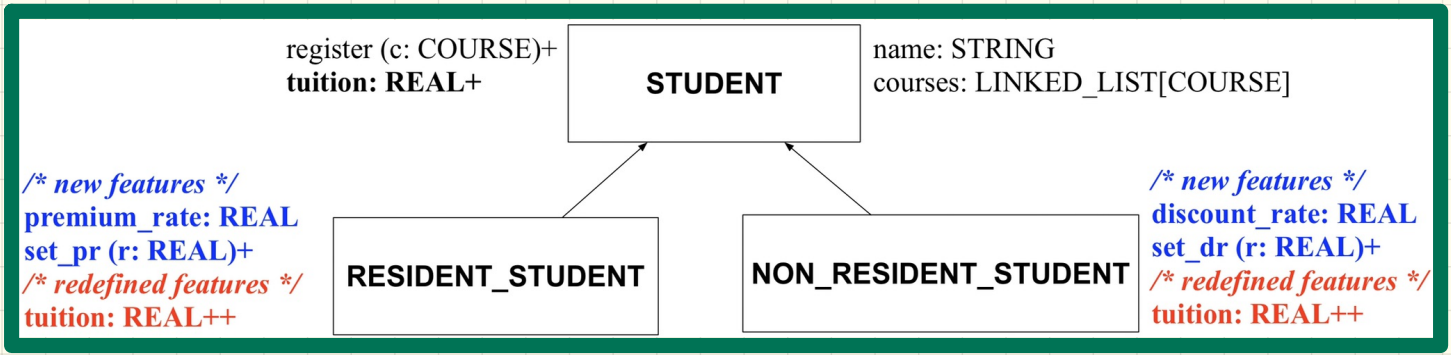
Lecture 7

Part 5

Static vs. Dynamic Types

Revisit: Inheritance for Code Reuse



Static Type vs. Dynamic Type

- In Java:

```
Student s = new Student("Alan");
Student rs = new ResidentStudent("Mark");
```

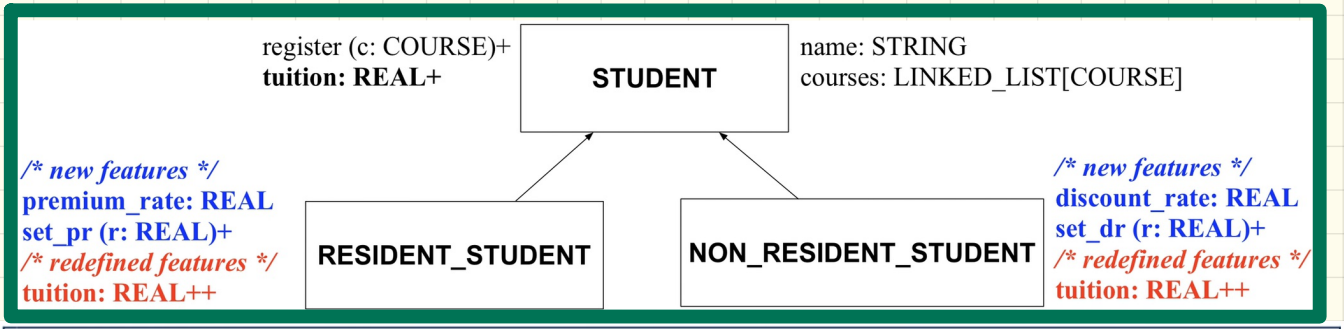
- In Eiffel:

```
local s: STUDENT
      rs: STUDENT
do create {STUDENT} s.make ("Alan")
   create {RESIDENT_STUDENT} rs.make ("Mark")
```

  - In Eiffel, the *dynamic type* can be omitted if it is meant to be the same as the *static type*:

```
local s: STUDENT
do create s.make ("Alan")
```

Student Classes (with Inheritance): Expectations




```
s1, s2, s3: STUDENT ; rs: RESIDENT_STUDENT ; nrs : NON_RESIDENT_STUDENT
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")
```

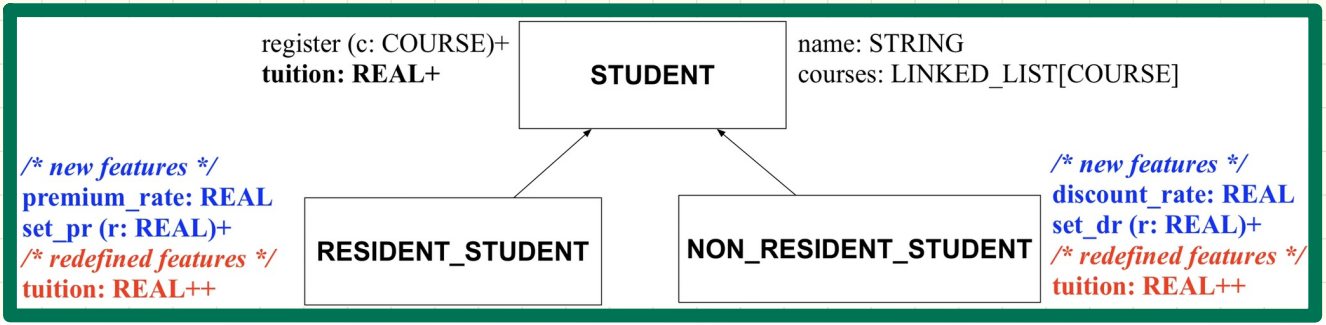
	name	courses	reg	tuition	pr	set_pr	dr	set_dr
s1								
s2								
s3								
rs								
nrs								

Lecture 7

Part 6

Intuition:
Polymorphism vs. Dynamic Binding

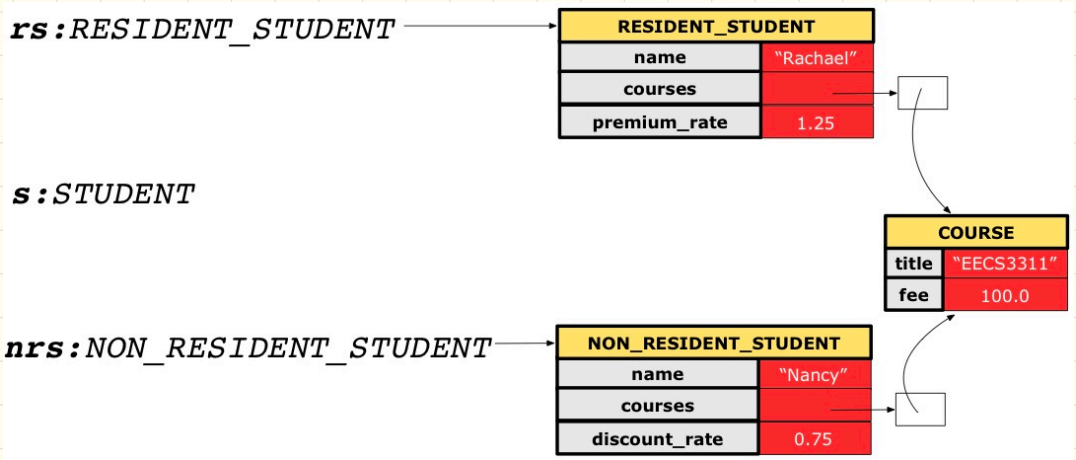
Polymorphism: Intuition




```
1  local
2    s: STUDENT
3    rs: RESIDENT_STUDENT
4  do
5    create s.make ("Stella")
6    create rs.make ("Rachael")
7    rs.set_pr (1.25)
8    s := rs /* Is this valid? */
9    rs := s /* Is this valid? */
```

Dynamic Binding: Intuition

```
1  local c : COURSE ; s : STUDENT
2        rs : RESIDENT_STUDENT ; nrs : NON_RESIDENT_STUDENT
3  do create c.make ("EECS3311", 100.0)
4     create {RESIDENT_STUDENT} rs.make("Rachael")
5     create {NON_RESIDENT_STUDENT} nrs.make("Nancy")
6     rs.set_pr(1.25); rs.register(c)
7     nrs.set_dr(0.75); nrs.register(c)
8     s := rs;  ; check s.tuition = [redacted] end
9     s := nrs; ; check s.tuition = [redacted] end
```

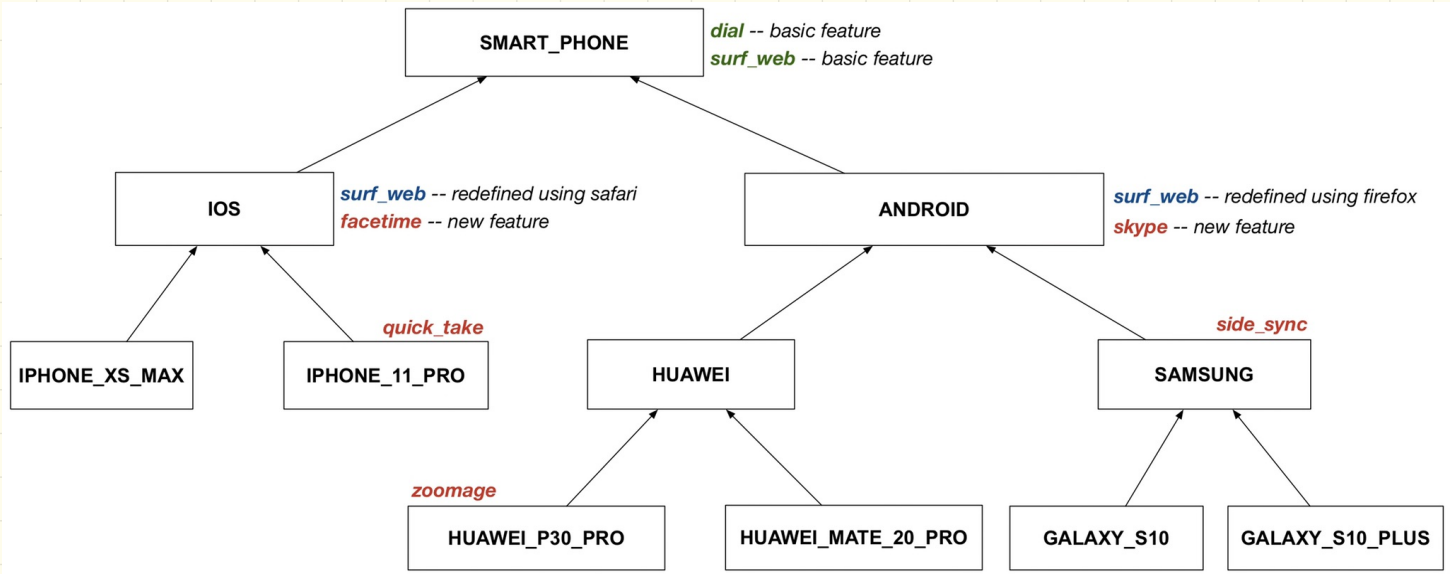


Lecture 7

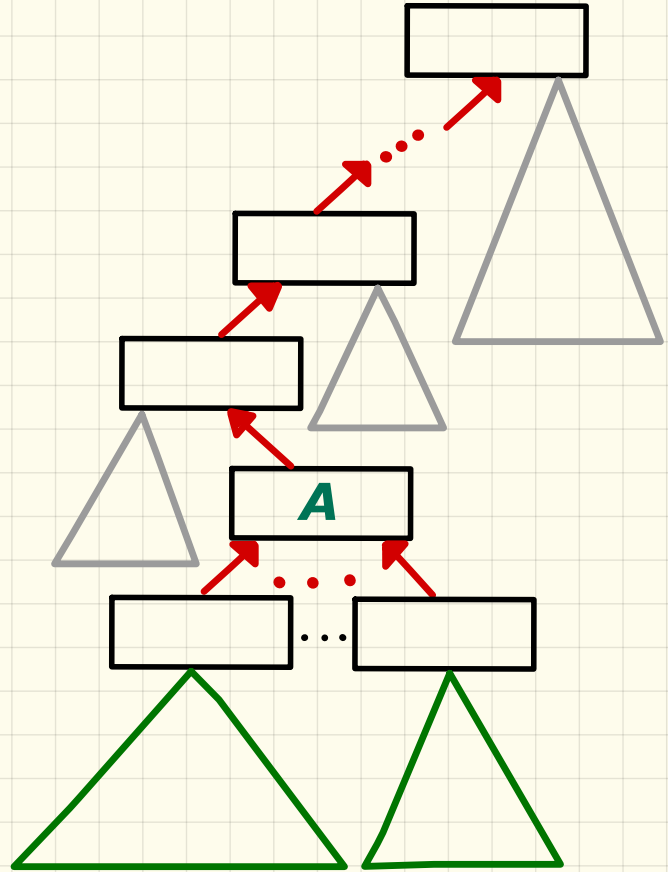
Part 7

Multi-Level Inheritance Hierarchy

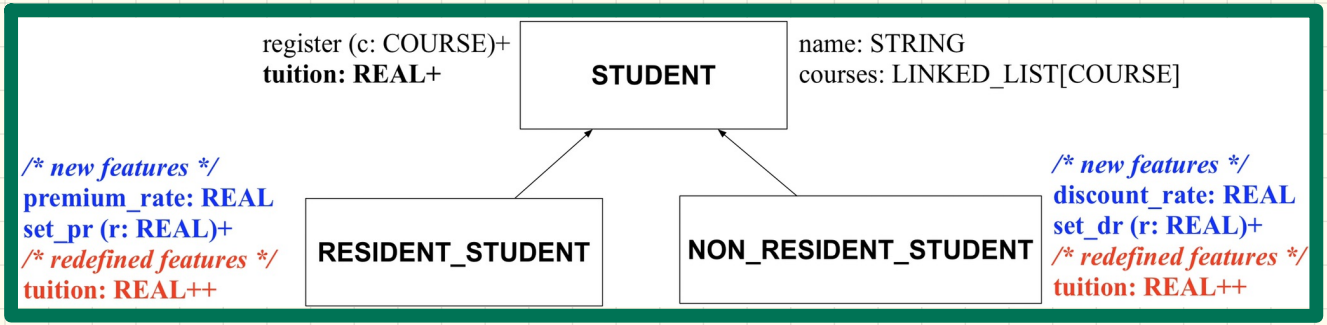
Multi-Level Inheritance Hierarchy of Smartphones



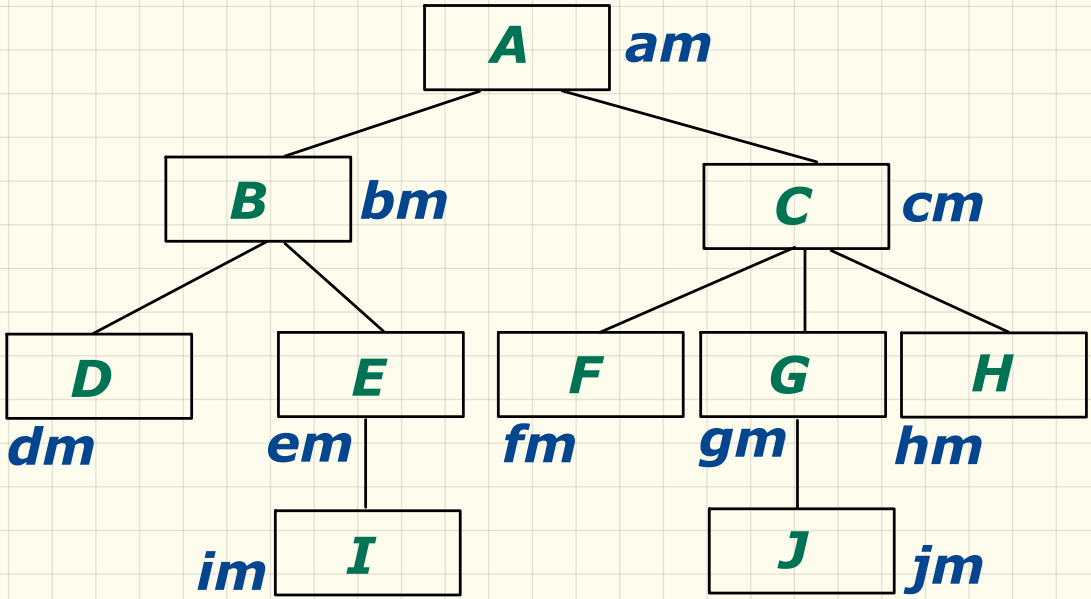
Ancestors, Expectations, Descendants, and Code Reuse



Substitutions by a Descendant Type

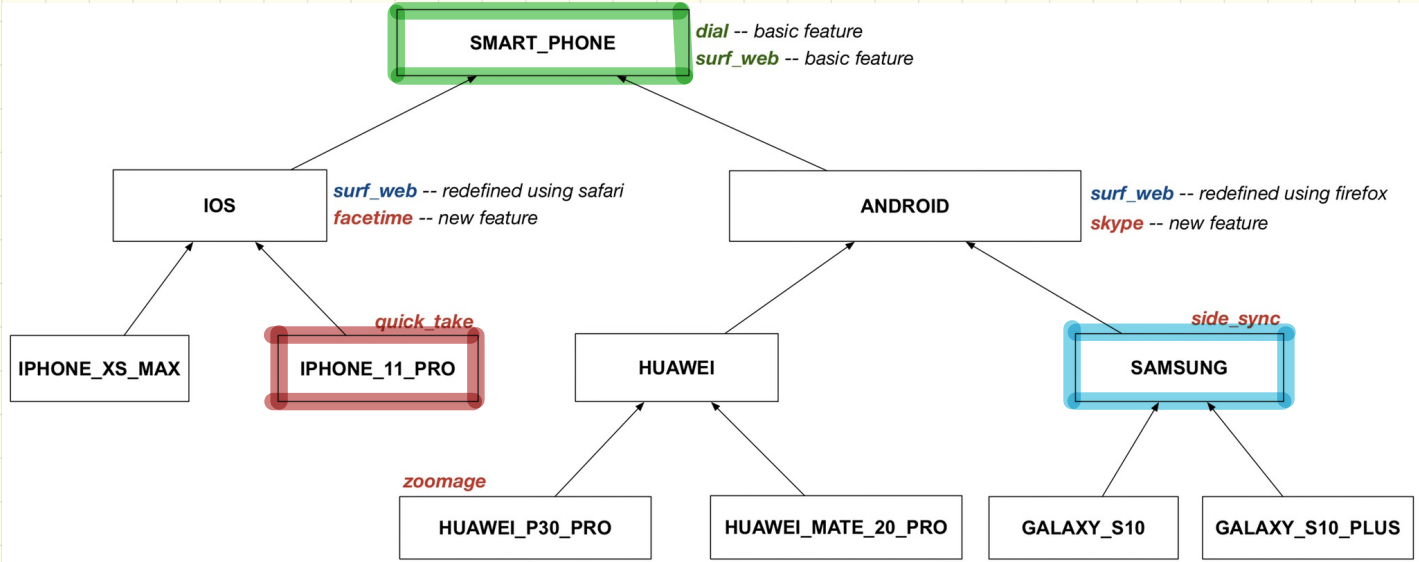


Inheritance Forms a Type Hierarchy (1)



	ancestors	expectations	descendants
B			
G			
J			

Inheritance Forms a Type Hierarchy (2)



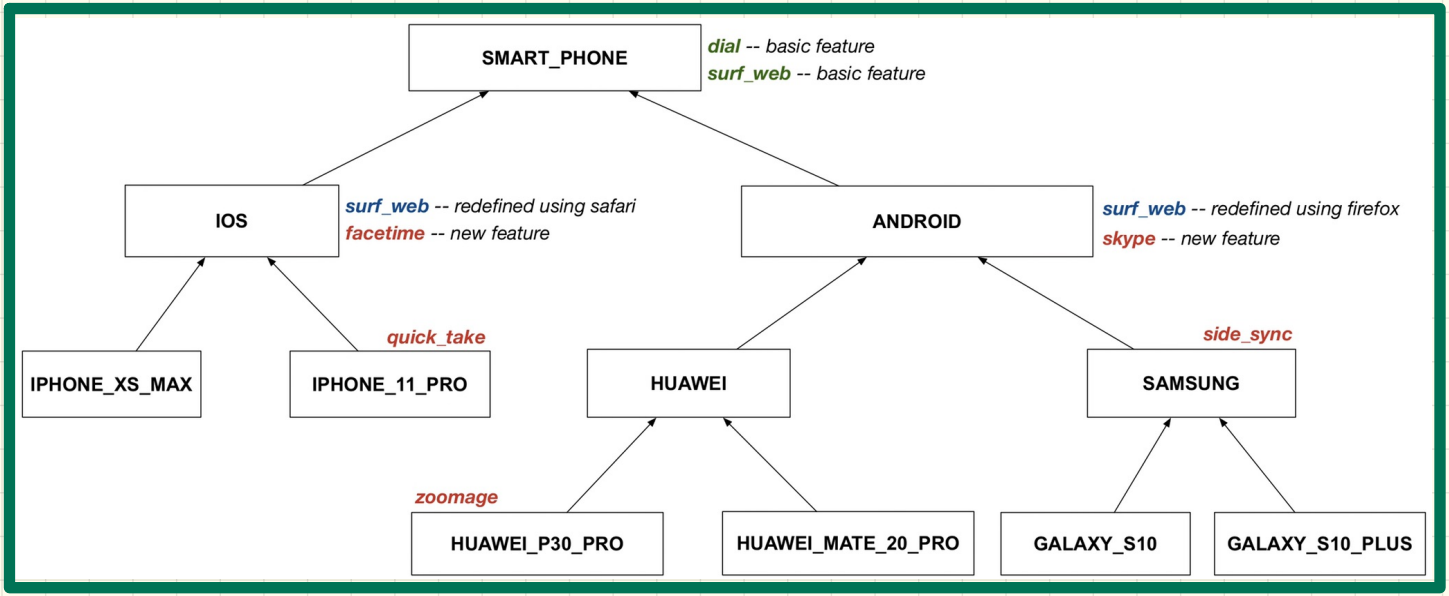
	ancestors	expectations	descendants

Lecture 7

Part 8

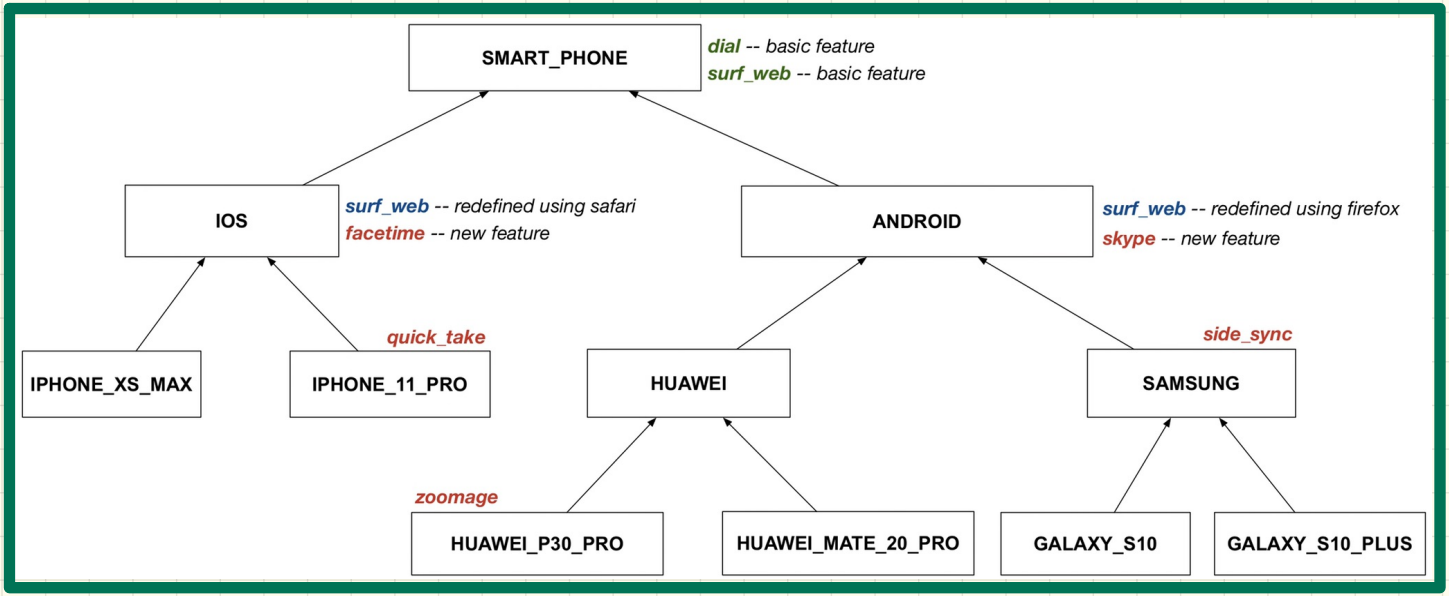
Rules of Substitutions via Assignments

Rules of Substitutions (1)



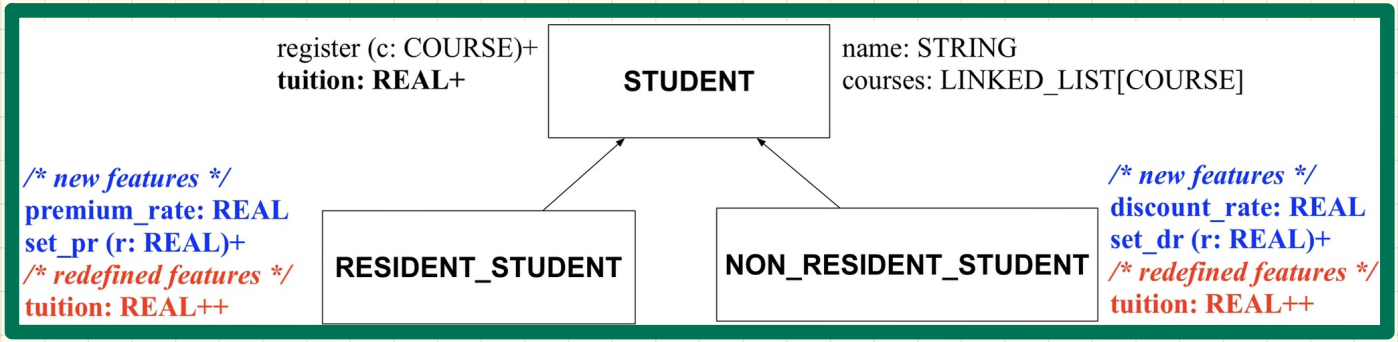
mp_1: **IOS**
p1: **I_PHONE_XS_MAX**
p2: **I_PHONE_11_PRO**

Rules of Substitutions (2)



mp_2: **IOS**
p3: **SMART_PHONE**

Reference Variables: Static Type



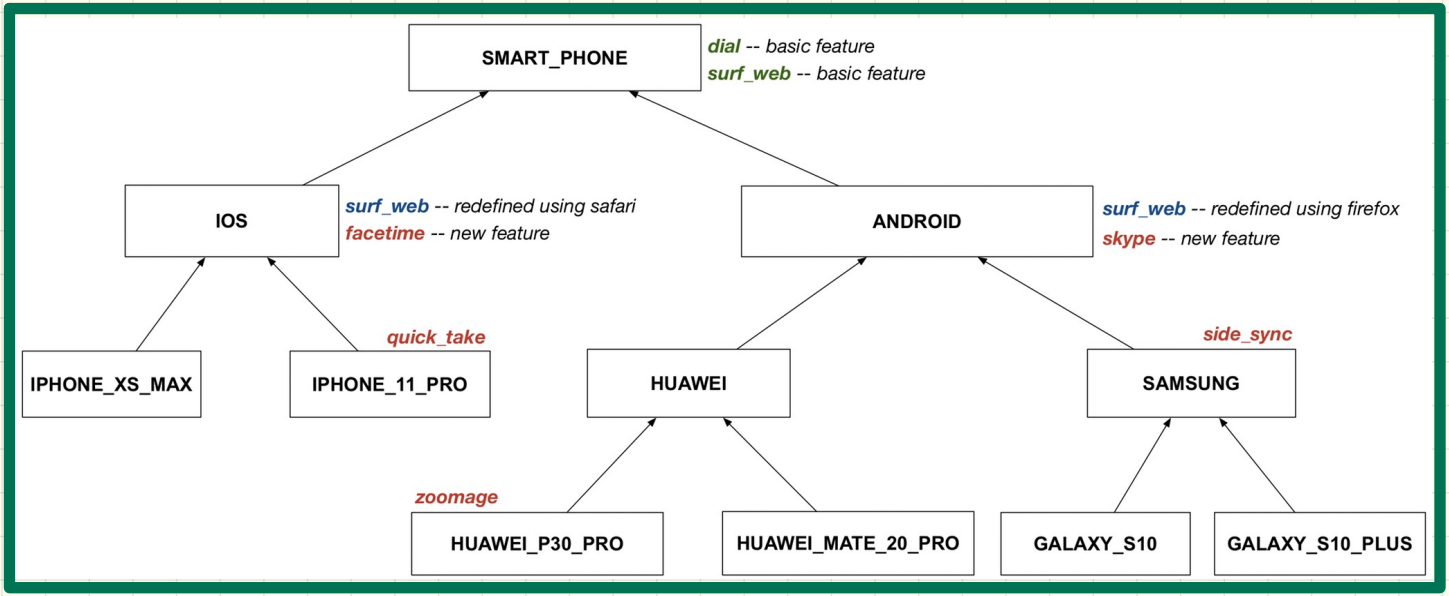
Design 1:

jim: **STUDENT**

Design 2:

jim: **RESIDENT_STUDENT**

Reference Variables: Static Type



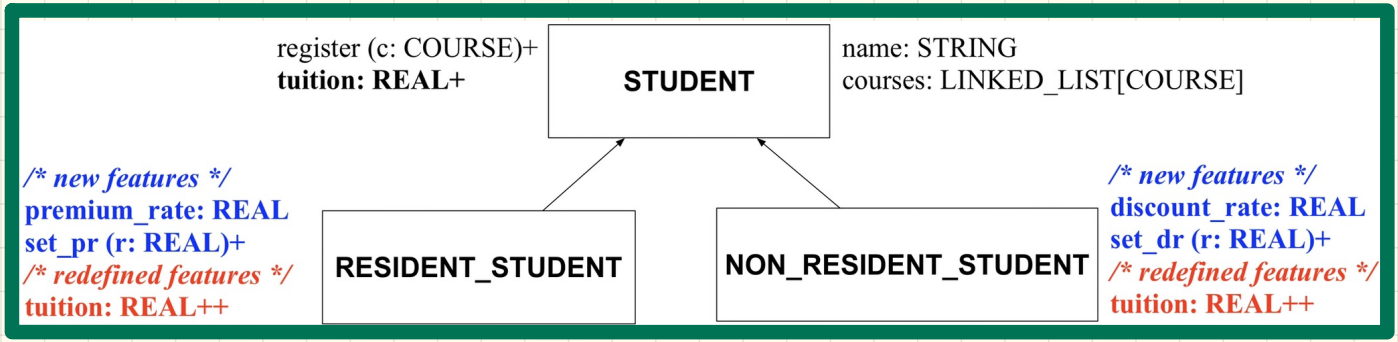
Design 1:

mp: **SMART_PHONE**

Design 2:

mp: **IOS**

Change of Dynamic Type (1)



Design 1:

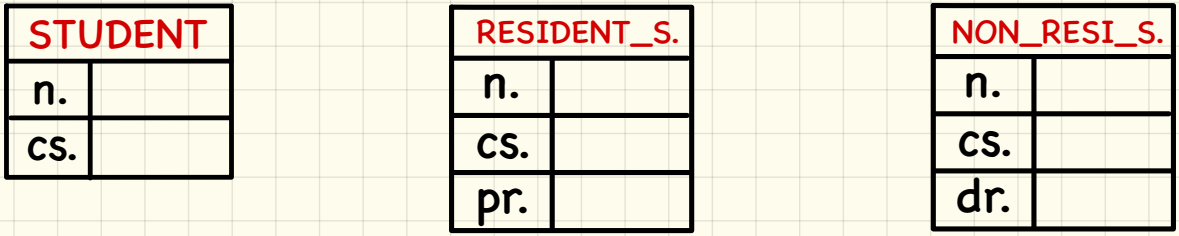
jim: **STUDENT**

Design 2:

jim: **RESIDENT_STUDENT**

Change of Dynamic Type (2)

```
1  test_polymorphism_students
2   local
3     jim: STUDENT
4     rs: RESIDENT_STUDENT
5     nrs: NON_RESIDENT_STUDENT
6   do
7     create {STUDENT} jim.make ("J. Davis")
8     create {RESIDENT_STUDENT} rs.make ("J. Davis")
9     create {NON_RESIDENT_STUDENT} nrs.make ("J. Davis")
10    jim := rs
11    rs := jim
12    jim := nrs
13    rs := jim
14  end
```



Testing of Dynamic Binding

RESIDENT_S.	
n.	
cs.	
pr.	

NON_RESI_S.	
n.	
cs.	
dr.	

STUDENT	
n.	
cs.	

```
class STUDENT
create make
feature -- Attributes
  name: STRING
  courses: LINKED_LIST[COURSE]
feature -- Commands that can be used as constructors.
  make (n: STRING) do name := n ; create courses.make end
feature -- Commands
  register (c: COURSE) do courses.extend (c) end
feature -- Queries
  tuition: REAL
    local base: REAL
    do base := 0.0
       across courses as c loop base := base + c.item.fee end
       Result := base
    end
end
```

```
test_dynamic_binding_students: BOOLEAN
  local
    jim: STUDENT
    rs: RESIDENT_STUDENT
    nrs: NON_RESIDENT_STUDENT
    c: COURSE
  do
    create c.make ("EECS3311", 500.0)
    create {STUDENT} jim.make ("J. Davis")
    create {RESIDENT_STUDENT} rs.make ("J. Davis")
    rs.register (c)
    rs.set_pr (1.5)
    jim := rs
    Result := jim.tuition = 750.0
    check Result end
    create {NON_RESIDENT_STUDENT} nrs.make ("J. Davis")
    nrs.register (c)
    nrs.set_dr (0.5)
    jim := nrs
    Result := jim.tuition = 250.0
end
```

COURSE	
t.	
fee	

```
class
  RESIDENT_STUDENT
inherit
  STUDENT
    redefine tuition end
create make
feature -- Attributes
  premium_rate: REAL
feature -- Commands
  set_pr (r: REAL) do premium_rate := r end
feature -- Queries
  tuition: REAL
    local base: REAL
    do base := Precursor ; Result := base * premium_rate end
end
```

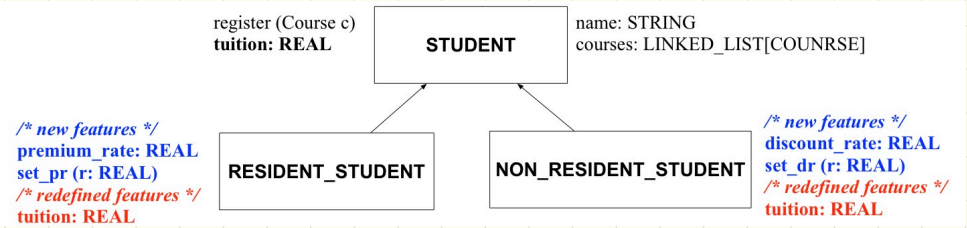
```
class
  NON_RESIDENT_STUDENT
inherit
  STUDENT
    redefine tuition end
create make
feature -- Attributes
  discount_rate: REAL
feature -- Commands
  set_dr (r: REAL) do discount_rate := r end
feature -- Queries
  tuition: REAL
    local base: REAL
    do base := Precursor ; Result := base * discount_rate end
end
```

Lecture 7

Part 9

Type Casting

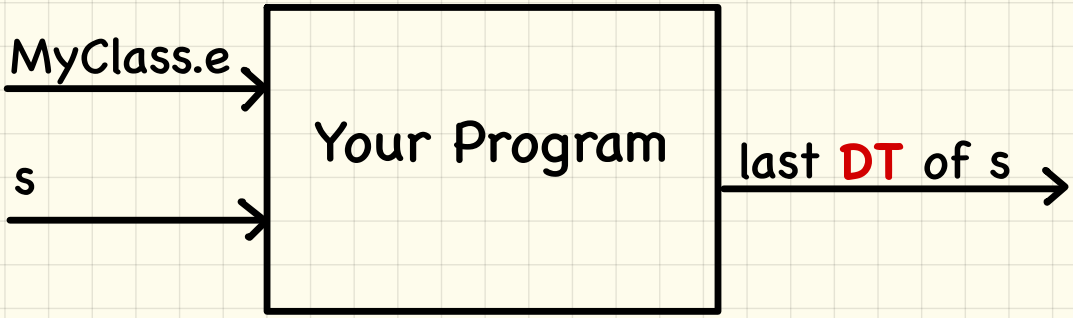
Type Cast: Motivation




```
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_S.	
n.	
cs.	
pr.	

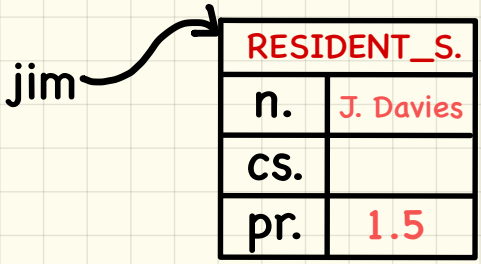
Inferring the DT of a Variable is Undecidable



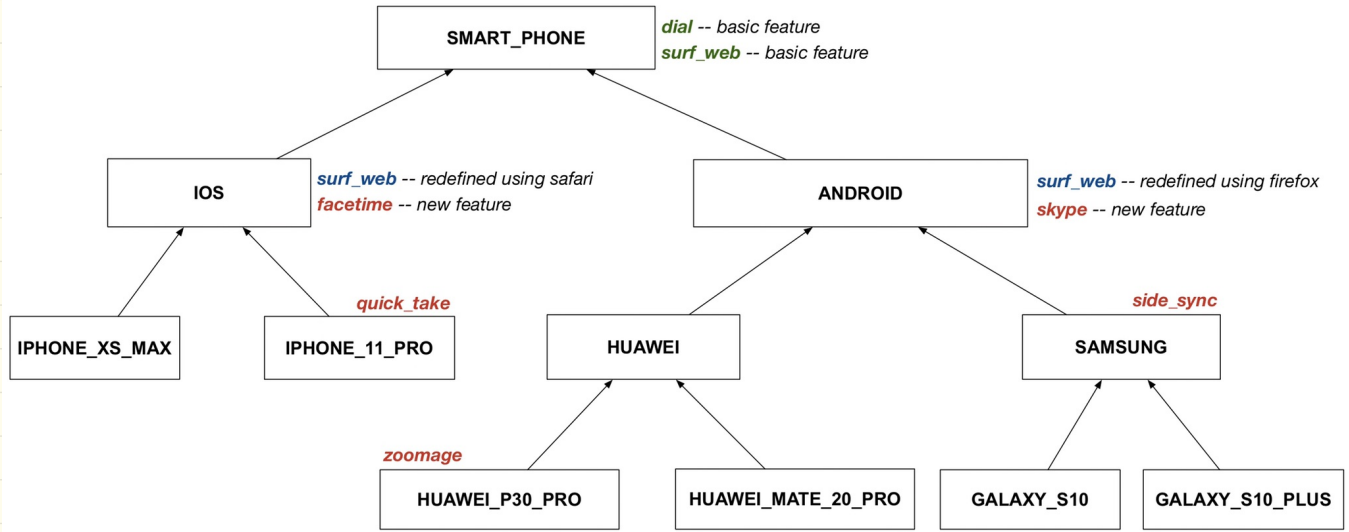

```
class MyClass
  make
    local
      s: STUDENT
    do
      create {RESIDENT_STUDENT} s.make
    end
end
```

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
```

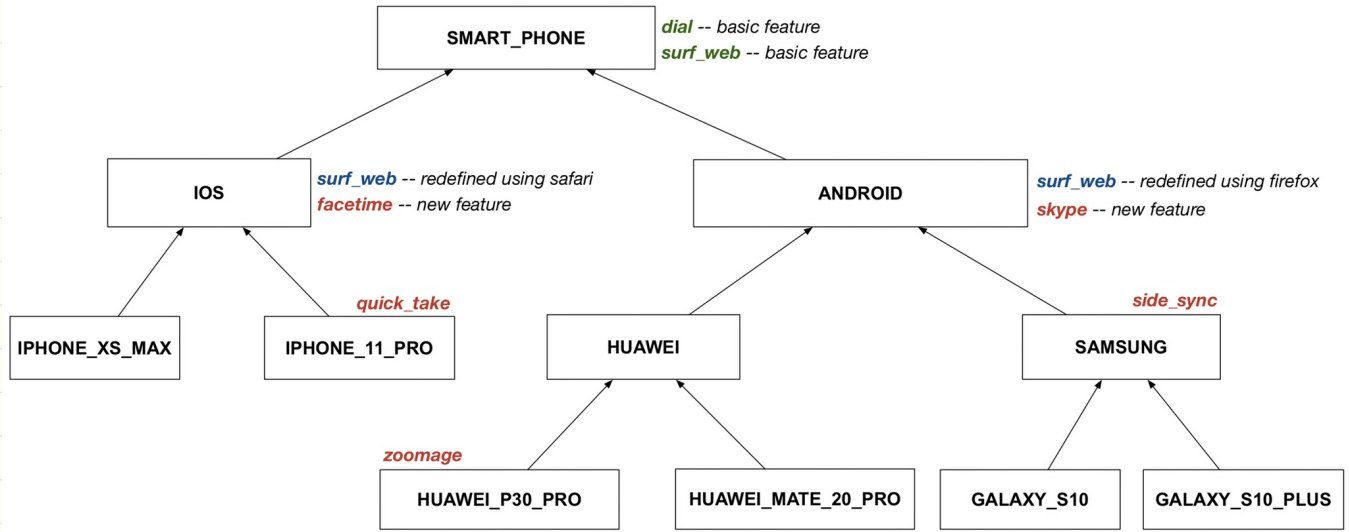


Violation-Free Cast: Upwards or Downwards (1)



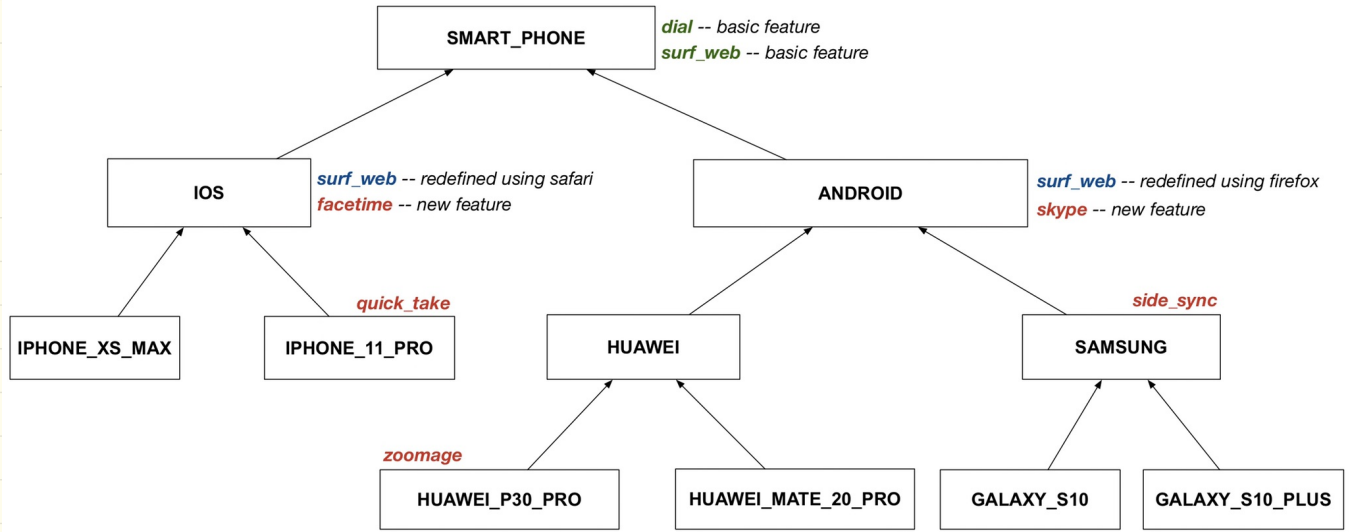

```
my_phone: IOS
create {IPHONE_11_PRO} my_phone.make
  -- can only call features defined in IOS on myPhone
  -- dial, surf_web, facetime ●  quick_take, skype, side_sync, zoomage ●
check attached {SMART_PHONE} my_phone as sp then
  -- can now call features defined in SMART_PHONE on sp
  -- dial, surf_web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



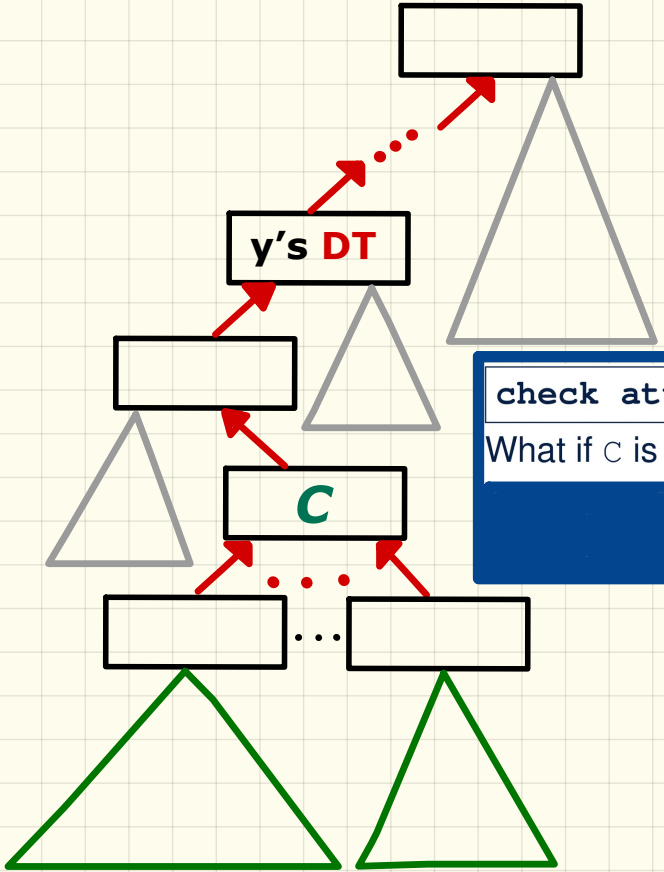

```
my_phone: IOS
create {IPHONE_11_PRO} my_phone.make
  -- can only call features defined in IOS on myPhone
  -- dial, surf_web, facetime ●  quick_take, skype, side_sync, zoomage ●
check attached {SMART_PHONE} my_phone as sp then
  -- can now call features defined in SMART_PHONE on sp
  -- dial, surf_web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

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
```

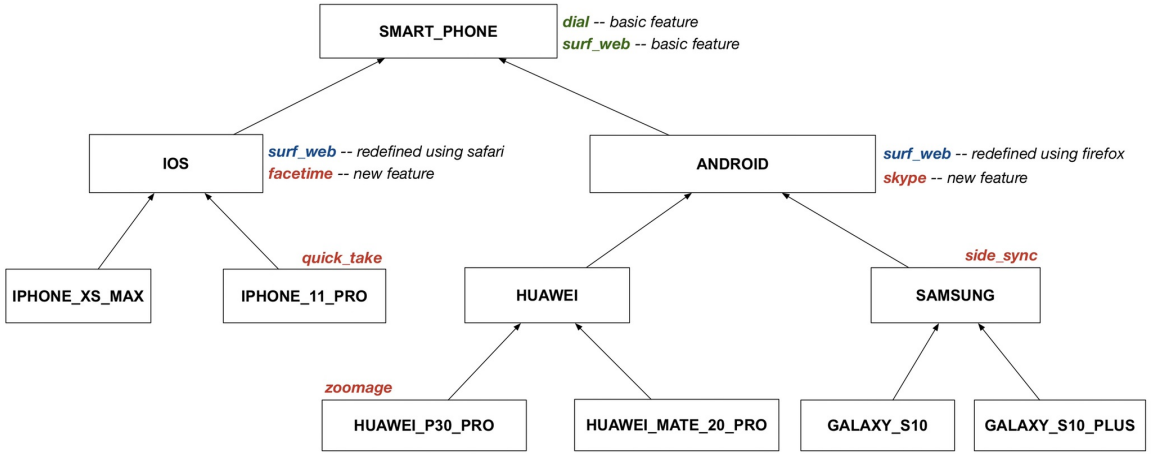
Ancestors, Expectations, Descendants, and Code Reuse



`check attached {C} y then ... end` *always compiles*

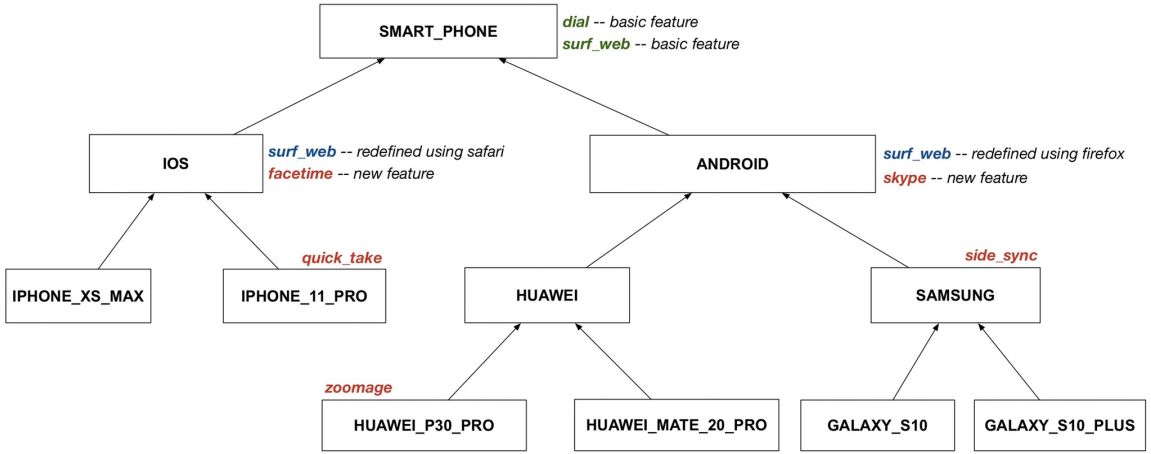
What if `C` is not an **ancestor** of `y`'s ***DT***?

Cast Violation at Runtime (1)



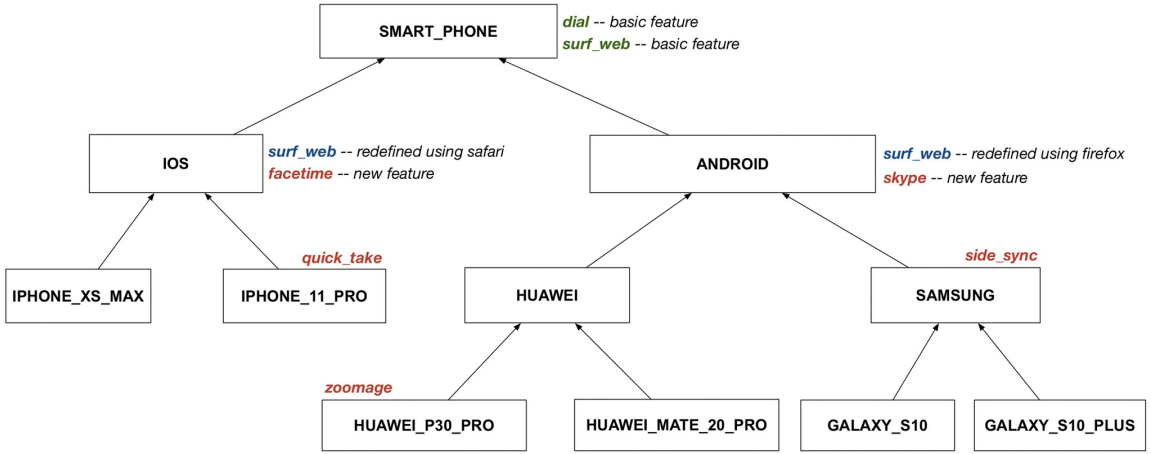

```
test_smart_phone_type_cast_violation
  local mine: ANDROID
  do create {HUAWEI} mine.make
    -- ST of mine is ANDROID; DT of mine is HUAWEI
    check attached {SMART_PHONE} mine as sp then ... end
    -- ST of sp is SMART_PHONE; DT of sp is HUAWEI
    check attached {HUAWEI} mine as huawei then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



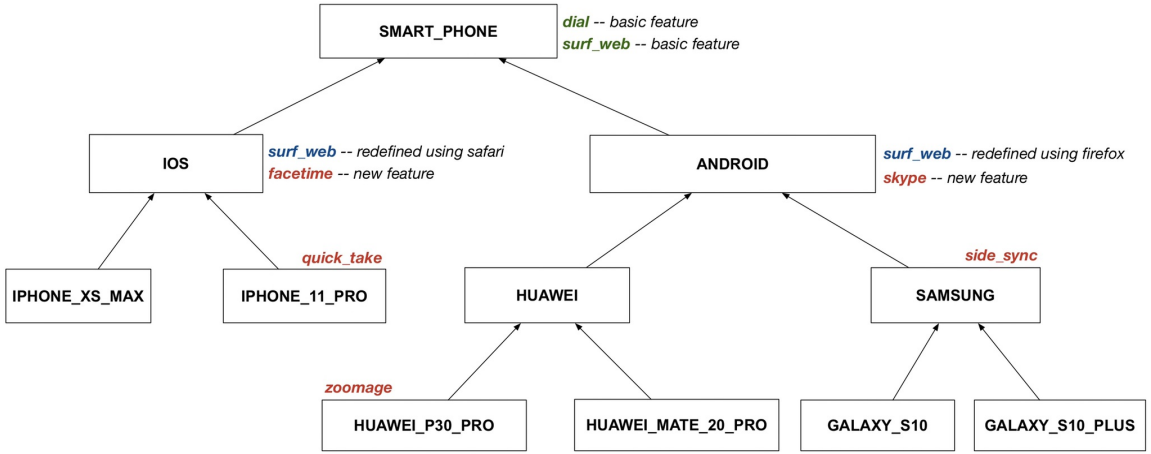

```
test_smart_phone_type_cast_violation
  local mine: ANDROID
  do create {HUAWEI} mine.make
    -- ST of mine is ANDROID; DT of mine is HUAWEI
    check attached {SMART_PHONE} mine as sp then ... end
    -- ST of sp is SMART_PHONE; DT of sp is HUAWEI
    check attached {HUAWEI} mine as huawei then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

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
```

Lecture 7

Part 10

Polymorphic Routine Arguments

Feature Call Parameters: 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)

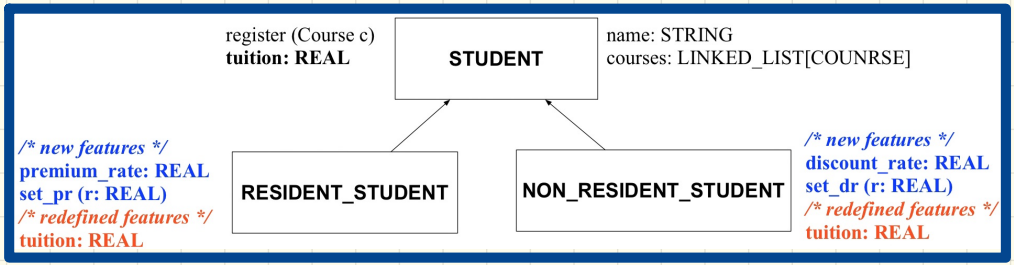
Lecture 7

Part 11

Polymorphic Collections

Polymorphic Collection

SMS	
ss	



RESIDENT_S.	
n.	
cs.	
pr.	

NON_RESI_S.	
n.	
cs.	
dr.	

COURSE	
t.	
fee	

```
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
```

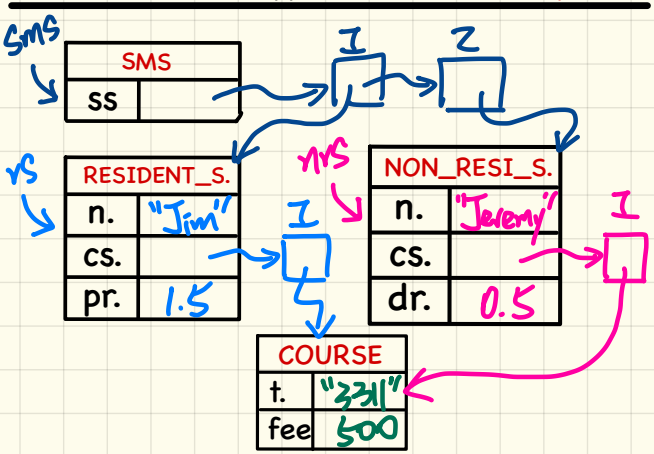
```
class STUDENT_MANAGEMENT_SYSETM
  students: LINKED_LIST[STUDENT]
  add_student(s: STUDENT)
    do
      students.extend (s)
    end
  registerAll (c: COURSE)
    do
      across
        students as s
      loop
        s.item.register (c)
      end
    end
end
```

Lecture 7

Part 12

Polymorphic Return Values

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
```

```
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
```

Possible DT of Result?