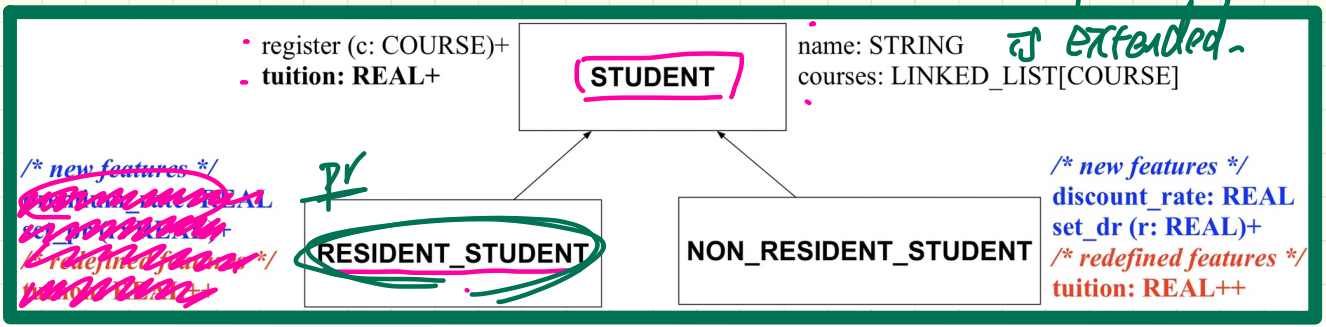
EECS3311 Software Design (Fall 2020)

Q&A - Lecture Series W7

Monday, November 2

Polymorphism: Intuition

What if at a later point, RS is extended.




```
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? */
```

still is invalid.

After the change.

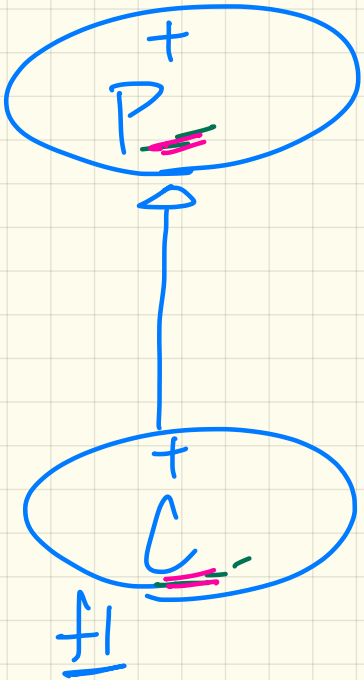
exp(RS) = exp(STU). ✓

not valid

∵ exp(RS) ⊄ exp(STU)

pr pr

What happens if the RESIDENGT_STUDENT does not declare any additional attributes and routines(it just simply inherits from STUDENT), will line 9 compiles? Because now the expectation of STUDENT and RESIDENT_STUDENT will be the same?



i : INTEGER

op : P

oc : C

create op.make

create oc.make

op := oc ✓

oc := op ✗

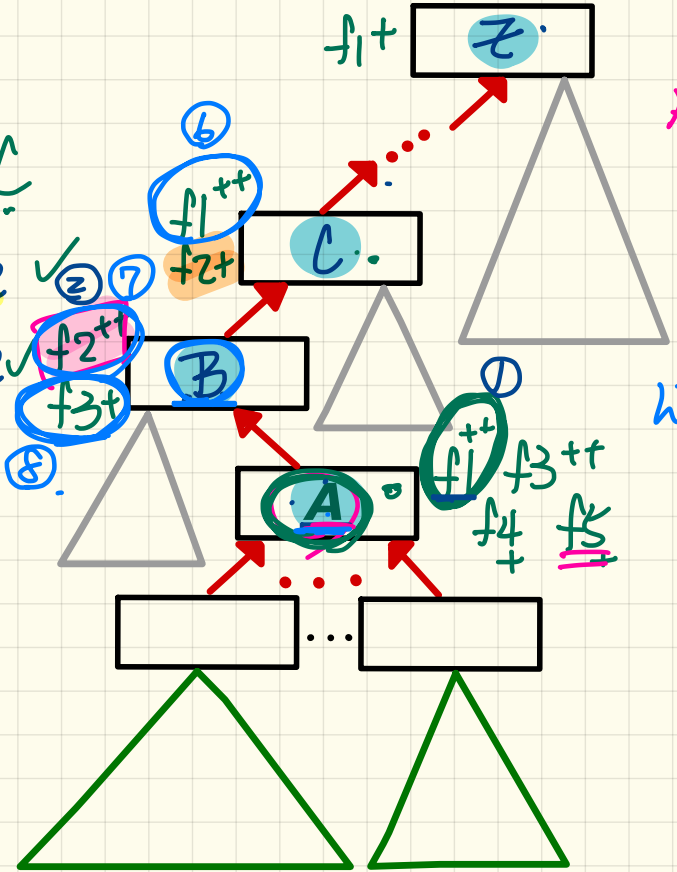
Ancestors, Expectations, Descendants, and Code Reuse

oc : C

ob : B

create {A} oc.make ST: C

create {B} ob.make DT: A



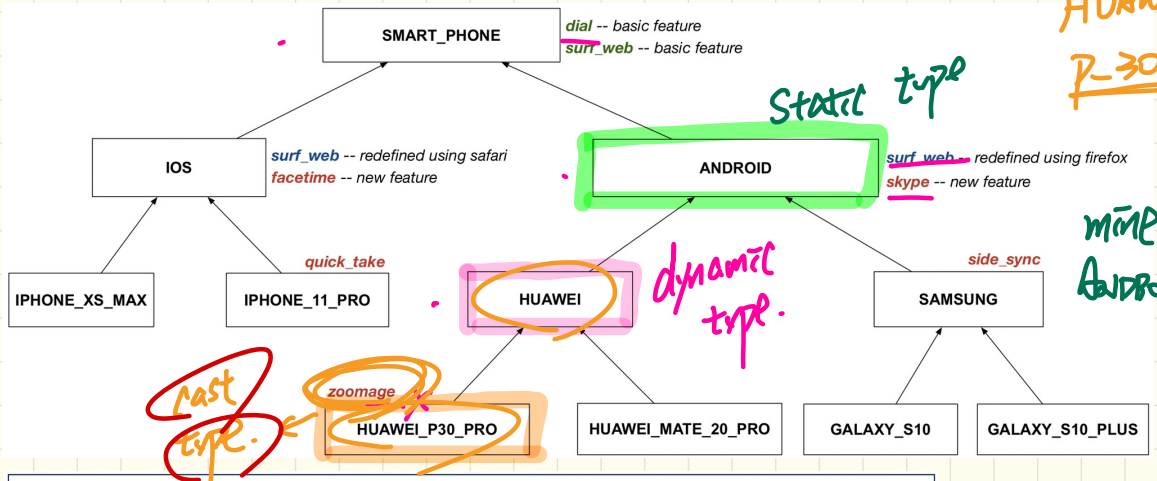
How many versions of f2 do we have?

Which exprs do not compile?

① oc.f1	⑤ ob.f1
② oc.f2	⑦ ob.f2
③ oc.f3	⑧ ob.f3
④ oc.f4	⑨ ob.f4
⑤ oc.f5	⑩ ob.f5

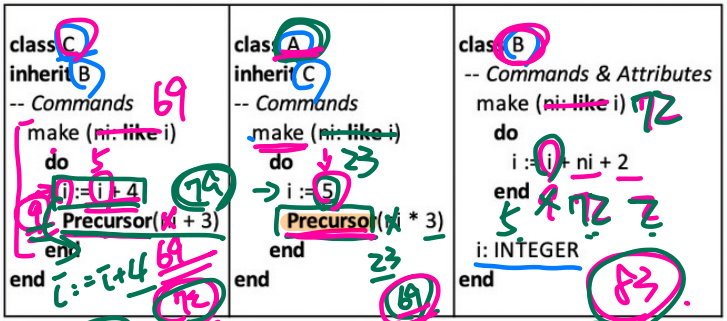
↳ does not compile even though DT A supports it.

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
```

Consider the following 3 classes:



```
class C
inherit B
-- Commands
  make (ni: like i)
    do
      i := i + 4
      Precursor(ni + 3)
    end
end
```

```
class A
inherit C
-- Commands
  make (ni: like i)
    do
      i := 5
      Precursor(ni * 3)
    end
end
```

```
class B
-- Commands & Attributes
  make (ni: like i)
    do
      i := i + ni + 2
    end
  i: INTEGER
end
```

In each of the above classes:

- The `make` command is declared as the constructor.
- Where applicable, a **redefine** clause, declaring that the inherited `make` command is redefined/overridden, is omitted.

Now consider the following variable declaration:

obj: A

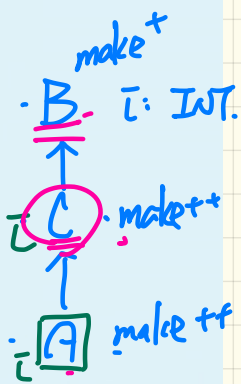
After the following initialization:

create obj.make(23)

What's the value of `**obj.i**`? Enter an integer value in the answer box.

Note. There is another similar question, but consider this question **independently**.

Answer: 83.



super.f1

super.super.f1 X

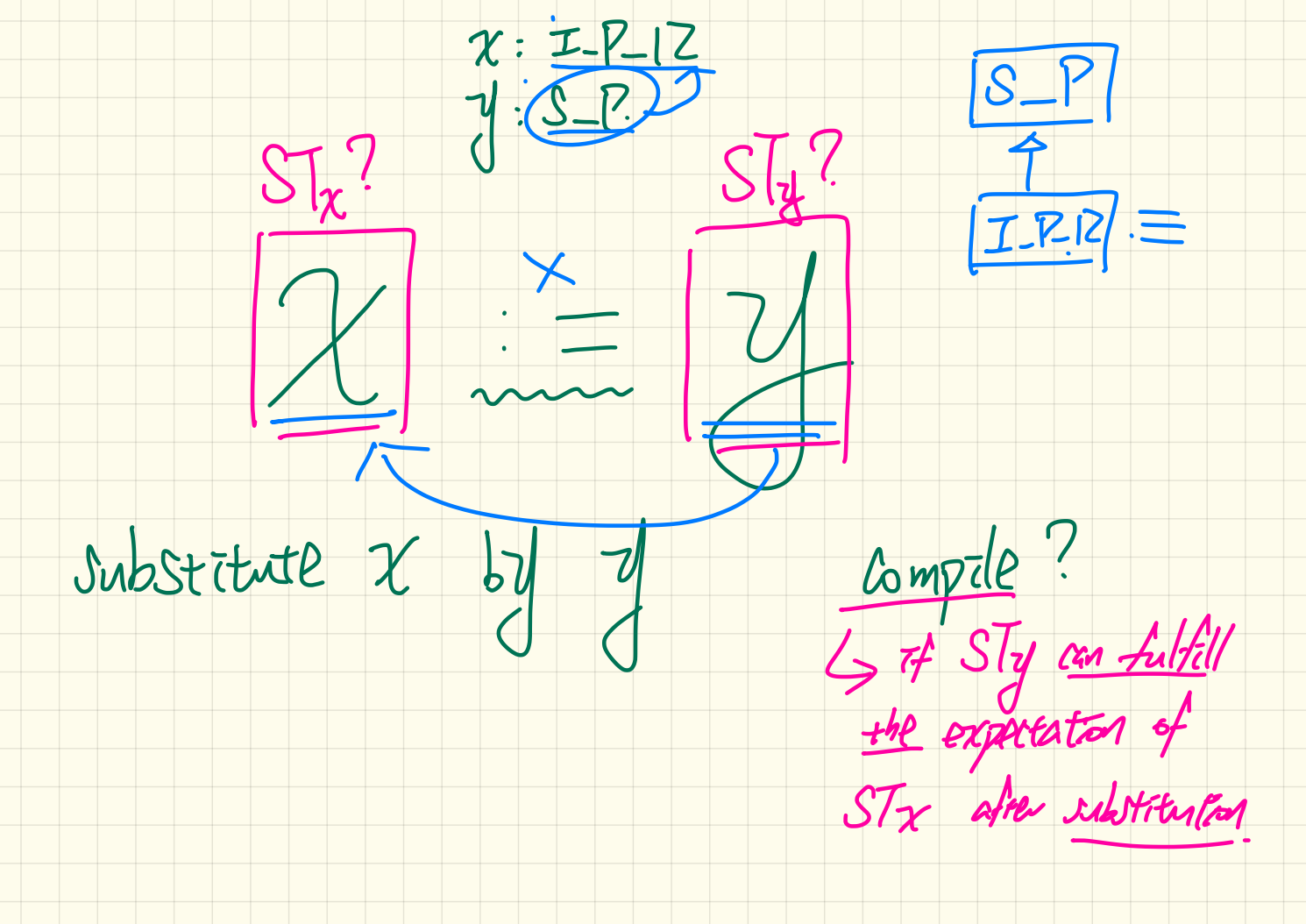
Precursor . Precursor

create {A} obj. make(23)

make

→ Precursor(..)

→ Precursor(...) ✓

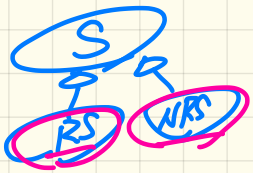

x : I_P_12
y : S_P
S_P
I_P_12 ≡
ST_x ?
ST_y ?
x
≡
y
substitute x by y
Compile ?
↳ if ST_y can fulfill the expectation of ST_x after substitution.

2nd Design Attempt

```
class
   STUDENT
create
   make
feature -- attribures
   courses: LINKED_LIST[COURSE]
   kind: INTEGER
   premiumRate: REAL
   discountRate: REAL
feature -- command
   make (kind: INTEGER)
     do
       kind := a_kind
     end
   ...
end
```

kind = 1 or 2

not violating cohesion




```
get_tuition: REAL
   local
      tuition: REAL
   do
      across courses is c loop
         tuition := tuition + c.fee
      end
      if kind = 1 then
         Result := tuition * premiumRate
      elseif kind = 2 then
         Result := tuition * discountRate
      end
   end
```

```
register (c: COURSE)
   local
      max: INTEGER
   do
      if kind = 1 then MAX := 6
      elseif kind = 2 then MAX := 4
      end
      if courses.count = MAX then -- Error
      else courses.extend (c)
      end
   end
```