Lecture 9

Thursday October 3

local data1, data2 : DATA
da1, da2 : DATA_ACCESS
do
create da1.make
create da2.make

data1 := da1.data
data2 := da2.data
Result := data1 = data2

DATA_ACCESS

data : DATA
once ... end
— invariant —
data = data

data f

DATA

er
er
make → constructor

create instance

1st call

subsequent call ⇒ cached ref.

R1. sharing create {DATA}.make

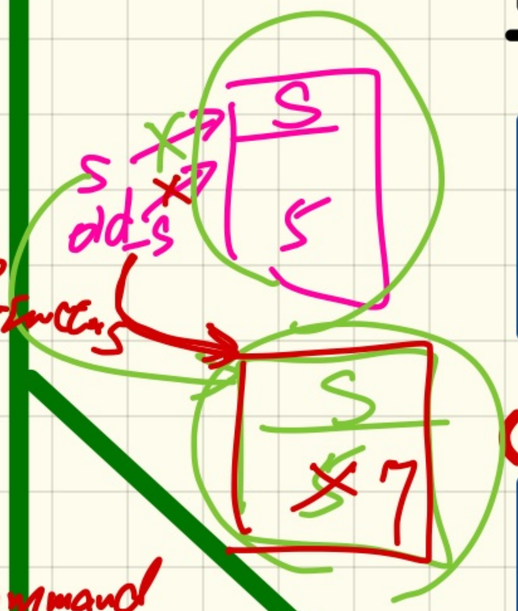
R2. cohesion create {DATA}.make

Export Status Case 1

```
class CLIENT_1
  ...
  test: BOOLEAN
    local
      s, old_s: SUPPLIER
    do
      create s.make (5)
      old_s := s
      create s.make (5)
      print (old_s = s)
      old_s := s
      s.make (7)
      print (old_s = s)
    end
end
```

calls to constructors

call to command




```
class CLIENT_2
  ...
  test: BOOLEAN
    local
      s, old_s: SUPPLIER
    do
      create s.make (5)
      old_s := s
      create s.make (5)
      print (old_s = s)
      old_s := s
      s.make (7)
      print (old_s = s)
    end
end
```

```
class SUPPLIER

create {ANY}
    make

feature {ANY} - Command
    make (init_i: INTEGER)
        do
            i := init_i
        end

feature
    i: INTEGER
end
```

Export Status Case 2

```
class CLIENT_1
  ...
  test: BOOLEAN
    local
      s, old_s: SUPPLIER
    do
      create s.make (5)
      old_s := s
      create s.make (5)
      print (old_s = s)
      old_s := s
      s.make (7)
      print (old_s = s)
    end
end
```

```
class CLIENT_2
  ...
  test: BOOLEAN
    local
      s, old_s: SUPPLIER
    do
      create s.make (5)
      old_s := s
      create s.make (5)
      print (old_s = s)
      old_s := s
      s.make (7)
      print (old_s = s)
    end
end
```

```
class SUPPLIER

create {CLIENT_1}
    make

feature
    make (init_i: INTEGER)
      do
        i := init_i
      end

feature
    i: INTEGER
end
```

only C_1 can call make as a creation proc.

Export Status Case 3

```
class CLIENT_1
  ...
  test: BOOLEAN
    local
      s, old_s: SUPPLIER
    do
      create s.make (5)
      old_s := s
      create s.make (5)
      print (old_s = s)
      old_s := s
      s.make (7)
      print (old_s = s)
    end
end
```

```
class CLIENT_2
  ...
  test: BOOLEAN
    local
      s, old_s: SUPPLIER
    do
      create s.make (5)
      old_s := s
      create s.make (5)
      print (old_s = s)
      old_s := s
      s.make (7)
      print (old_s = s)
    end
end
```

```
class SUPPLIER

create
    make

feature {CLIENT_1}
    make (init_i: INTEGER)
      do
        i := init_i
      end

feature
    i: INTEGER
end
```

Export Status Case 4

```
class CLIENT_1
  ...
  test: BOOLEAN
    local
      s, old_s: SUPPLIER
    do
      create s.make (5)
      old_s := s
      create s.make (5)
      print (old_s = s)
      old_s := s
      s.make (7)
      print (old_s = s)
    end
end
```

```
class CLIENT_2
  ...
  test: BOOLEAN
    local
      s, old_s: SUPPLIER
    do
      create s.make (5)
      old_s := s
      create s.make (5)
      print (old_s = s)
      old_s := s
      s.make (7)
      print (old_s = s)
    end
end
```

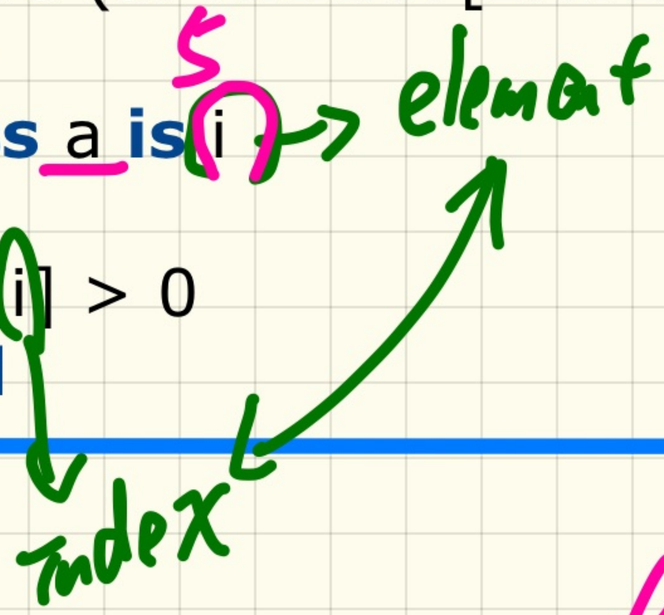
```
class SUPPLIER

create {CLIENT_1}
    make

feature {CLIENT_1}
    make (init_i: INTEGER)
      do
        i := init_i
      end
feature → set_i(...)
feature
    i: INTEGER
end
```

Writing Postcondition: Exercise 4

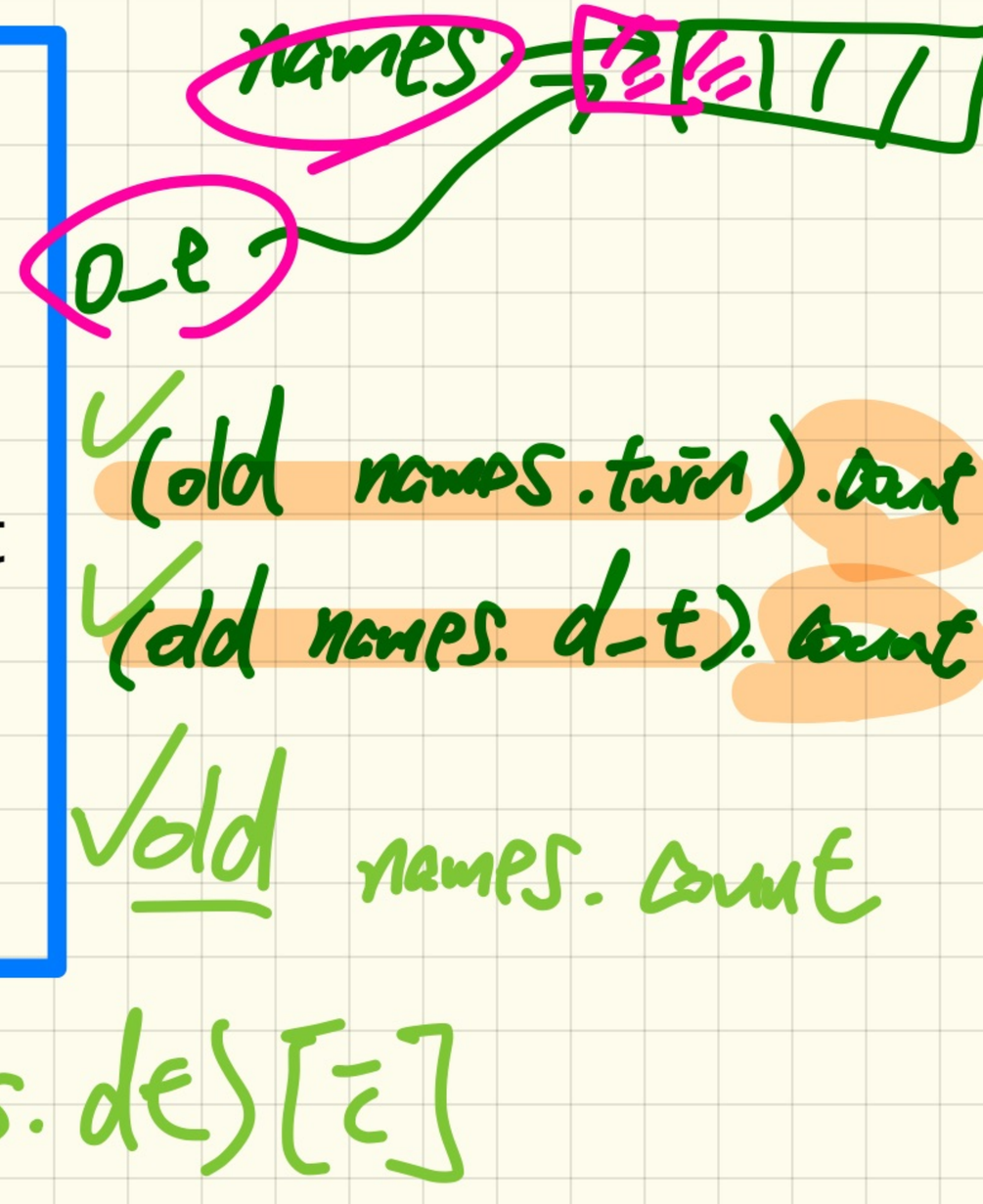
```
is_all_positive (a: ARRAY[INTEGER]): BOOLEAN
    ensure
        across a is i
            all
                a[i] > 0
            end
```



a[5] X

Writing Postcondition: Exercise 5

```
names: ARRAY[STRING]
name_at (i: INTEGER)
  require
    names.valid_index(i)
  ensure
    names.count = (old names).count
    across 1 |..| names.count is j
    all
      names[j] ~ (old names[j])
    end
```



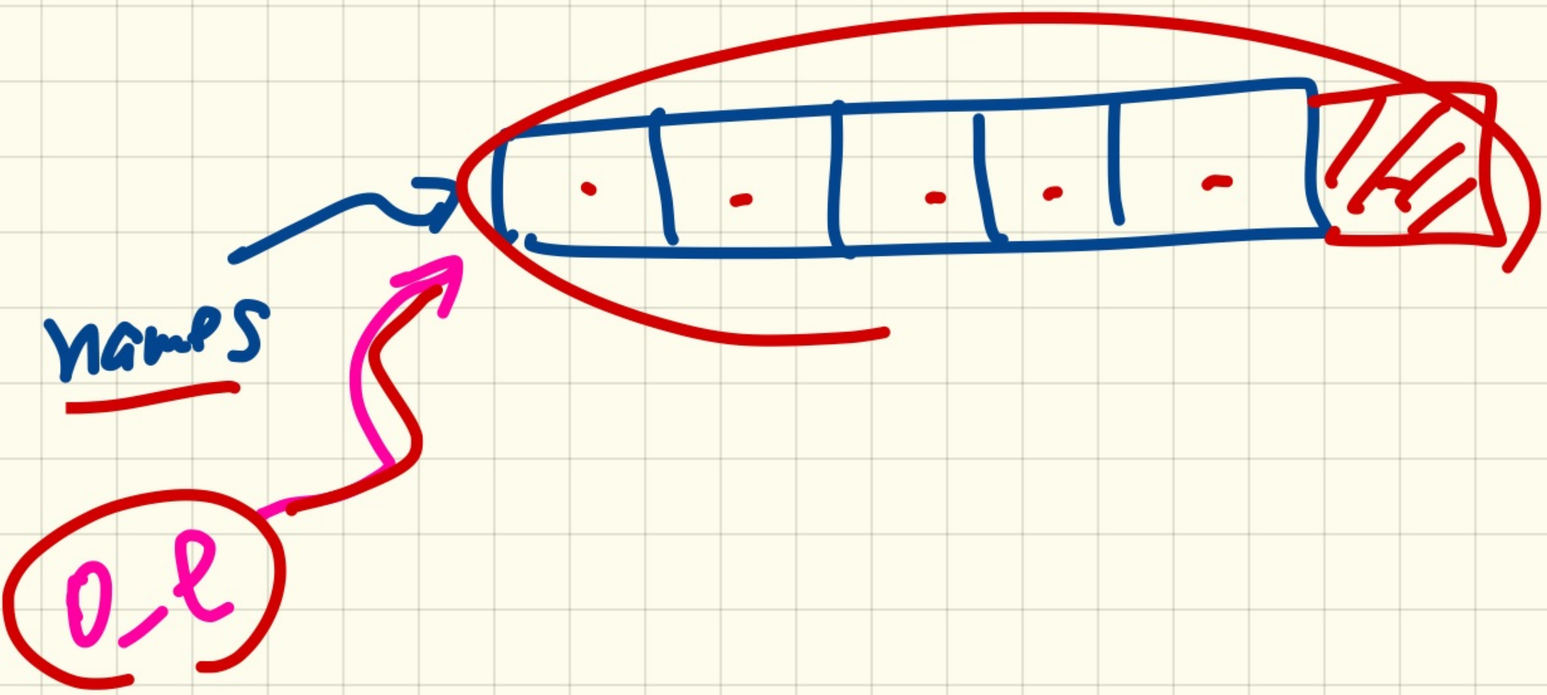
(**old** names.**twin**).count or **old** names.count

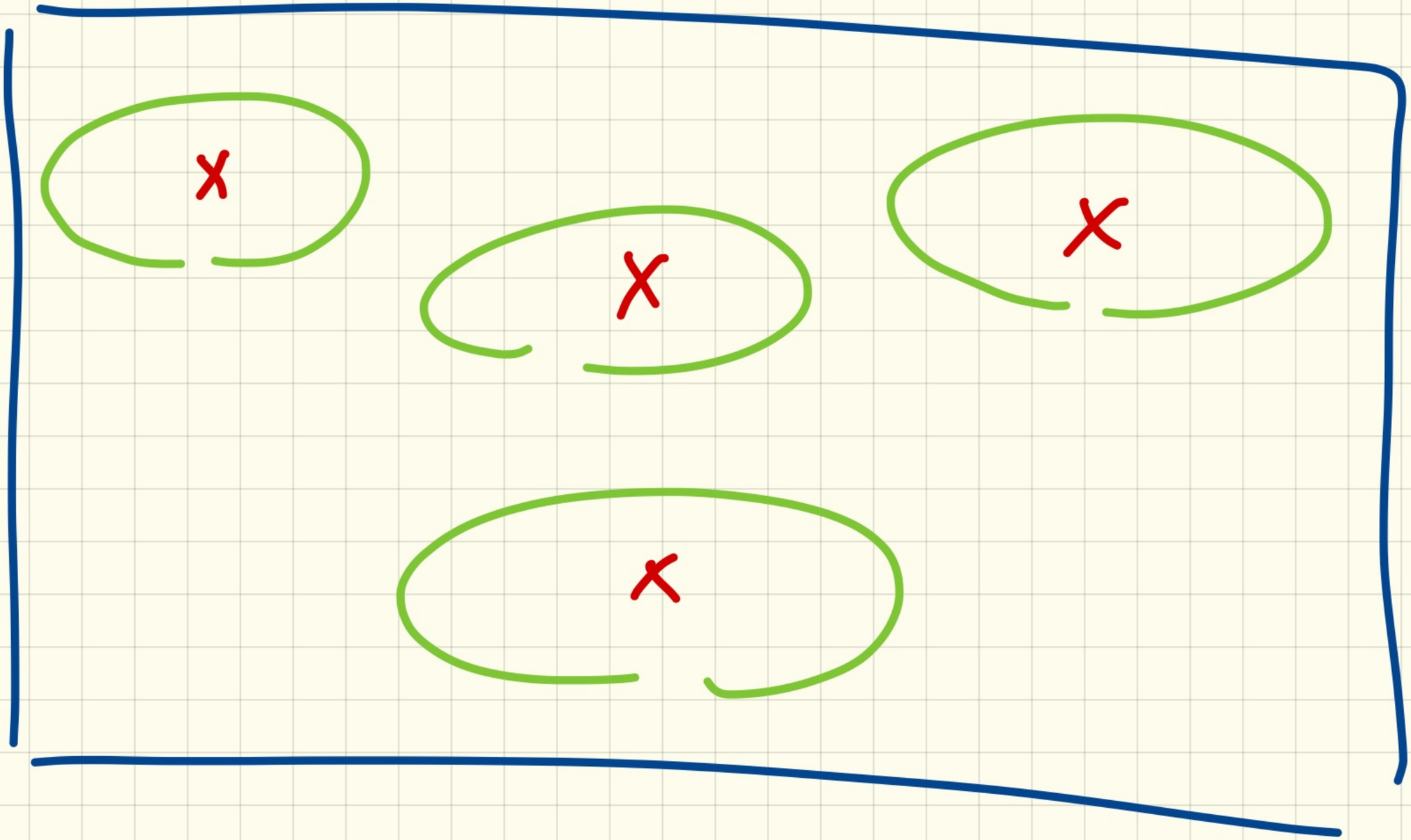
(**old** names.**deep_twin**)[i]

→ o_e := names

ensure

names.count = (old names).count



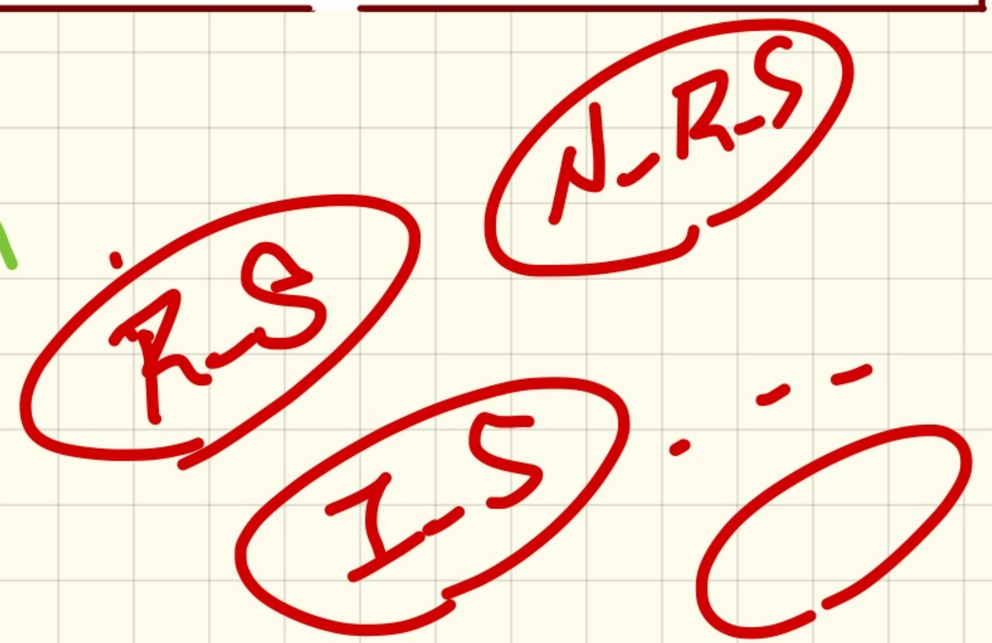


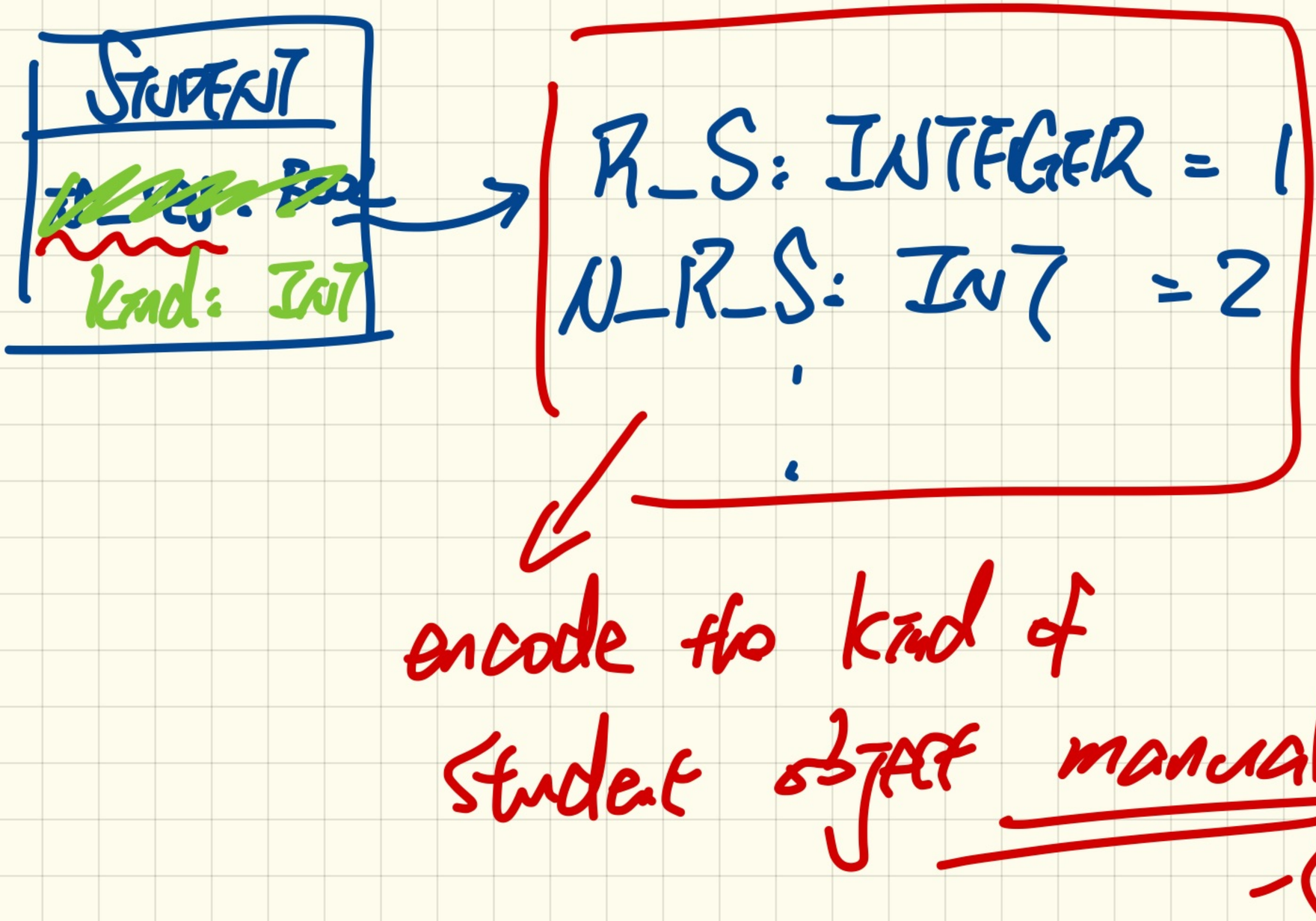
Violation of the Single Choice Principle

```
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 courses.count < 5 then




STUDENT
kind: INT
R_S: INTEGER = 1
N_R_S: INT = 2
encode the kind of
student object manually

STUDENT

kind: INT

invariant

1 ≤ kind ≤ 10

SCP

charge (s: STUDENT)

do

if s.kind = 1 then

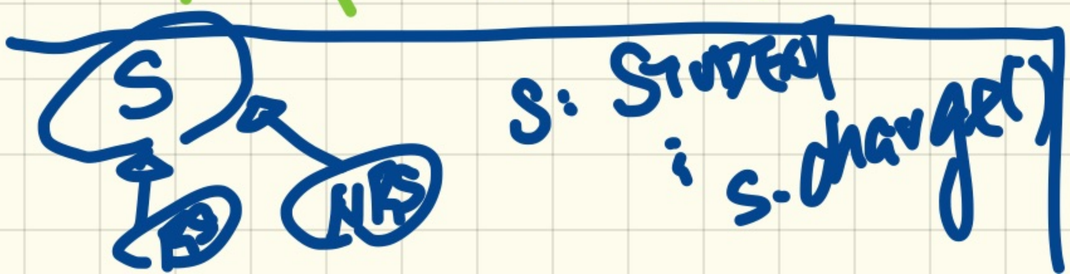
elseif s.kind = 2 then

elseif s.kind = 3 then

end

end

Q. What if adding a 11th kind student.



get_tuition (s: STUDENT): INT

do

if s.kind = 1 then

elseif s.kind = 2 then

elseif s.kind = 3 then

end

end

Without inheritance -

STUDENT

kind: INT

change(..)
do
if kind = 1 then
elseif kind = 2 then
end

tuition(..)
if kind = 1 then
elseif kind = 2 then
end

RS

URS

RS
change
do
end
tuition
do

URS
change
do
end
tuition
do

end

STUDENT

change *

tuition *

S: STUDENT end
create {RS} S.make
S.change