Tuesday Oct. 16

Lecture 10

- Lab 1 marks (programming) by Friday
- Midterm
  - Coverage: until Wednesday's lecture
  - Review: Friday 3 pm
- Lab 3
  - Tutorial Videos
  - Tomorrow's Lab Session

Type-Checking Rules

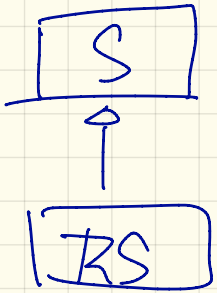
CODE	CONDITION TO BE TYPE CORRECT
`x := y`	`y`'s ***ST*** a **descendant** of `x`'s ***ST***
`x.f(y)`	Feature `f` defined in `x`'s ***ST*** `y`'s ***ST*** a **descendant** of `f`'s parameter's ***ST***
`z := x.f(y)`	Feature `f` defined in `x`'s ***ST*** `y`'s ***ST*** a **descendant** of `f`'s parameter's ***ST*** ***ST*** of `m`'s return value a **descendant** of `z`'s ***ST***
`check attached` `{C} y` **`then`** `...` **`end`**	`C` an **ancestor** or a **descendant** of `y`'s ***ST***
`check attached` `{C} y` **`as`** `temp` **`then`** `x := temp` **`end`**	`C` an **ancestor** or a **descendant** of `y`'s ***ST*** `C` a **descendant** of `x`'s ***ST***
`check attached` `{C} y` **`as`** `temp` **`then`** `x.f(temp)` **`end`**	`C` an **ancestor** or a **descendant** of `y`'s ***ST*** Feature `f` defined in `x`'s ***ST*** `C` a **descendant** of `f`'s parameter's ***ST***

x : STUDENT

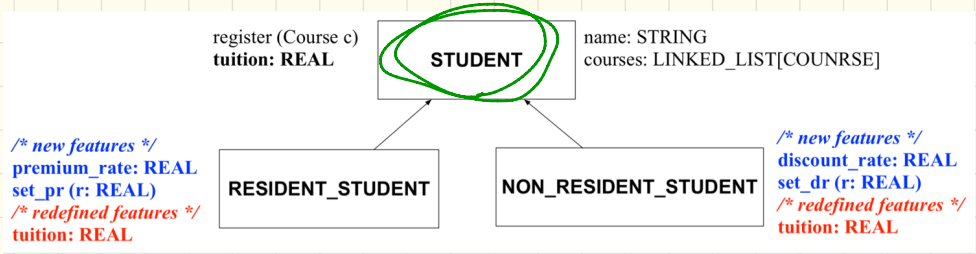
y : RS

x := y

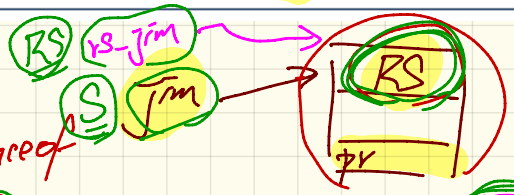
y := x



Cast: Motivation

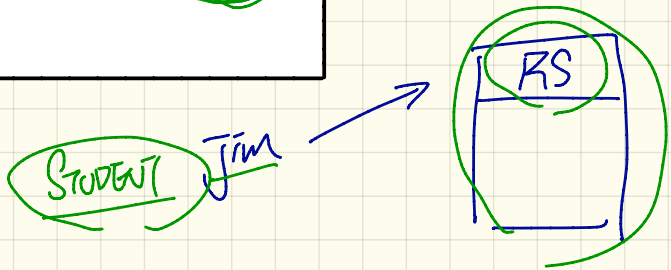



```
local jim: STUDENT  rs: RESIDENT_STUDENT
do create {RESIDENT_STUDENT} jim.make ("J. Davis")
  rs := jim
  rs.setPremiumRate(1.5)
```

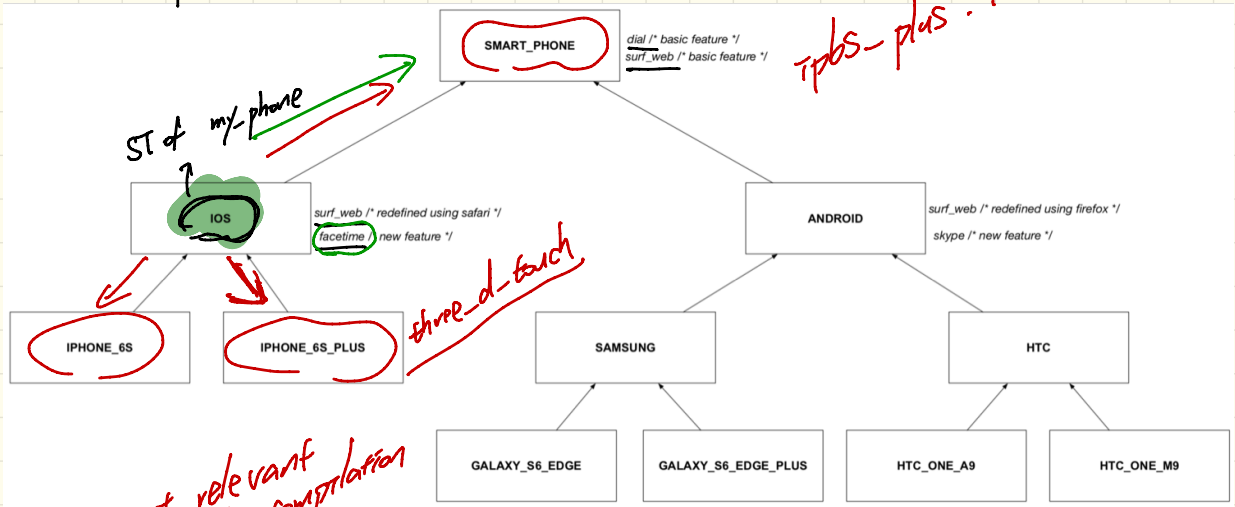


jim.set_pr X

```
check attached {RESIDENT_STUDENT} jim as rs_jim then
  rs := rs_jim
  rs.set_pr (1.5)
end
```

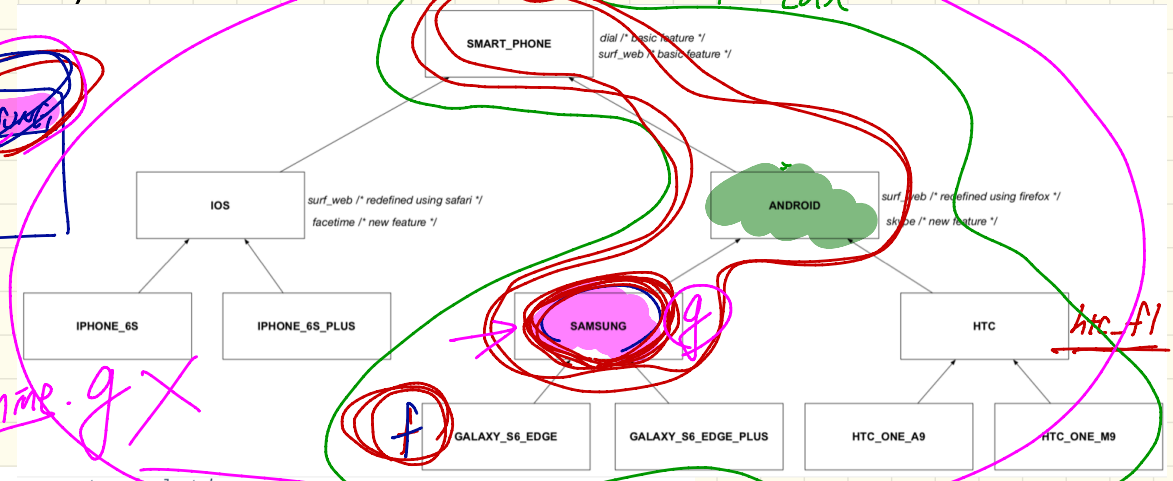


Compilable Cast : Upward or Downward




```
my_phone: IOS
create {IPHONE_6S_PLUS} my_phone.make
-- can only call features defined in IOS on myPhone
-- dial, surf_web, facetime ✓  three_d_touch, skype ×
check attached {SMART_PHONE} my_phone as sp then
  -- can now call features defined in SMART_PHONE on sp
  -- dial, surf_web ✓  facetime, three_d_touch, skype ×
end
check attached {IPHONE_6S_PLUS} my_phone as ip6s_plus then
  -- can now call features defined in IPHONE_6S_PLUS on ip6s_plus
  -- dial, surf_web, facetime, three_d_touch ✓  skype ×
end
```

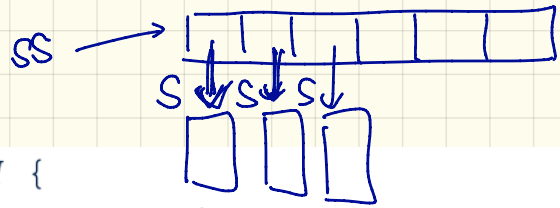
Compilable Cast May Fail at Runtime




```
test_smart_phone_type_cast_violation
local mine: ANDROID
  do create {SAMSUNG} mine.make
    -- ST of mine is ANDROID; DT of mine is SAMSUNG
    check attached {SMART_PHONE} mine as sp then ... end
    -- ST of sp is SMART_PHONE; DT of sp is SAMSUNG
    check attached {SAMSUNG} mine as samsung then ... end
    -- ST of samsung is SAMSUNG; DT of samsung is SAMSUNG
    check attached {HTC} mine as htc then ... end
    -- Compiles ∵ HTC is descendant of mine's ST (ANDROID)
    -- Assertion violation
    -- ∵ HTC is not ancestor of mine's DT (SAMSUNG)
    check attached {GALAXY_S6_EDGE} mine as galaxy then ... end
    -- Compiles ∵ GALAXY_S6_EDGE is descendant of mine's ST (ANDROI
    -- Assertion violation
    -- ∵ GALAXY_S6_EDGE is not ancestor of mine's DT (SAMSUNG)
end
```

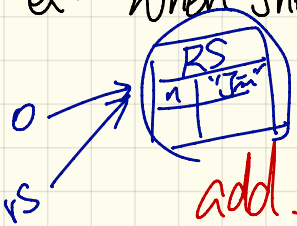
Assume this cast could succeed
⇒ galaxy.f would fail
∵ DT is SAMSUNG

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
```

Q: When should the call sms.add_rs(o) compile?



add_rs (rs: RS) do ... end

rs := o

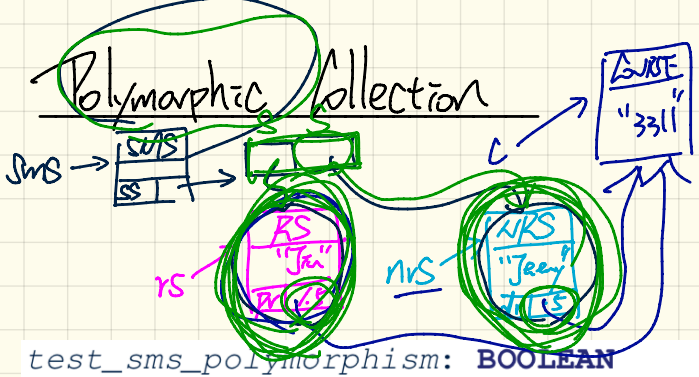
Feature Call Arguments : Client

```
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(s|)

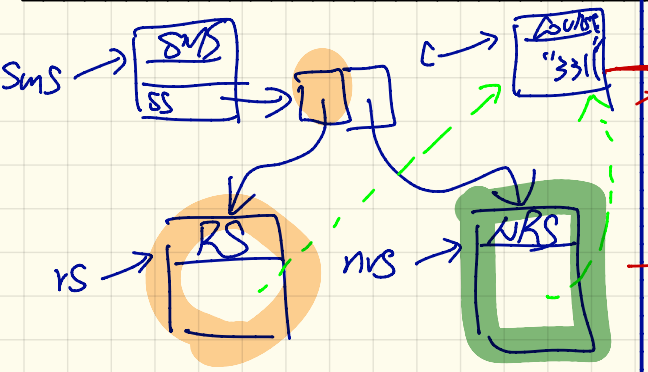
s




```
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 Value



Supplier

```
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
```

Result : STUDENT

Client

```
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
```

DT: RS

DT: NRS

Q: Possible DTs of Result?