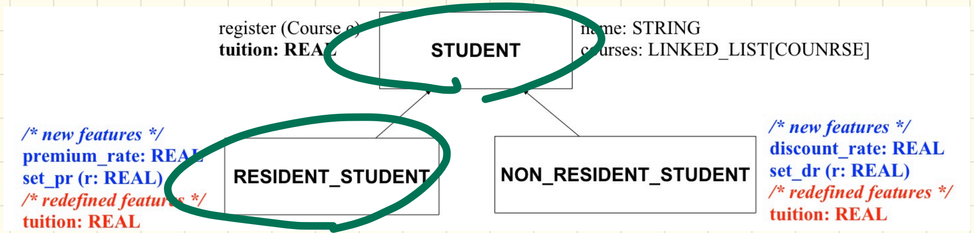
LECTURE 17

MONDAY MARCH 9

Labtest 2 (course wiki/forum):

- undo/redo design pattern
- Reading: OOSC Ch. 21
- Exercise from Github

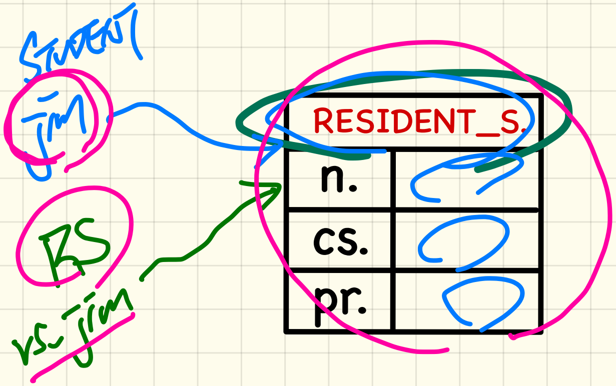
Type Cast: Motivation



RS S

```
local jim: STUDENT; rs: RESIDENT_STUDENT
do create {RESIDENT_STUDENT} jim.make ("J. Davis")
   rs := jim
   rs.setPremiumRate(1.5)
```

X



alias

ST

check attached {RS} jim as rs_jim then

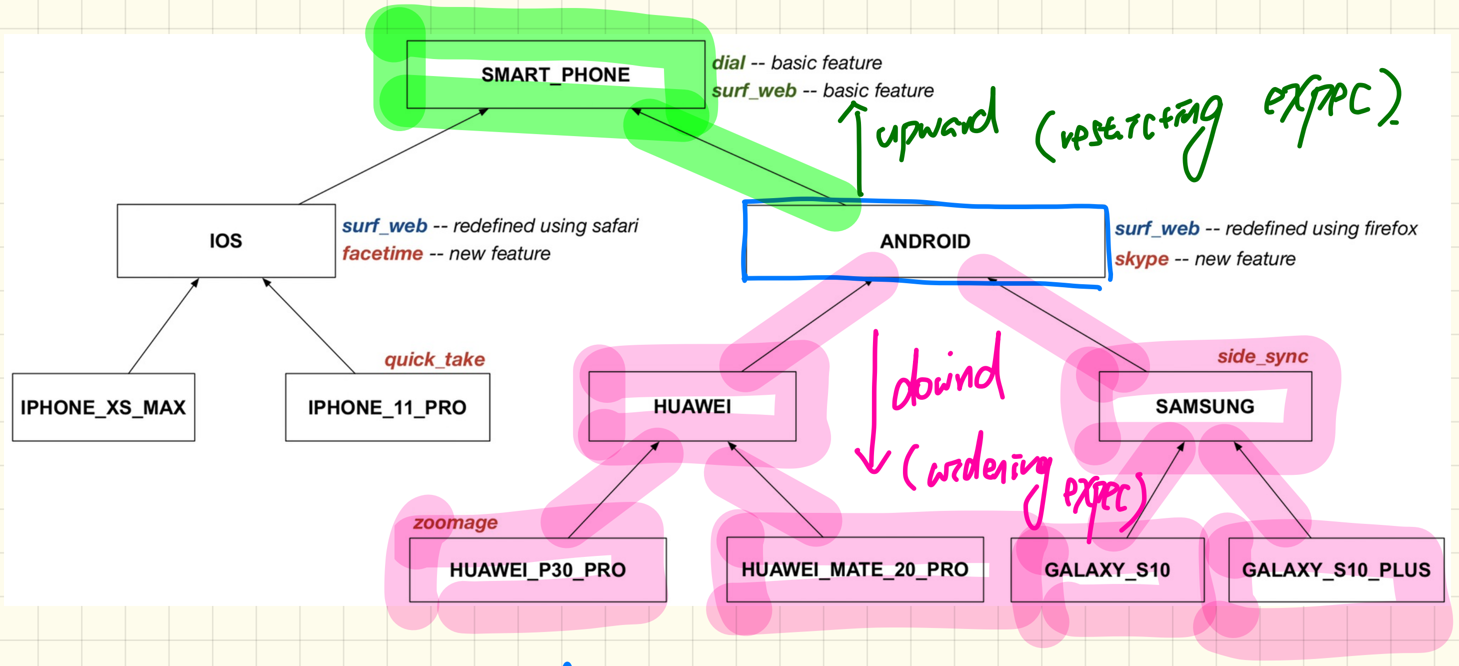
[rs := rs_jim

ST:RS ST: RS

end

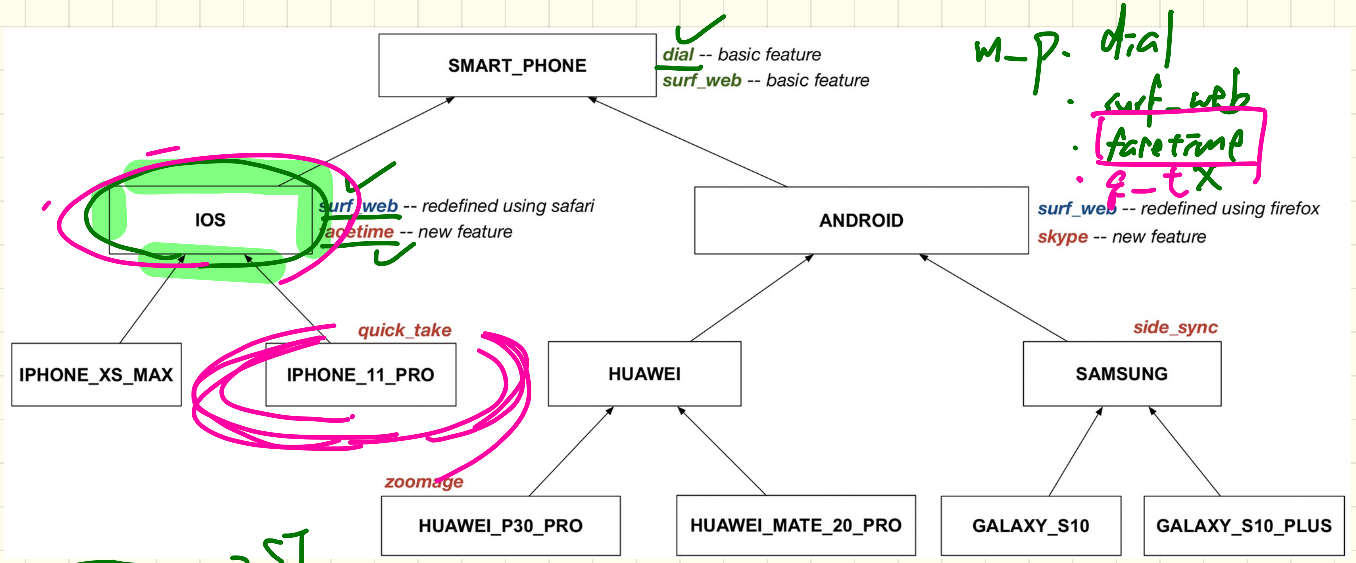
rs_jim X

Multi-Level Inheritance Hierarchy of Smartphones



P : ANDROID
↳ ST

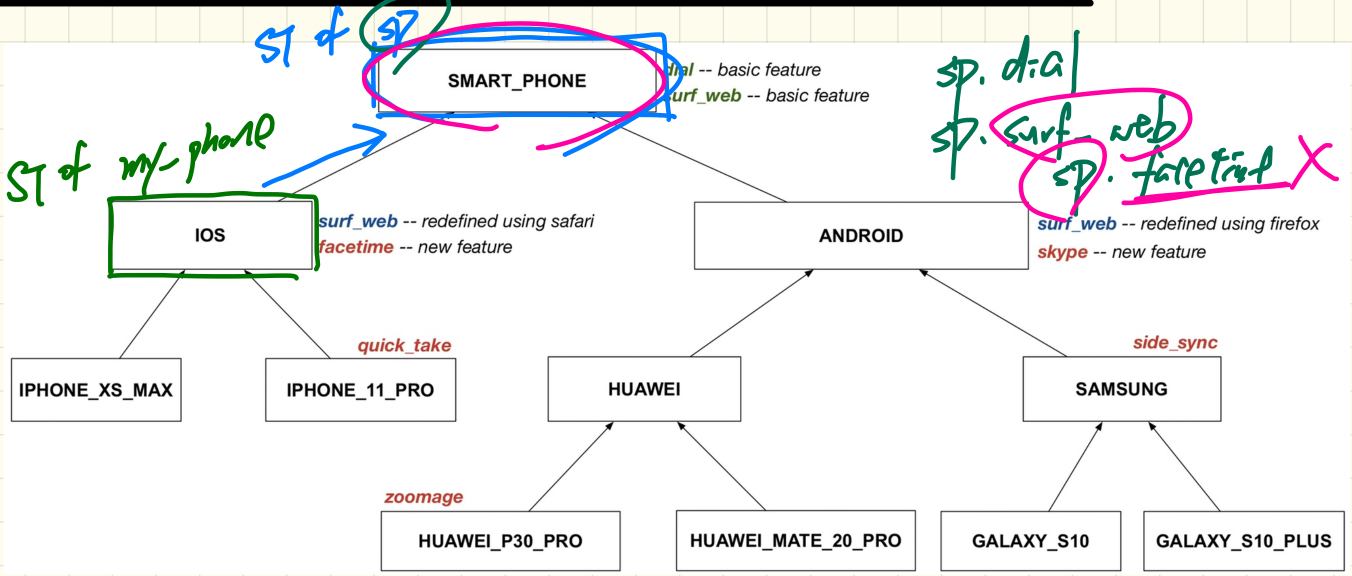
Violation-Free Cast: Upwards or Downwards (1)



```
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
```

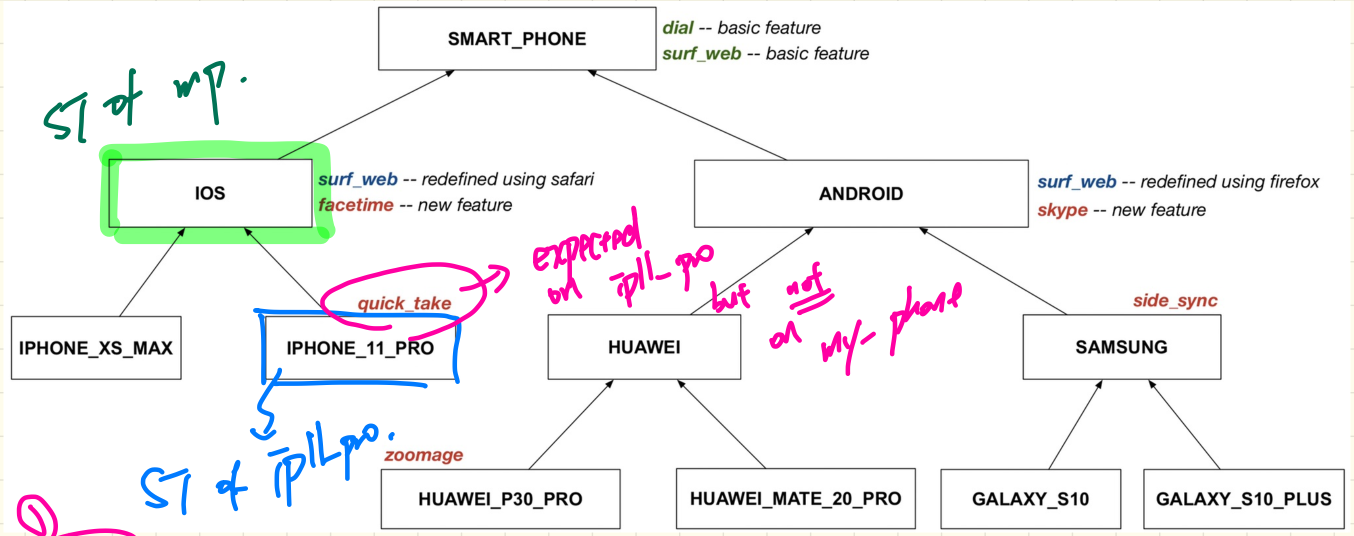
→ ST
→ DT

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
```

Violation-Free Cast: Upwards or Downwards (3)

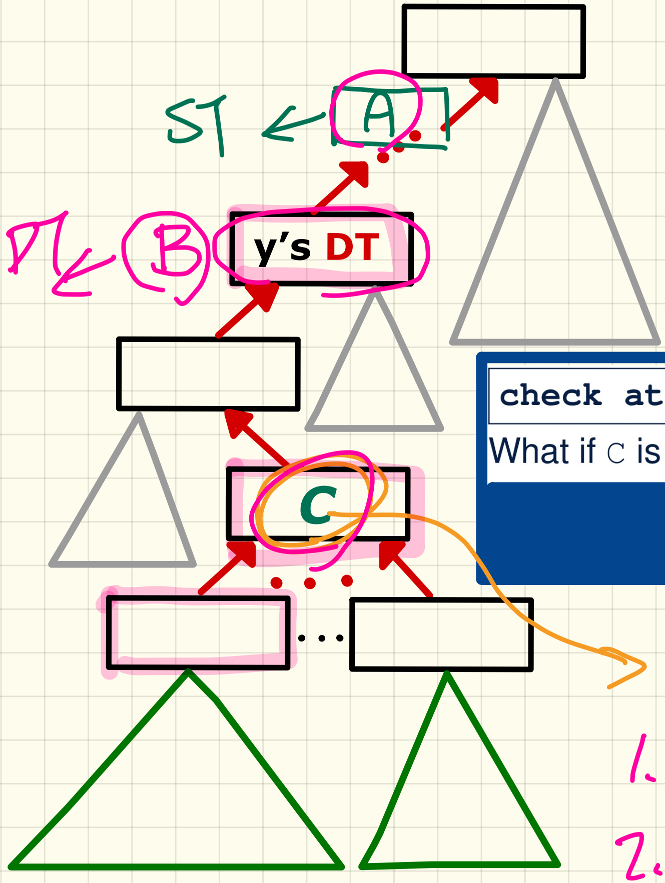


①

```
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
```

②

Ancestors, Expectations, Descendants, and Code Reuse



obj: A
create {B} obj.make

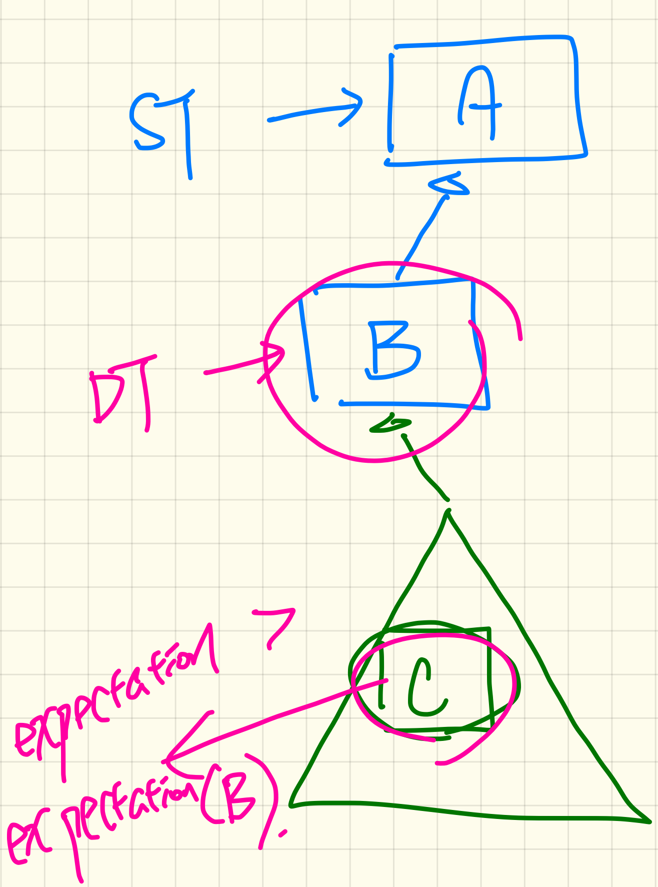
```
check attached {C} ~~y~~ obj then ... end
```

always compiles

What if C is not an **ancestor** of y's ***DT***?

the type to cast obj into

1. casting obj downto C compiles
2. Runtime?



obj : A

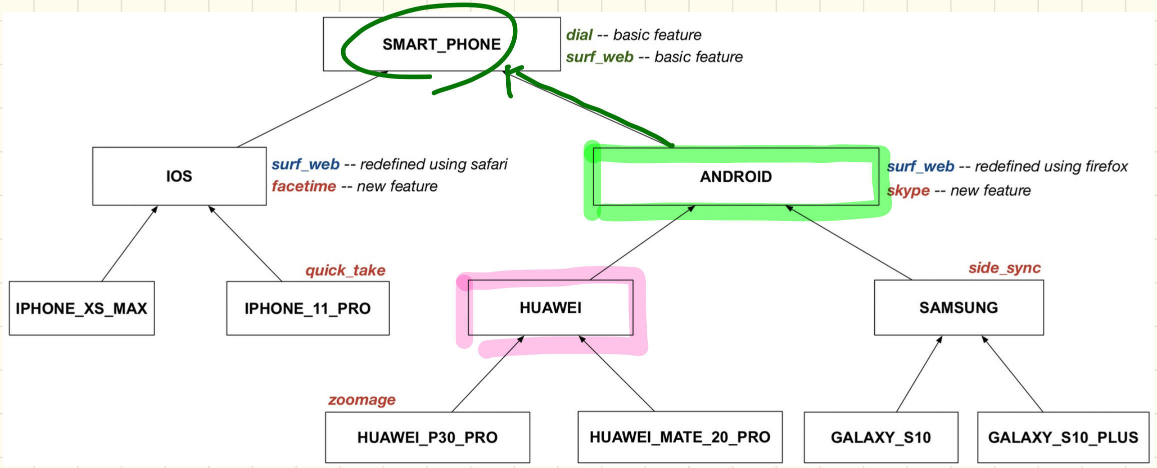
create {B} obj.make

```
check attached {C} obj as C_obj
    C_obj
end
```

↳ cast violation at runtime

∵ C_obj would be expected to be called features from C

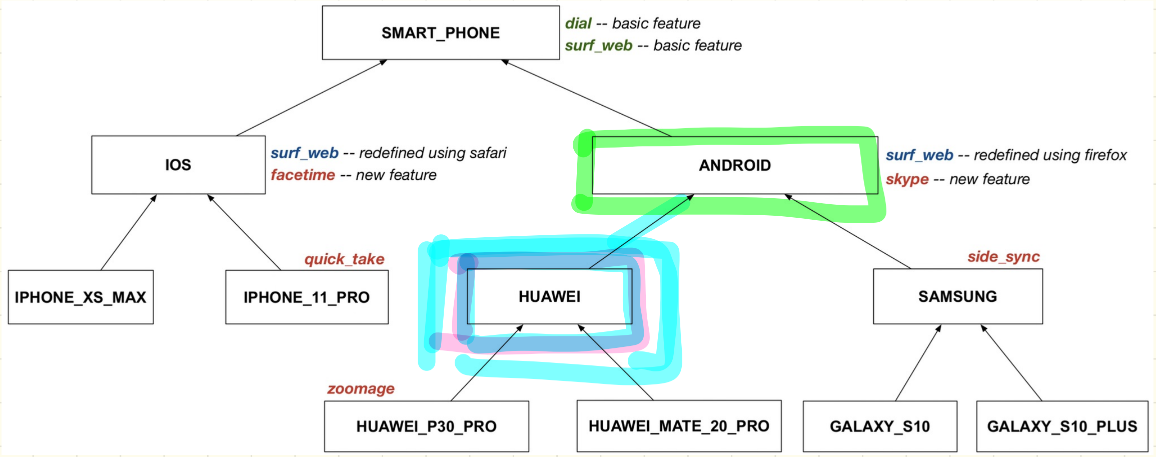
Cast Violation at Runtime (1)



```
test_smart_phone_type_cast_violation
  local mine: ANDROID
  do create {HUAWEI} mine.make
    -- ST of mine is ANDROID; DT of mine is HUAWEI
    check attached {SMART_PHONE} mine as sp then ... end
    -- ST of sp is SMART_PHONE; DT of sp is HUAWEI
    check attached {HUAWEI} mine as huawei then ... end
    -- ST of huawei is HUAWEI; DT of huawei is HUAWEI
    check attached {SAMSUNG} mine as samsung then ... end
    -- Assertion violation
    -- ∵ SAMSUNG is not ancestor of mine's DT (HUAWEI)
    check attached {HUAWEI_P30_PRO} mine as p30_pro then ... end
    -- Assertion violation
    -- ∵ HUAWEI_P30_PRO is not ancestor of mine's DT (HUAWEI)
end
```

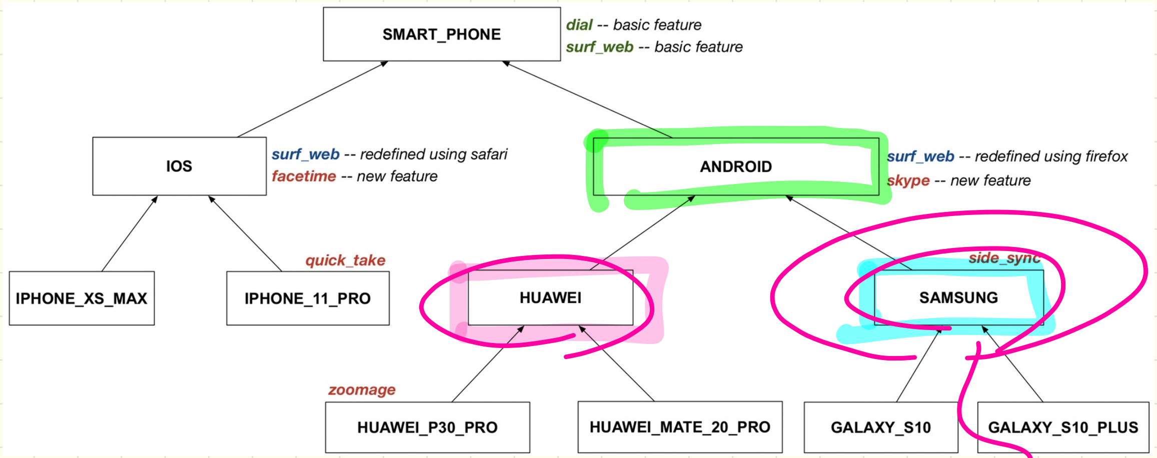
✓

Cast Violation at Runtime (2)



```
test_smart_phone_type_cast_violation
  local mine: ANDROID
  do create {HUAWEI} mine.make
    -- ST of mine is ANDROID; DT of mine is HUAWEI
    check attached {SMART_PHONE} mine as sp then ... end
    -- ST of sp is SMART_PHONE; DT of sp is HUAWEI
    check attached {HUAWEI} mine as huawei then ... end
    -- ST of huawei is HUAWEI; DT of huawei is HUAWEI
    check attached {SAMSUNG} mine as samsung then ... end
    -- Assertion violation
    -- ∵ SAMSUNG is not ancestor of mine's DT (HUAWEI)
    check attached {HUAWEI_P30_PRO} mine as p30_pro then ... end
    -- Assertion violation
    -- ∵ HUAWEI_P30_PRO is not ancestor of mine's DT (HUAWEI)
end
```

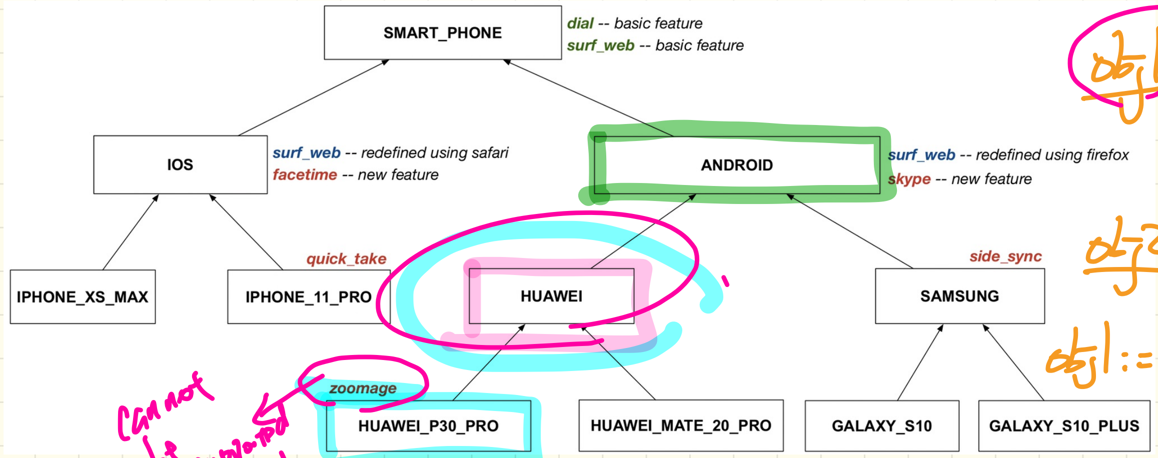
Cast Violation at Runtime (3)

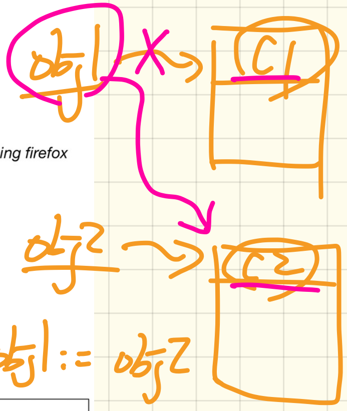


```
test_smart_phone_type_cast_violation
  local mine: ANDROID
  do create {HUAWEI} mine.make
    -- ST of mine is ANDROID; DT of mine is HUAWEI
    check attached {SMART_PHONE} mine as sp then ... end
    -- ST of sp is SMART_PHONE; DT of sp is HUAWEI
    check attached {HUAWEI} mine as huawei then ... end
    -- ST of huawei is HUAWEI; DT of huawei is HUAWEI
    check attached {SAMSUNG} mine as samsung then ... end
    -- Assertion violation
    -- ∵ SAMSUNG is not ancestor of mine's DT (HUAWEI)
    check attached {HUAWEI_P30_PRO} mine as p30_pro then ... end
    -- Assertion violation
    -- ∵ HUAWEI_P30_PRO is not ancestor of mine's DT (HUAWEI)
end
```

Runtime violation
∵ DT cannot support expec on SAMSUNG.

Cast Violation at Runtime (4)





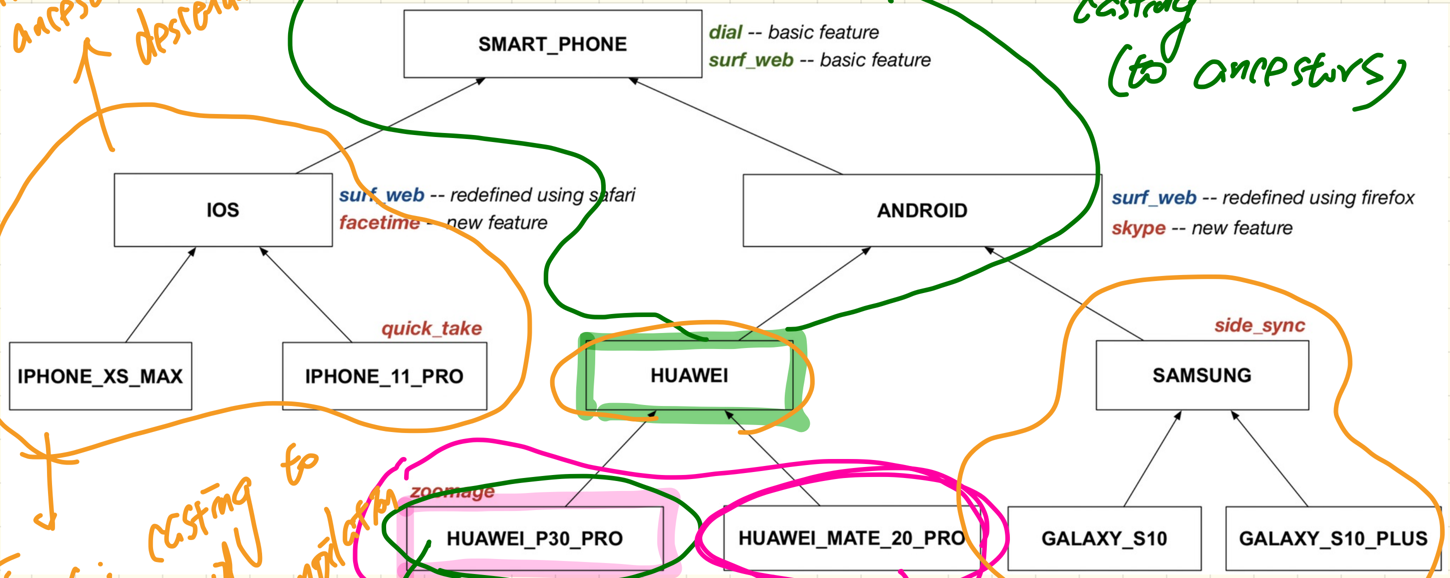
```
test_smart_phone_type_cast_violation
  local mine: ANDROID
  do create {HUAWEI} mine.make
    -- ST of mine is ANDROID; DT of mine is HUAWEI
    check attached {SMART_PHONE} mine as sp then ... end
    -- ST of sp is SMART_PHONE; DT of sp is HUAWEI
    check attached {HUAWEI} mine as huawei then ... end
    -- ST of huawei is HUAWEI; DT of huawei is HUAWEI
    check attached {SAMSUNG} mine as samsung then ... end
    -- Assertion violation
    -- ∵ SAMSUNG is not ancestor of mine's DT (HUAWEI)
    check attached {HUAWEI_P30_PRO} mine as p30_pro then ... end
    -- Assertion violation
    -- ∵ HUAWEI_P30_PRO is not ancestor of mine's DT (HUAWEI)
end
```

Rule for avoiding
RT cast violation

Do Not cast
Lower than DT

neither ancestors nor descendants.

upward casting (to ancestors)



Java: casting to this category results in a compilation error

Eiffel: compile.

can cast to any of DT's ancestors without violation

downward casting (to descendants)

e.g. cast violation

Feature Call Arguments: Supplier

```
class STUDENT_MANAGEMENT_SYSTEM {
  ss : ARRAY[STUDENT]  -- ss[i] has static type Student
  add_s (s: STUDENT) do ss[0] := s end
  add_rs (rs: RESIDENT_STUDENT) do ss[0] := rs end
  add_nrs (nrs: NON_RESIDENT_STUDENT) do ss[0] := nrs end
```

STUDENT

ST: ?

ST RS

parameter

ss[1], ss[2], . . . ST: STUDENT.

Say:
   sms: STUDENT_MANAGEMENT_SYSTEM

When should the following calls compile?

```
sms.add_s (o)
sms.add_rs (o)
sms.add_nrs (o)
```

argument

pass by value

parament := argument

s := o

supplier.

add_s ($\underline{s}$: STUDENT)

s := rs

client

rs : RS

sms.add_s (rs)

Feature Call Arguments: Client

```
class STUDENT_MANAGEMENT_SYSTEM {
  ss : ARRAY[STUDENT] -- ss[i] has static type Student
  add_s (s: STUDENT) do ss[0] := s end
  add_rs (rs: RESIDENT_STUDENT) do ss[0] := rs end
  add_nrs (nrs: NON_RESIDENT_STUDENT) do ss[0] := nrs end
```

```
test_polymorphism_feature_arguments
  local
    s1, s2, s3: STUDENT
    rs: RESIDENT_STUDENT ; nrs: NON_RESIDENT_STUDENT
    sms: STUDENT_MANAGEMENT_SYSTEM
  do
    create sms.make
    create {STUDENT} s1.make ("s1")
    create {RESIDENT_STUDENT} s2.make ("s2")
    create {NON_RESIDENT_STUDENT} s3.make ("s3")
    create {RESIDENT_STUDENT} rs.make ("rs")
    create {NON_RESIDENT_STUDENT} nrs.make ("nrs")
```

sms. add_s(s1)

s := s1

sms.add_s (rs)

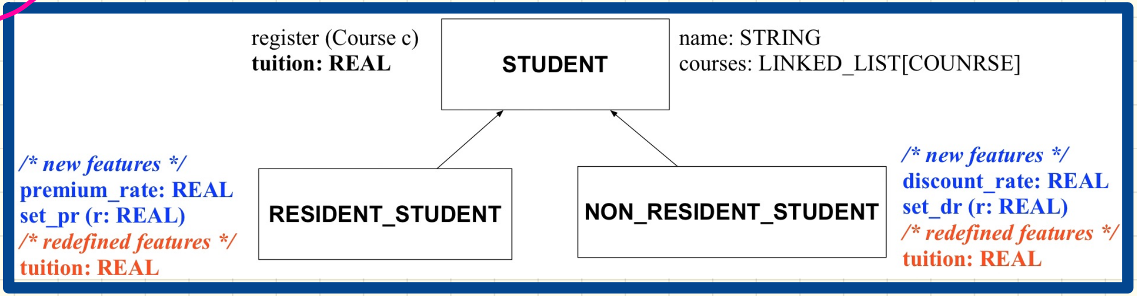
sms.add_rs (s1)

Polymorphic Collection

SMS	
ss	

RESIDENT_S.	
n.	
cs.	
pr.	

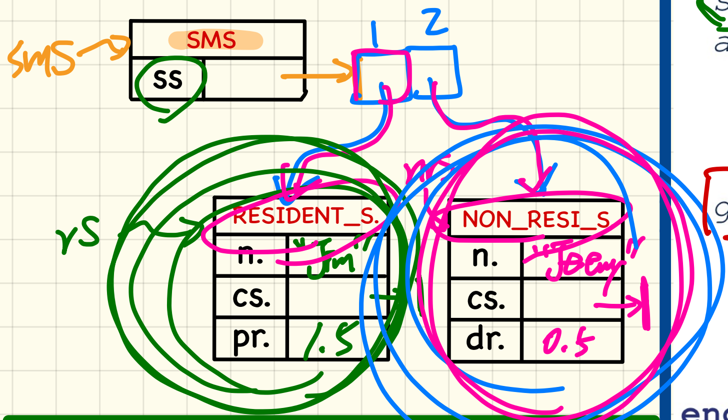
NON_RESI_S.	
n.	
cs.	
dr.	



```
class STUDENT_MANAGEMENT_SYSETM
  students: LINKED_LIST[STUDENT]
  add_student(s: STUDENT)
    do
      students.extend (s)
    end
  registerAll (c: COURSE)
    do
      across
        students as s
      loop
        s.item.register (c)
      end
    end
end
```

```
test_sms_polymorphism: BOOLEAN
  local
    rs: RESIDENT_STUDENT
    nrs: NON_RESIDENT_STUDENT
    c: COURSE
    sms: STUDENT_MANAGEMENT_SYSTEM
  do
    create rs.make ("Jim")
    rs.set_pr (1.5)
    create nrs.make ("Jeremy")
    nrs.set_dr (0.5)
    create sms.make
    sms.add_s (rs)
    sms.add_s (nrs)
    create c.make ("EECS3311", 500)
    sms.register_all (c)
    Result := sms.ss[1].tuition = 750 and sms.ss[2].tuition = 250
  end
```

Feature Call Return Values



```
class STUDENT_MANAGEMENT_SYSTEM {
  ss : LINKED_LIST[STUDENT]
  add_s (s: STUDENT)
    do
      ss.extend (s)
    end
  get_student(i: INTEGER): STUDENT
    require 1 <= i and i <= ss.count
    do
      Result := ss[i]
    end
end
```

ST of R

ST: STUDENT

```
test_sms_polymorphism: BOOLEAN
local
  rs: RESIDENT_STUDENT ; nrs: NON_RESIDENT_STUDENT
  c: COURSE ; sms: STUDENT_MANAGEMENT_SYSTEM
do
  create rs.make ("Jim") ; rs.set_pr (1.5)
  create nrs.make ("Jeremy") ; nrs.set_dr (0.5)
  create sms.make ; sms.add_s (rs) ; sms.add_s (nrs)
  create c.make ("EECS3311", 500) ; sms.register_all (c)
  Result :=
    get_student(1).tuition = 750
    and get_student(2).tuition = 250
end
```

2 3

ss[i]. pr

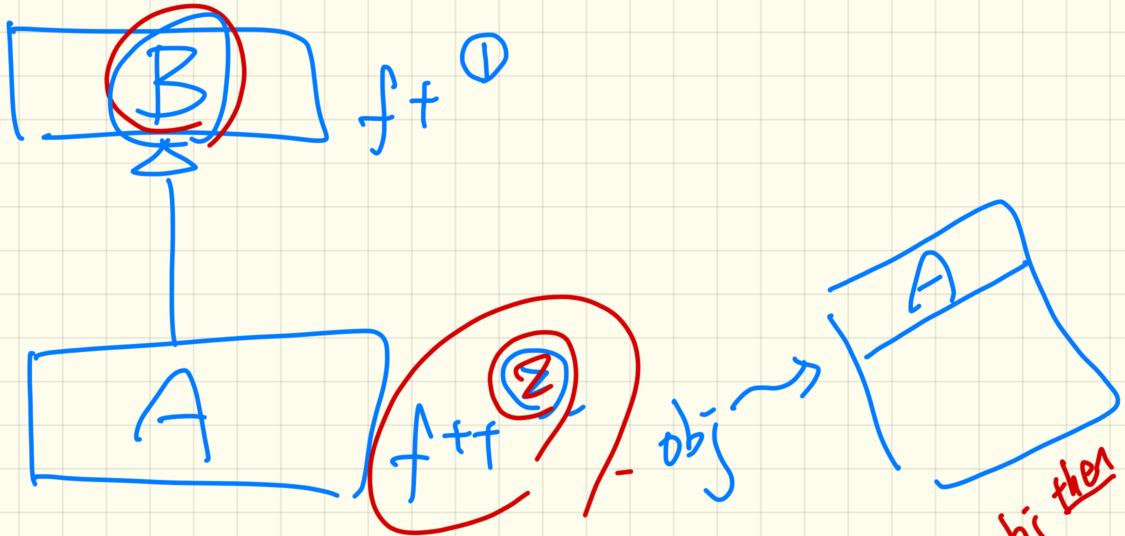
ST: STUDENT

get_student(2). dr X

Possible **DT** of **Result**?

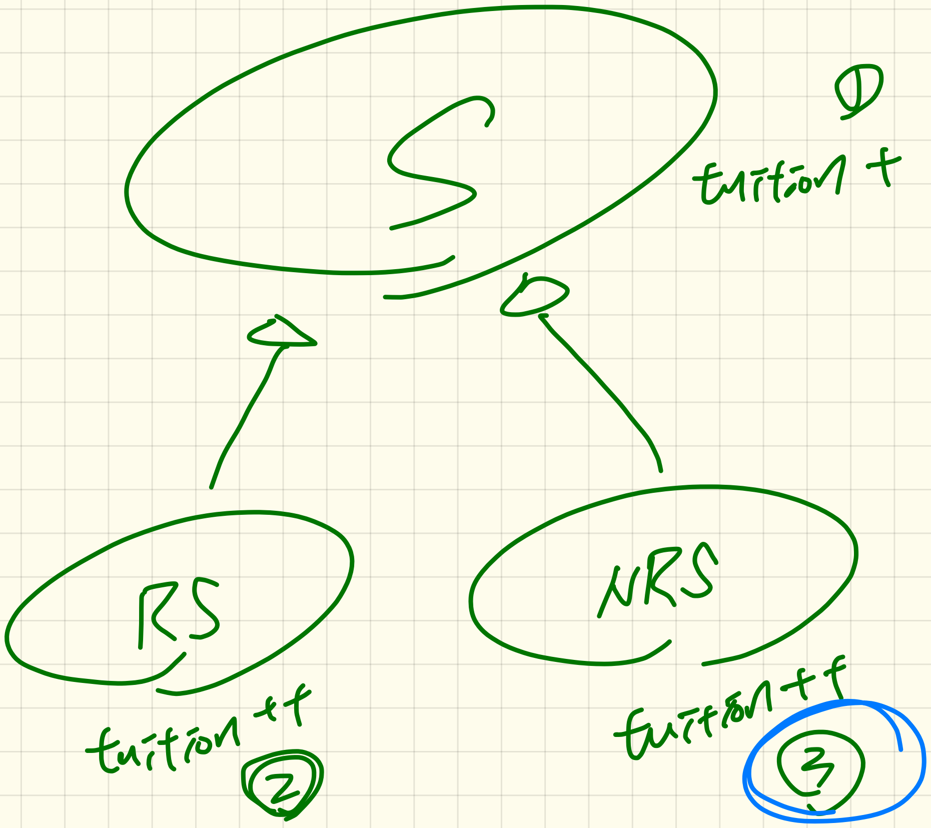
descendant classes of ST

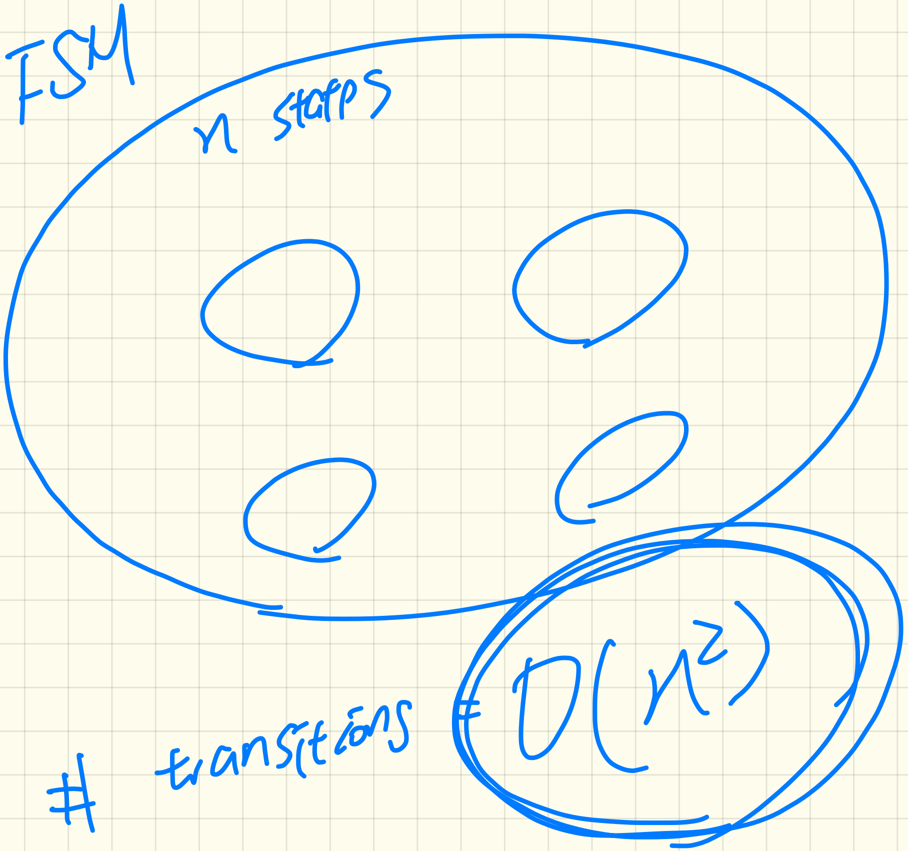
	ST	DT
sms[1]	STUDENT	RS
sms[2]		NRS



```
obj : A
create {A} obj.make
check attached {B} obj as b_obj then
    b_obj.f      -- compiles ②
end
```

DB: version to be called depends on DT.


S
tuition ↑
①
RS
tuition ↑↑
②
NRS
tuition ↑↑
③


FSM
n states
# transitions = O(n²)