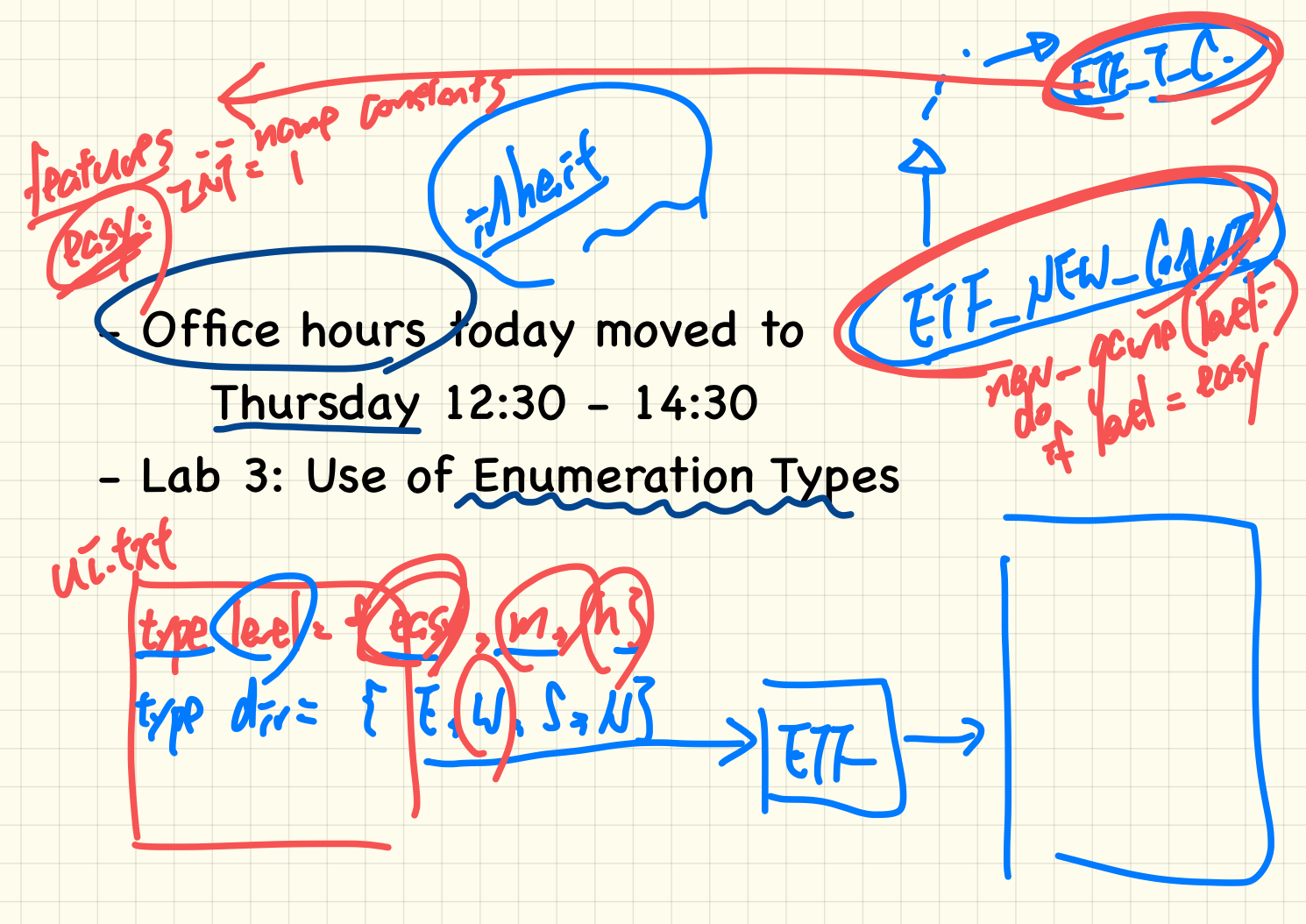
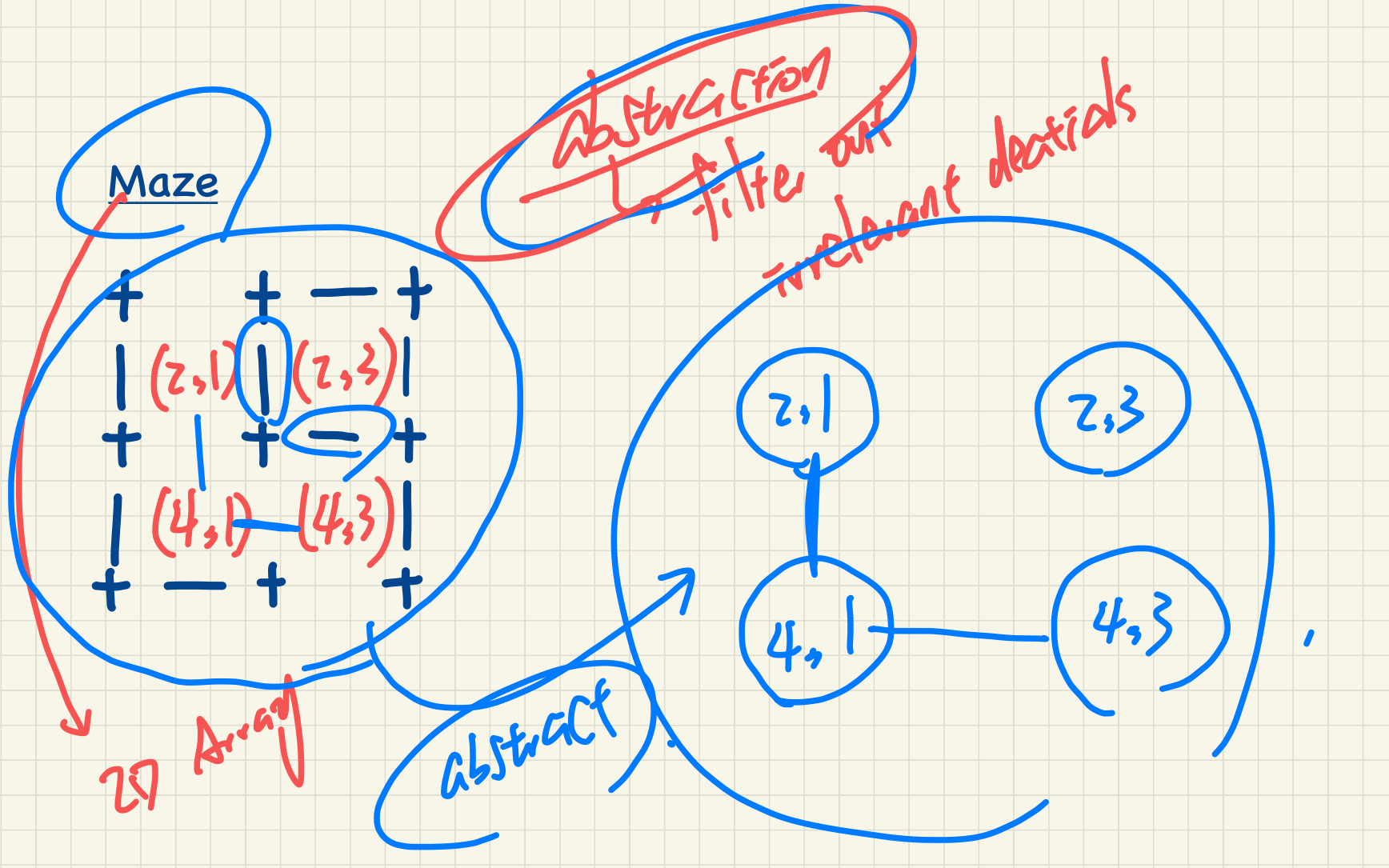
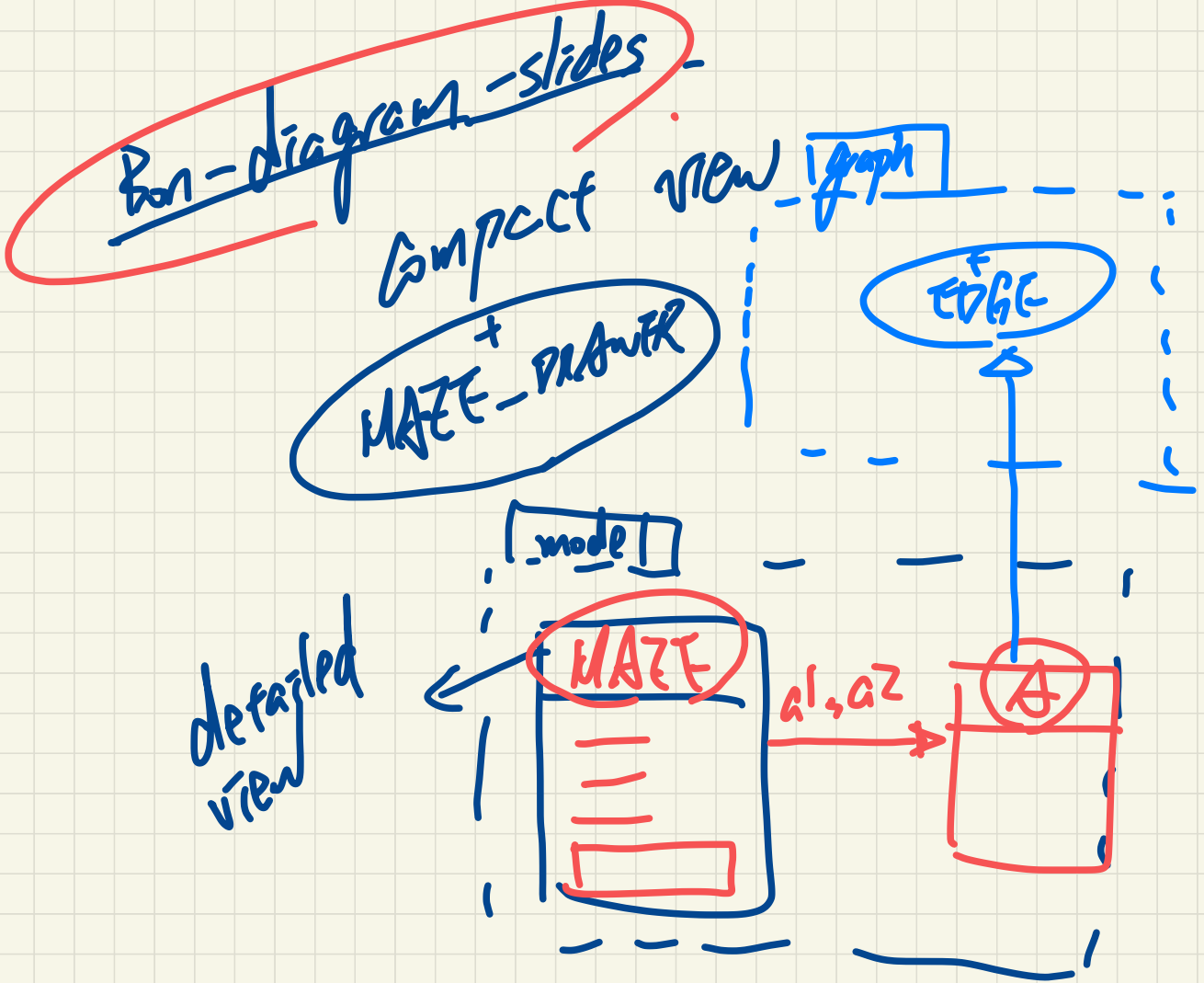
Lecture 14
Wednesday February 26


- Office hours today moved to
Thursday 12:30 - 14:30
- Lab 3: Use of Enumeration Types
ETF


Maze
(2,1)
(2,3)
(4,1)
(4,3)
2D Array
abstract
abstraction
filter out
irrelevant details
2,1
2,3
4,1
4,3

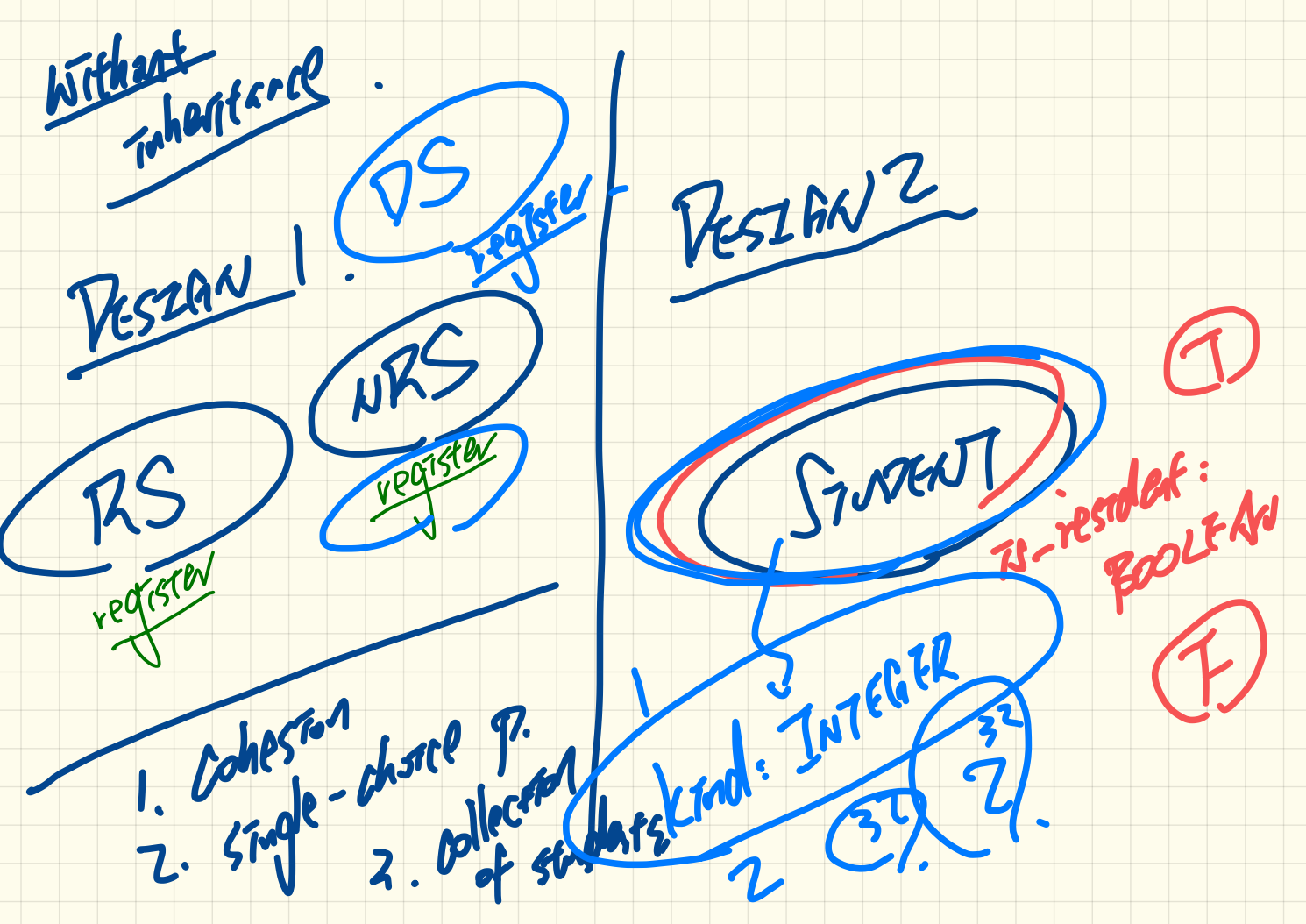

Bon-diagram-slides
Compact view
graph
MAZE_DRAWER
EDGE
node
MAZE
detailed view
a1, a2
A

Inheritance: Motivating Problem

Nouns -> classes, attributes, accessors

Verbs -> mutators

Problem: A *student management system* stores data about students. There are two kinds of university students: *resident* students and *non-resident* students. Both kinds of students have a *name* and a list of *registered courses*. Both kinds of students are restricted to *register* for no more than 10 courses. When *calculating the tuition* for a student, a base amount is first determined from the list of courses they are currently registered (each course has an associated fee). For a non-resident student, there is a *discount rate* applied to the base amount to waive the fee for on-campus accommodation. For a resident student, there is a *premium rate* applied to the base amount to account for the fee for on-campus accommodation and meals.


Without inheritance
Design 1
DS
register
NRS
register
RS
register
Design 2
STUDENT
T
F
1. Cohesion
2. Single-choice P.
2. collection of
kind: INTEGER
1
2
3
32

1st Design Attempt

```
class RESIDENT_STUDENT
create make
feature -- Attributes
  name: STRING
  courses: LINKED_LIST[COURSE]
  premium_rate:  REAL
feature -- Constructor
  make (n: STRING)
    do name := n ; create courses.make end
feature -- Commands
  set_pr (r:  REAL) do premium_rate := r end
  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 * premium_rate
    end
end
```

```
class NON_RESIDENT_STUDENT
create make
feature -- Attributes
  name: STRING
  courses: LINKED_LIST[COURSE]
  discount_rate:  REAL
feature -- Constructor
  make (n: STRING)
    do name := n ; create courses.make end
feature -- Commands
  set_dr (r:  REAL) do discount_rate := r end
  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 * discount_rate
    end
end
```

1st Design Attempt

Good design?

Judge by Cohesion

```
class RESIDENT_STUDENT
create make
feature -- Attributes
  name: STRING
  courses: LINKED_LIST[COURSE]
  premium_rate: REAL
feature -- Constructor
  make (n: STRING)
    do name := n ; create courses.make end
feature -- Commands
  set_pr (r: REAL) do premium_rate := r end
  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 * premium_rate
    end
end
```

specific to RS

```
class NON_RESIDENT_STUDENT
create make
feature -- Attributes
  name: STRING
  courses: LINKED_LIST[COURSE]
  discount_rate: REAL
feature -- Constructor
  make (n: STRING)
    do name := n ; create courses.make end
feature -- Commands
  set_dr (r: REAL) do discount_rate := r end
  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 * discount_rate
    end
end
```

specific to NRS

1st Design Attempt

Good design?

Judge by Single Choice Principle

- A new kind is introduced?
- Change on registration policy?

any duplicates?

```
class NON_RESIDENT_STUDENT
create make
feature -- Attributes
  name: STRING
  courses: LINKED_LIST[COURSE]
  discount_rate: REAL
feature -- Constructor
  make (n: STRING)
    do name := n ; create courses.make end
feature -- Commands
  set_dr (r: REAL) do discount_rate := r end
  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 * discount_rate
    end
end
```

```
class RESIDENT_STUDENT
create make
feature -- Attributes
  name: STRING
  courses: LINKED_LIST[COURSE]
  premium_rate: REAL
feature -- Constructor
  make (n: STRING)
    do name := n ; create courses.make end
feature -- Commands
  set_pr (r: REAL) do premium_rate := r end
  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 * premium_rate
    end
end
```

if cs.count < 7 then cs.extend(c) else ... end


unrelated.
class RESIDENT_STUDENT
create make
feature -- Attributes
premium_rate: REAL
feature -- Constructor
feature -- Commands
set_pr (r: REAL) do premium_rate := r end
feature -- Queries
tuition: REAL
local base: REAL
Result := base * premium_rate
end
end
register
class NON_RESIDENT_STUDENT
create make
feature -- Attributes
discount_rate: REAL
feature -- Constructor
feature -- Commands
set_dr (r: REAL) do discount_rate := r end
feature -- Queries
tuition: REAL
local base: REAL
Result := base * discount_rate
end
end
register

1st Design Attempt

Good design?

How do you build a **STUDENT_MANGEMENT_SYSTEM** class accordingly?

```
class RESIDENT_STUDENT
create make
feature -- Attributes
  name: STRING
  courses: LINKED_LIST[COURSE]
  premium_rate: REAL
feature -- Constructor
  make (n: STRING)
    do name := n ; create courses.make end
feature -- Commands
  set_pr (r: REAL) do premium_rate := r end
  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 * premium_rate
    end
end
```

```
class NON_RESIDENT_STUDENT
create make
feature -- Attributes
  name: STRING
  courses: LINKED_LIST[COURSE]
  discount_rate: REAL
feature -- Constructor
  make (n: STRING)
    do name := n ; create courses.make end
feature -- Commands
  set_dr (r: REAL) do discount_rate := r end
  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 * discount_rate
    end
end
```

RS NRS

class SMS Object

students: LL[AWC] RS NRS

static type

create {RS} rs. make (..)

create {NRS} nrs. make (..)

sms. add (rs)

sms. add (nrs)

sms. add (mazel)

ST. AWC

compile sms. x

① sms. students [1] register

② sms. students [1]. att_pr (2.3)

Rd

Without Inheritance (Design 1) Collection of Students

```
class STUDENT_MANAGEMENT_SYSETM
  rs : LINKED_LIST[RESIDENT_STUDENT]
  nrs : LINKED_LIST[NON_RESIDENT_STUDENT]
  add_rs (rs: RESIDENT_STUDENT) do ... end
  add_nrs (nrs: NON_RESIDENT_STUDENT) do ... end
  register_all (Course c) -- Register a common course 'c'
    do
      across rs as c loop c.item.register (c) end
      across nrs as c loop c.item.register (c) end
    end
end
```

duplicated

Clinet's Code

```
c: COURSE
rs: RESIDENT_STUDENT
nrs: NON_RESIDENT_STUDENT
sms: SMS
create c.make("3311")
create sms.make
```

```
sms.add_rs(rs)
sms.add_nrs(nrs)
sms.register_all(c)
```

Q: What if more kinds of students are to be introduced?

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
```

create {STUDENT} ks.make(1)

ms.make(2)

```
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
```

2nd Design Attempt

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

not belonging to the same kind of student

Good design?

Judge by 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
```

2nd Design Attempt

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

```
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
```

Good design?

Judge by Single Choice Principle

- A new kind is introduced?
- An existing kind is obeselete?

2nd Design Attempt

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

```
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
```

max: ARRAY[INT]

tuition: ARRAY[REAL]

INT

Good design?

How do you build a STUDENT_MANGEMENT_SYSTEM class accordingly?

Without Inheritance (Design 2) Collection of Students

```
class
  STUDENT_MANAGEMENT_SYSTEM
feature -- attributes
  students: LINKED_LIST[STUDENT]
feature -- command
  add_student(s: STUDENT)
      do
        students.extend(s)
      end
  register_all (c: COURSE)
      do
       across students is s
          loop
            s.register(c)
          end
      end
end
```

violates SCP.

Clinet's Code

```
c: COURSE
rs: STUDENT
nrs: STUDENT
sms: SMS
create c.make("3311")
create sms.make
create rs.make(1)
create nrs.make(2)

sms.add_student(rs)
sms.add_student(nrs)
sms.register_all(c)
```

Q: What if more kinds of students are to be introduced?