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
```

```
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 Single Choice Principle

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

```
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?

How do you build a

STUDENT_MANGEMENT_SYSTEM

class accordingly?

```
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
```

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
```

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

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

Good design?

Judge by Single Choice Principle

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

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

Good design?

How do you build a

STUDENT_MANGEMENT_SYSTEM

class accordingly?

```
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
```

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
```

Clinet's Code

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




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

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

Desigh 3:

Inheritance

Code Reuse

Cohesion?

Single Choice Principle?

Collection of Students?

```
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
```

```
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
```

Desigh 3:

Inheritance

Code Reuse

Cohesion?

Single Choice Principle?

Collection of Students?

```
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
```

```
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
```

Desigh 3:

Inheritance

Code Reuse

Cohesion?

Single Choice Principle?

Collection of Students?

```
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
```

```
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
```

With Inheritance (Design 3) 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
```

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




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

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