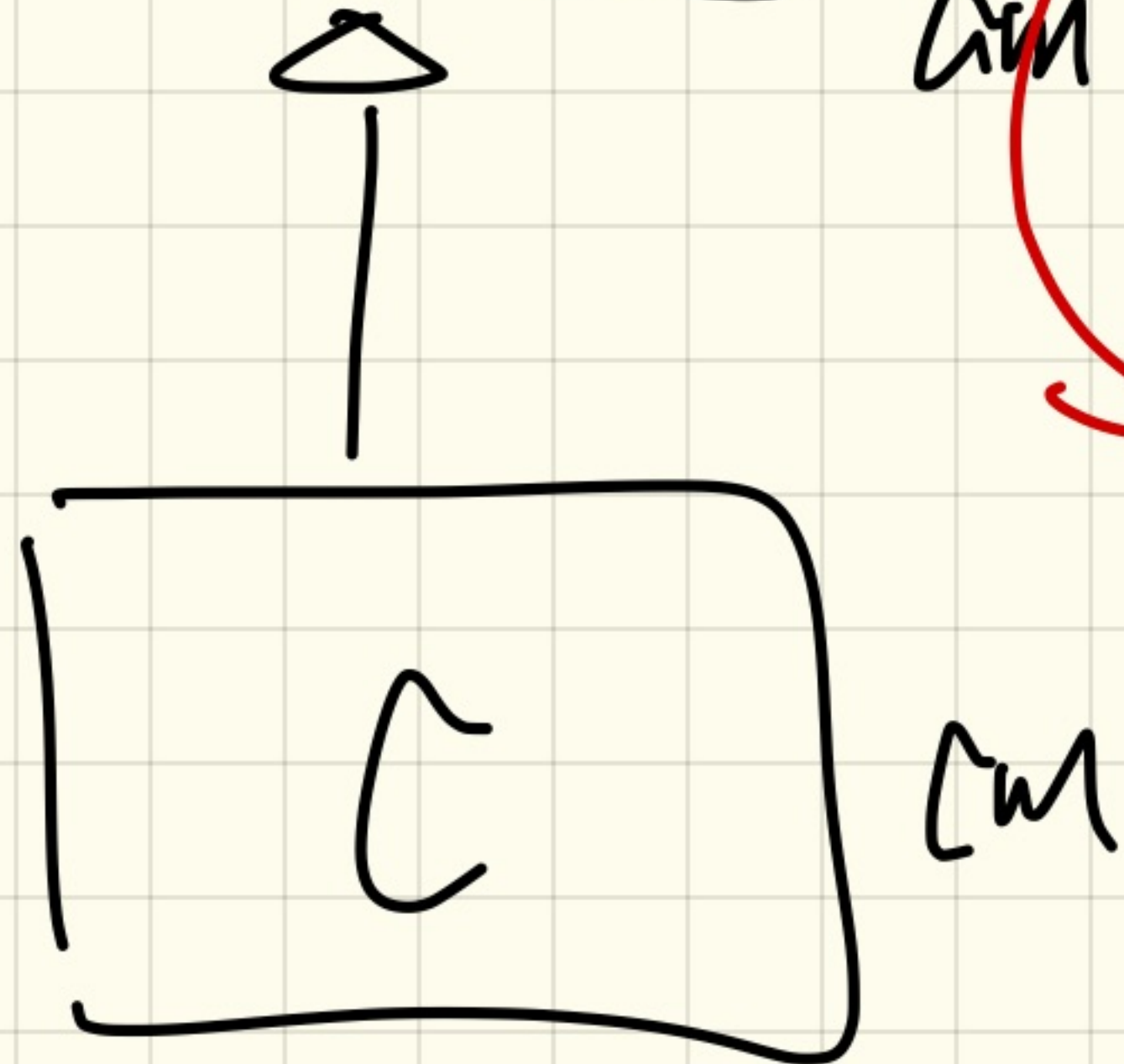
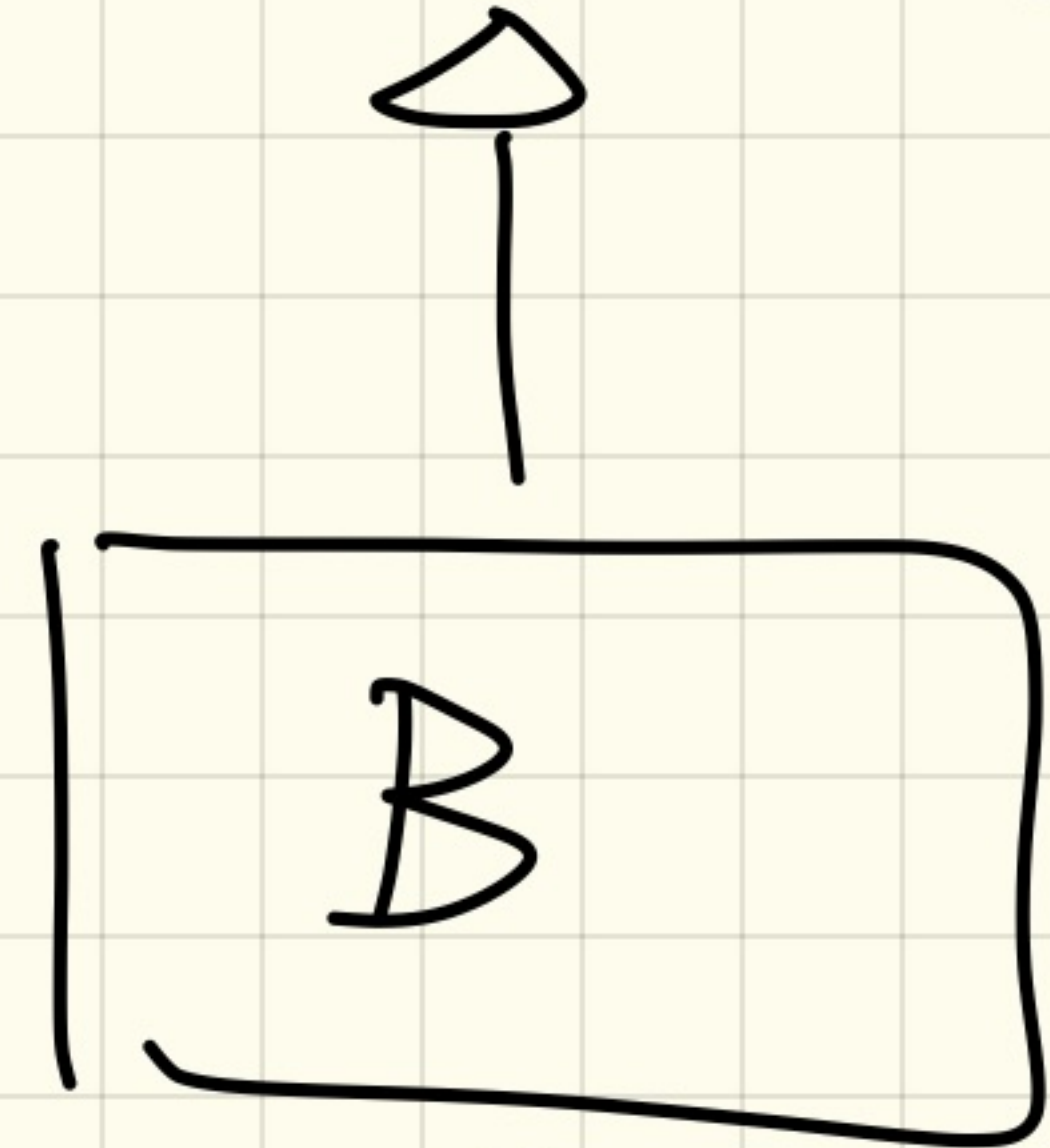
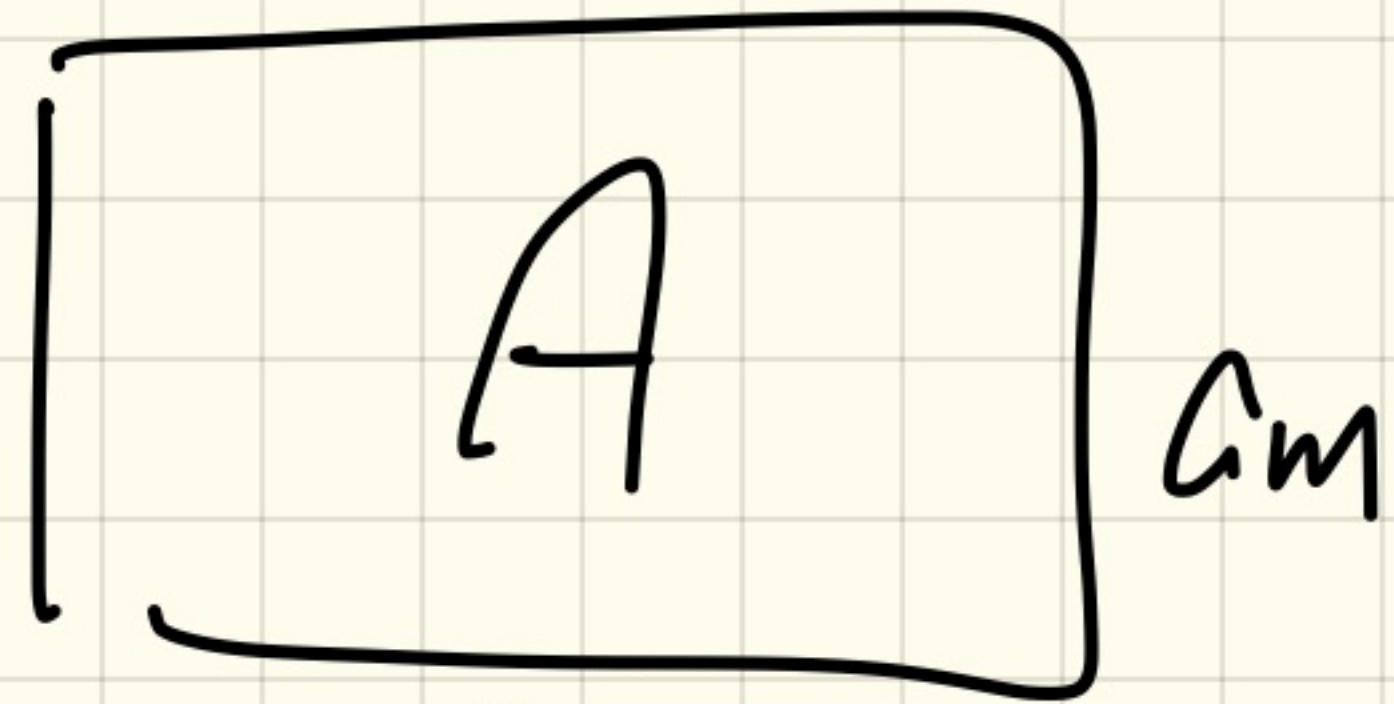
ST: A (a-o). am → "B.am"

end

ST: B

a-o
ST: A





b: ~~DA~~ check attached b as
C=O

bm
Am + f
end

whether this cast
succeeds or not
depends on if C
is an ancestor of
b's DT.

Does a line compile?

ST

Version of feature called?

DT.