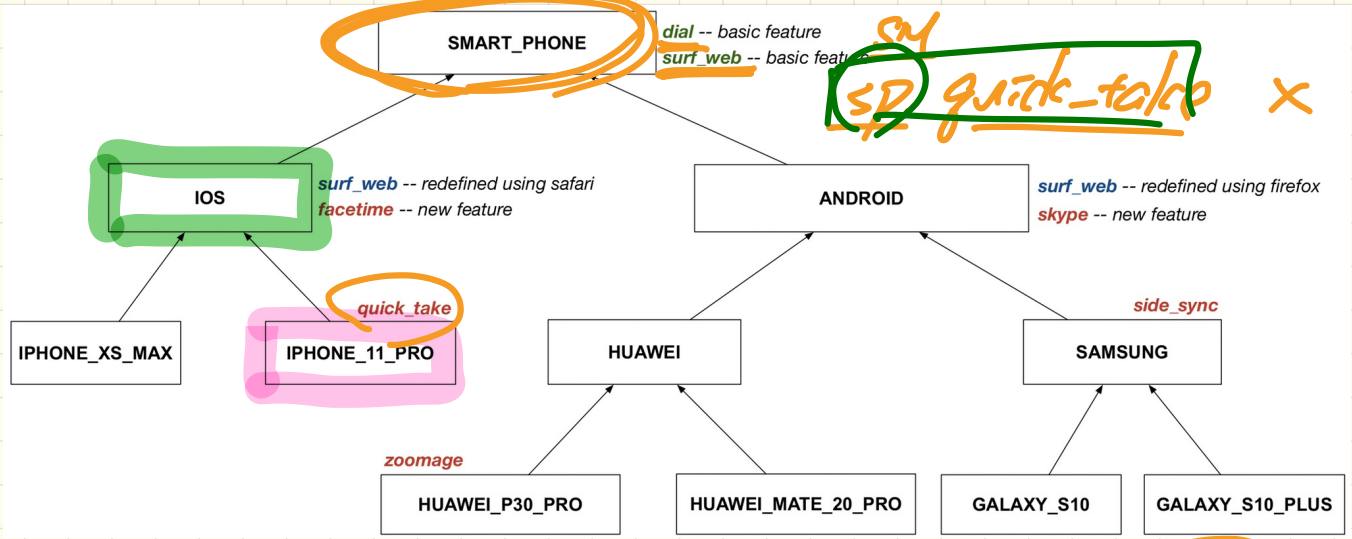
Lecture 12

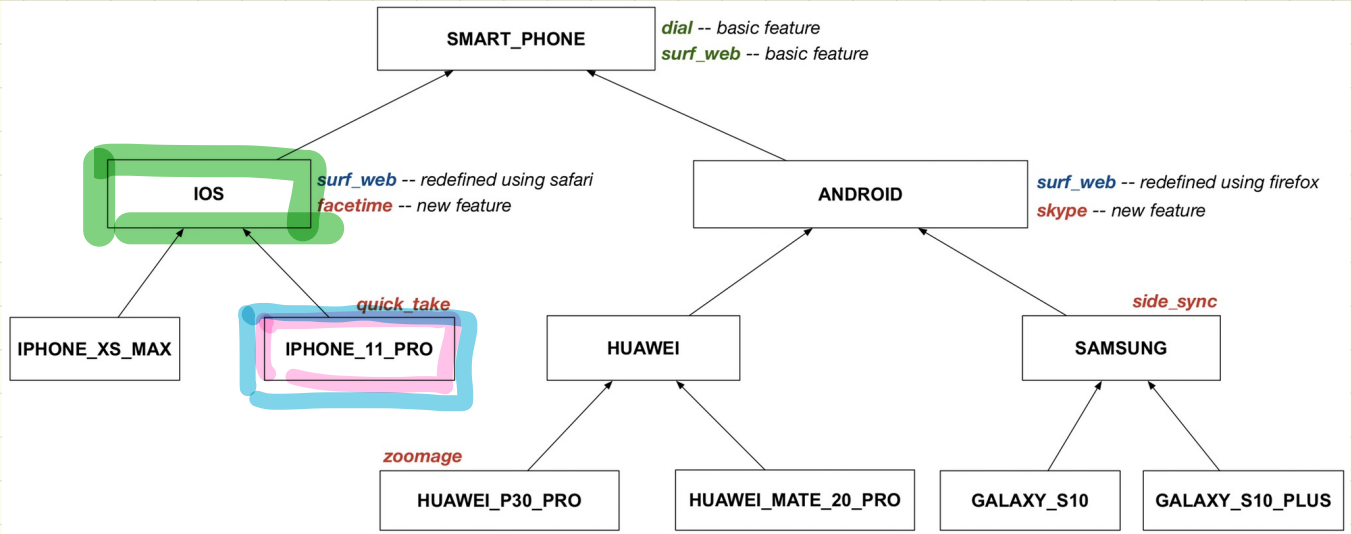
Tuesday October 22

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
```

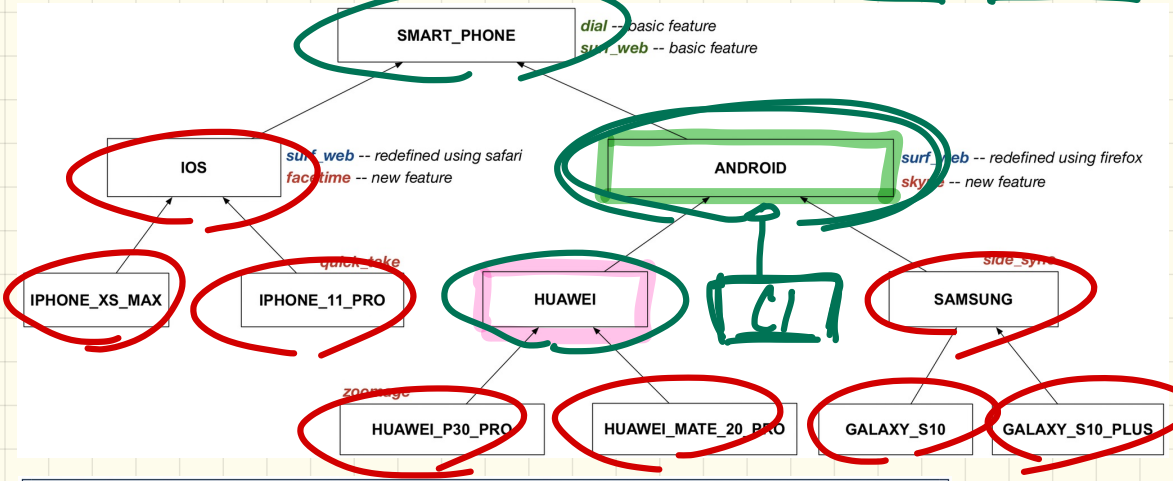
Violation-Free Cast: Upwards or Downwards (3)




```
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
```

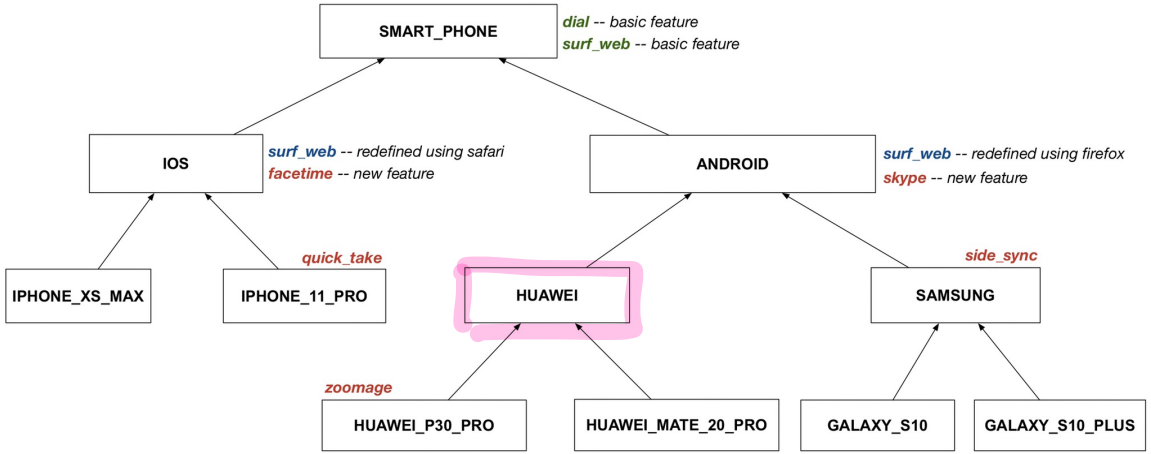
Cast Violation at Runtime (1)

check attached {C1} mine X
↳ assertion violation.



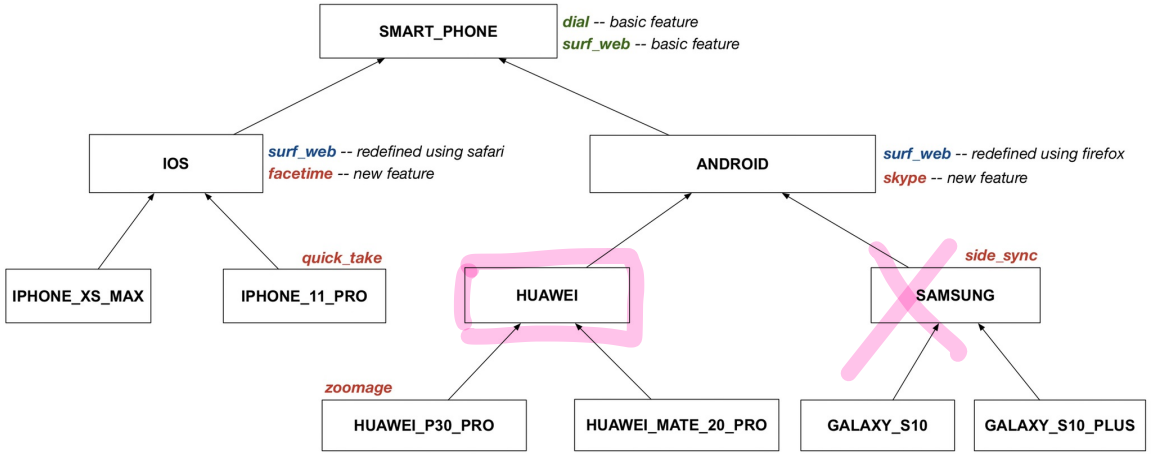

```
test_smart_phone_type_cast_violation
  local mine: ANDROID
  do create {HUAWEI} mine.make
    -- ST of mine is ANDROID; DT of mine is HUAWEI
    check attached {SMART_PHONE} mine as sp then ... end
    -- ST of sp is SMART_PHONE; DT of sp is HUAWEI
    check attached {HUAWEI} mine as huawei then ... 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)



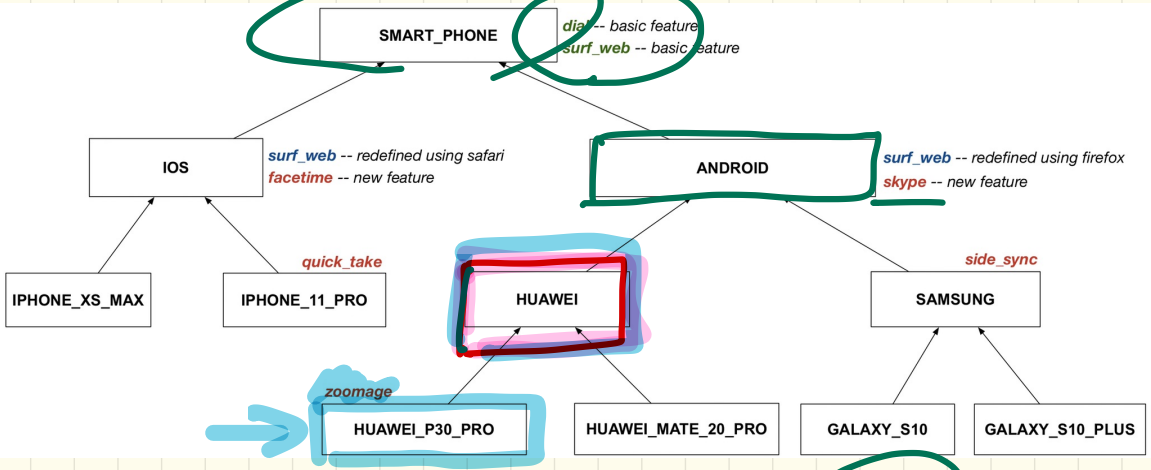

```
test_smart_phone_type_cast_violation
  local mine: ANDROID
  do create {HUAWEI} mine.make
    -- ST of mine is ANDROID; DT of mine is HUAWEI
    check attached {SMART_PHONE} mine as sp then ... end
    -- ST of sp is SMART_PHONE; DT of sp is HUAWEI
    check attached {HUAWEI} mine as huawei then ... 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)

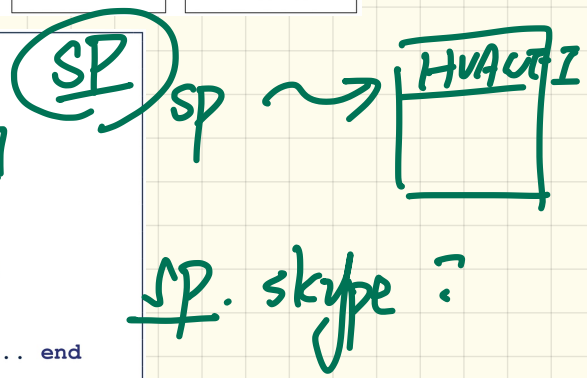



```
test_smart_phone_type_cast_violation
  local mine: ANDROID
  do create {HUAWEI} mine.make
    -- ST of mine is ANDROID; DT of mine is HUAWEI
    check attached {SMART_PHONE} mine as sp then ... end
    -- ST of sp is SMART_PHONE; DT of sp is HUAWEI
    check attached {HUAWEI} mine as huawei then ... 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
```

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
```



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
```

Say:

sms: STUDENT_MANAGEMENT_SYSTEM

When should the following calls compile?

sms.add_s (o)

sms.add_rs (o)

sms.add_nrs (o)

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 (rs)

sms.add_rs (s1)

S: ~~STUDENT~~ NRS → create {NRS} s.make(...)

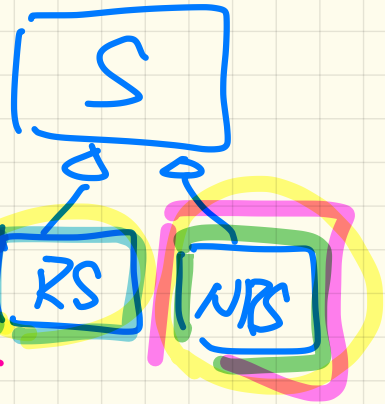
check attached {RS} v as rs1 then

sms.add_rs(rs1)

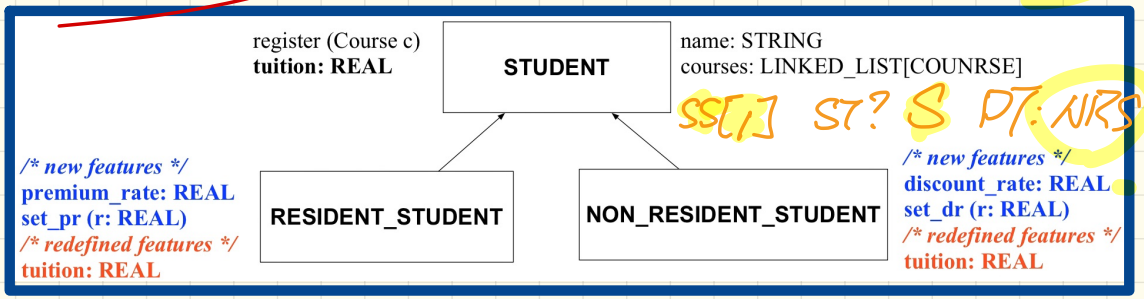
ad

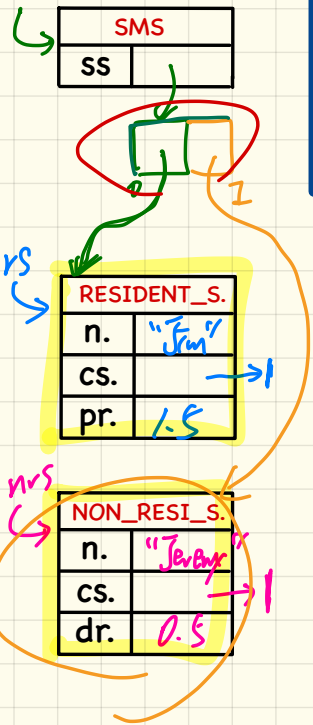
✓ RS := rs1

RS RS



Polymorphic Collection

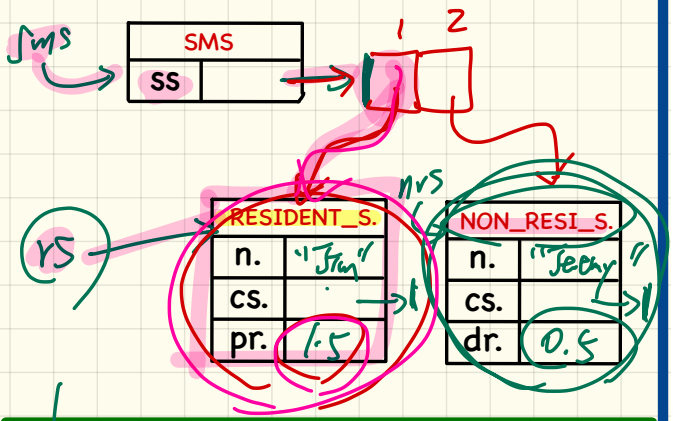





```
test_sms_polymorphism: BOOLEAN
  local
    rs: RESIDENT_STUDENT
    nrs: NON_RESIDENT_STUDENT
    c: COURSE
    sms: STUDENT_MANAGEMENT_SYSTEM
  do
    create rs.make ("Jim")
    rs.set_pr (1.5)
    create nrs.make ("Jeremy")
    nrs.set_dr (0.5)
    create sms.make
    sms.add_s (rs)
    sms.add_s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

```
class STUDENT_MANAGEMENT_SYSTEM
  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
```

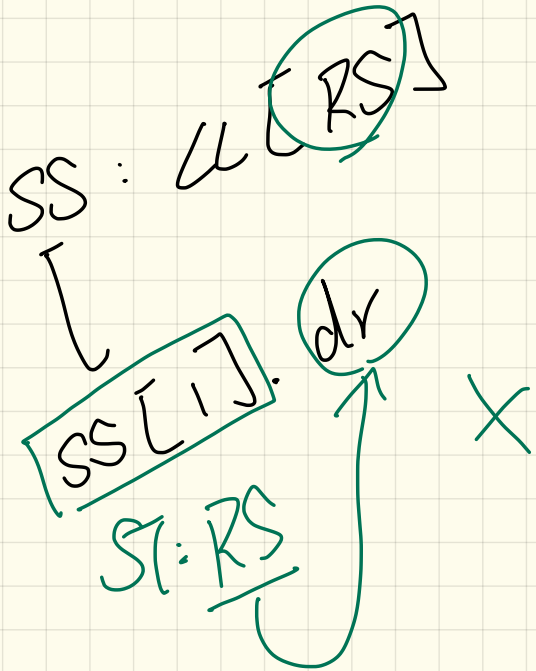
Feature Call Return Values



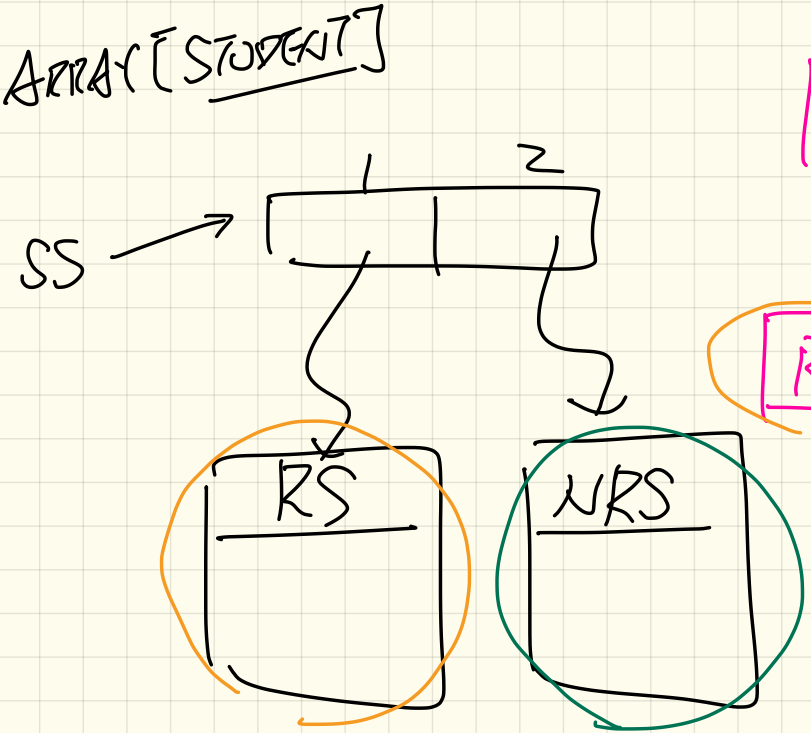

```
test_sms_polymorphism: BOOLEAN
local
  rs: RESIDENT_STUDENT ; nrs: NON_RESIDENT_STUDENT
  c: COURSE ; sms: STUDENT_MANAGEMENT_SYSTEM
do
  create rs.make ("Jim") ; rs.set_pr (1.5)
  create nrs.make ("Jeremy") ; nrs.set_dr (0.5)
  create sms.make ; sms.add_s (rs) ; sms.add_s (nrs)
  create c.make ("EECS3311",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

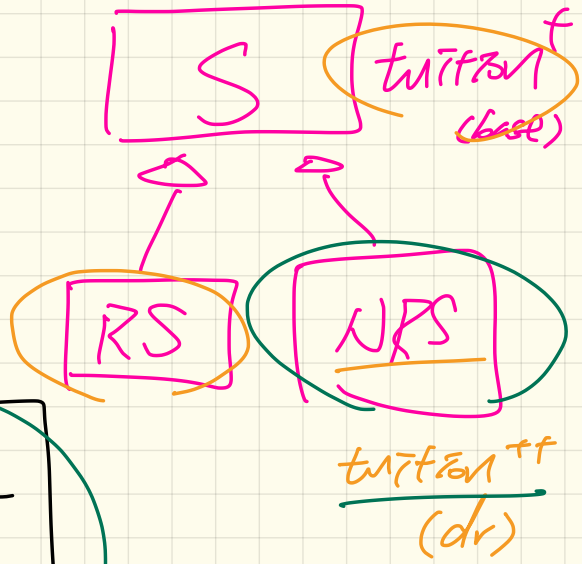
```
class STUDENT_MANAGEMENT_SYSTEM {
  ss: LINKED_LIST[STUDENT]
  add_s (s: STUDENT)
    do
      ss.extend (s)
    end
  get_student(i: INTEGER): STUDENT
    require 1 <= i and i <= ss.count
    do
      Result := ss[i]
    end
end
```

Possible DT of Result?


SS : [RS]
SS[1]
dr
S1:RS
X

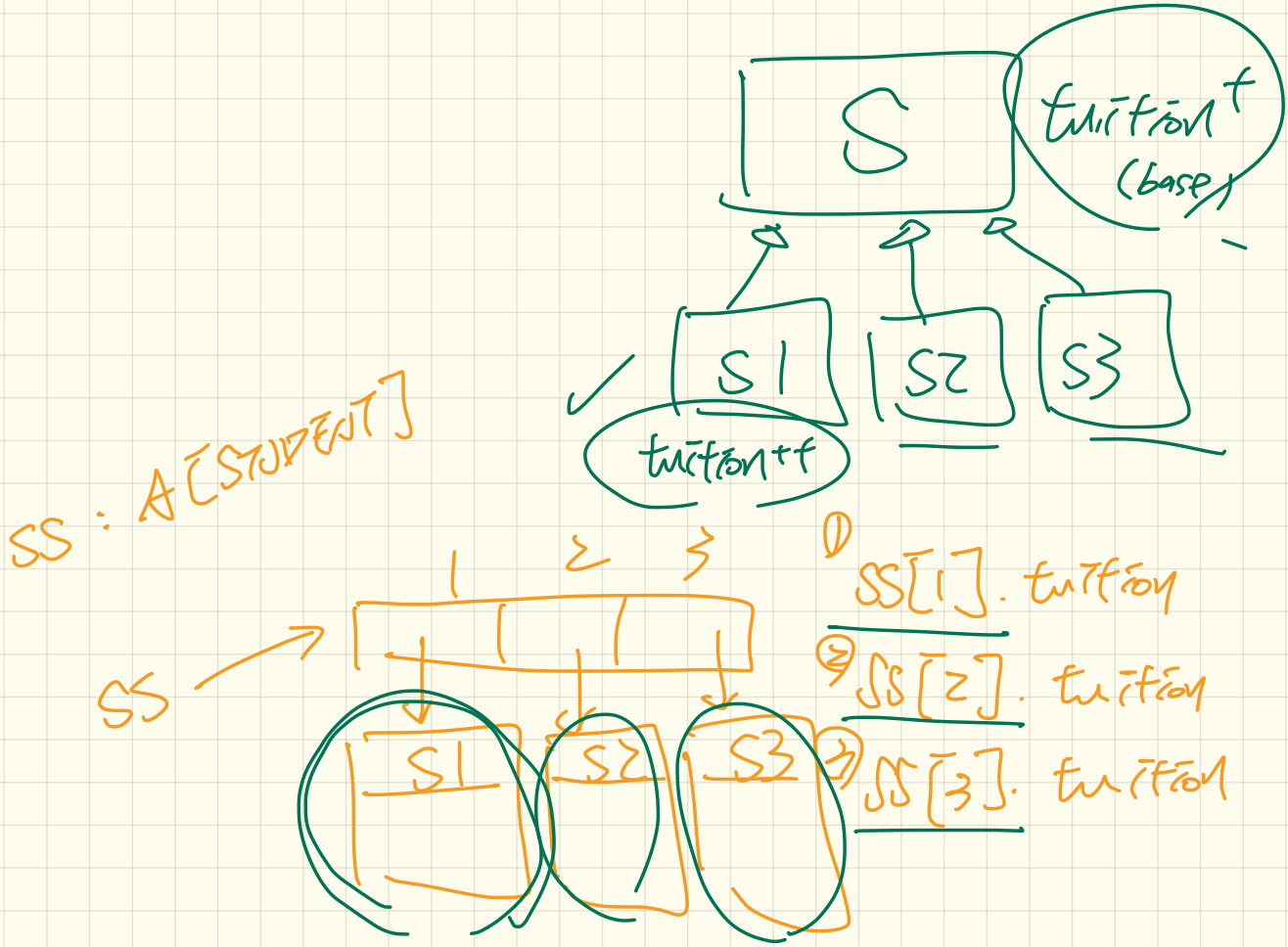
SS: ARRAY[STUDENT]

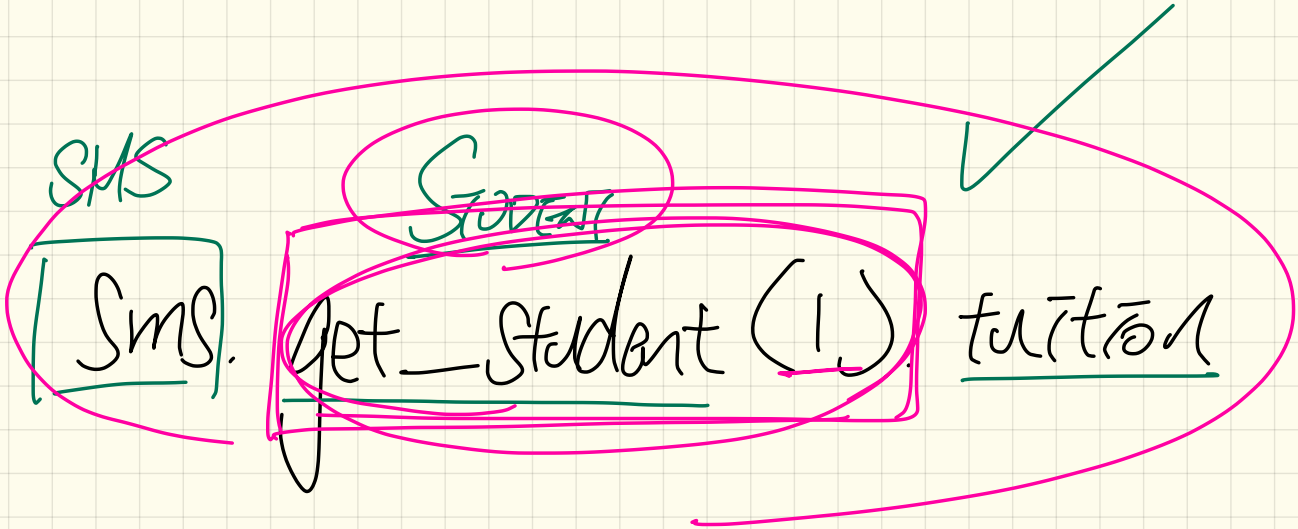




① SS[1].tuition

② SS[2].tuition


S
tuition
(base)
S1
S2
S3
tuition++
SS : A[STUDENT]
SS
1
2
3
SS[1]. tuition
SS[2]. tuition
SS[3]. tuition



1. Compile?

2. version of tuition run?