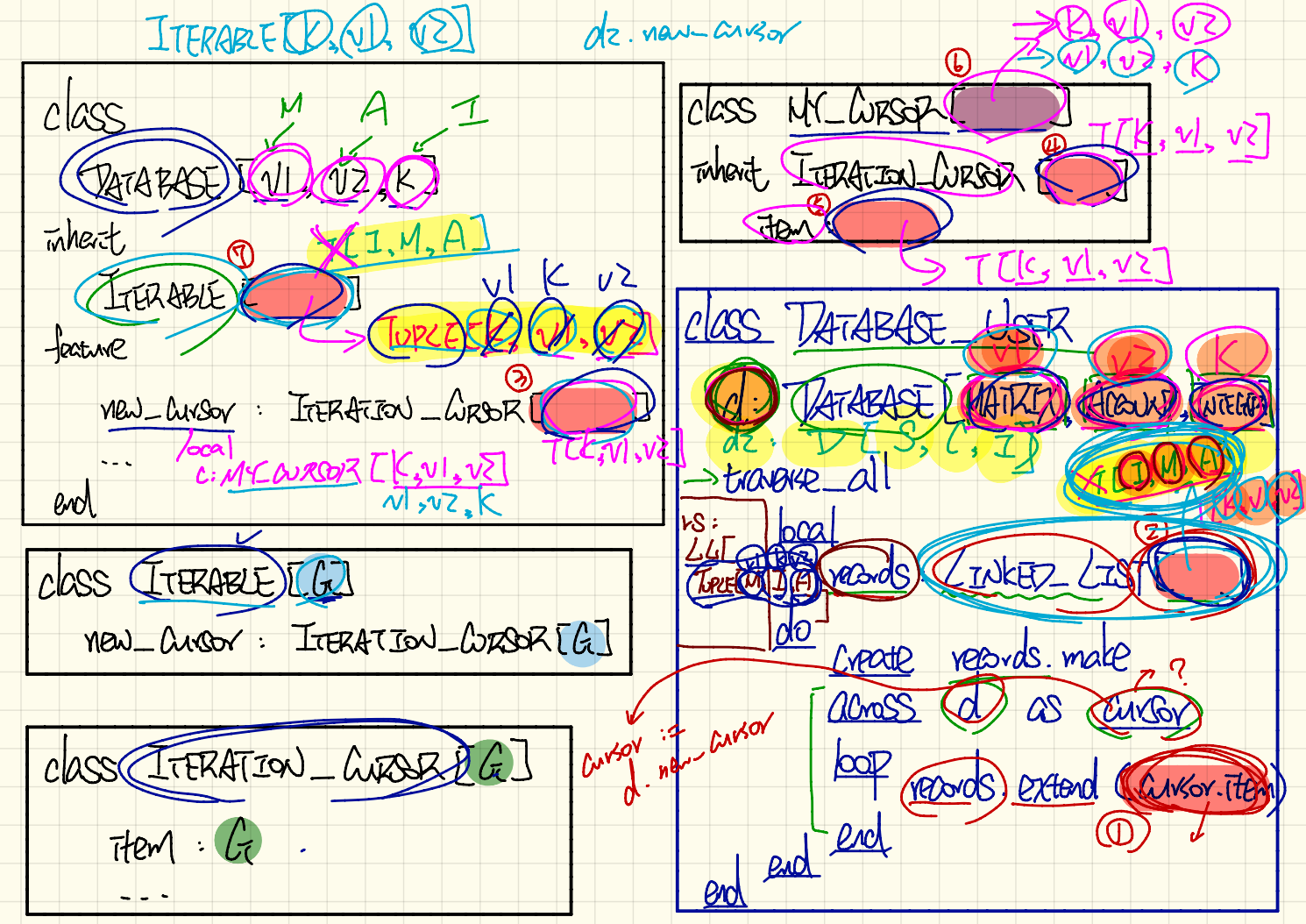
Tuesday Oct. 2

Lecture 8


ITERABLE [K, V1, V2]
dz.new_cursor
class
DATABASE [V1, V2, K]
M A I
inherit
ITERABLE [I]
X [I, M, A]
feature
new_cursor : ITERATION_CURSOR [TUPLE [K, V1, V2]]
local
c: MY_CURSOR [K, V1, V2]
V1, V2, K
end
class ITERABLE [G]
new_cursor : ITERATION_CURSOR [G]
class ITERATION_CURSOR [G]
item : G
class MY_CURSOR [ ]
inherit ITERATION_CURSOR [ ]
item
T [K, V1, V2]
T [G, V1, V2]
class DATABASE_USER
dz : DATABASE [MATRIX, CHARACTER, INTEGER]
traverse_all
vs : LL [TUPLE [V1, V2, K]]
local
records
LINKED_LIST
do
create records.make
across dz as cursor
loop
records.extend (cursor.item)
end
end
cursor := dz.new_cursor
X [I, M, A]
V1, V2, K

# Singleton Pattern: Code (1)

data ~ data

## Supplier:

```
class DATA
create {DATA_ACCESS} make
feature {DATA_ACCESS}
  make do v := 10 end
feature -- Data Attributes
  v: INTEGER
  change_v (nv: INTEGER)
    do v := nv end
end
```

```
expanded class
  DATA_ACCESS
feature
  data: DATA
      -- The one and only access
  once create Result.make end
invariant data = data
end
```

## Client:

```
test: BOOLEAN
  local
    access: DATA_ACCESS
    d1, d2: DATA
  do
    d1 := access.data
    d2 := access.data
    Result := d1 = d2
      and d1.v = 10 and d2.v = 10
    check Result end
    d1.change_v (15)
    Result := d1 = d2
      and d1.v = 15 and d2.v = 15
  end
end
```

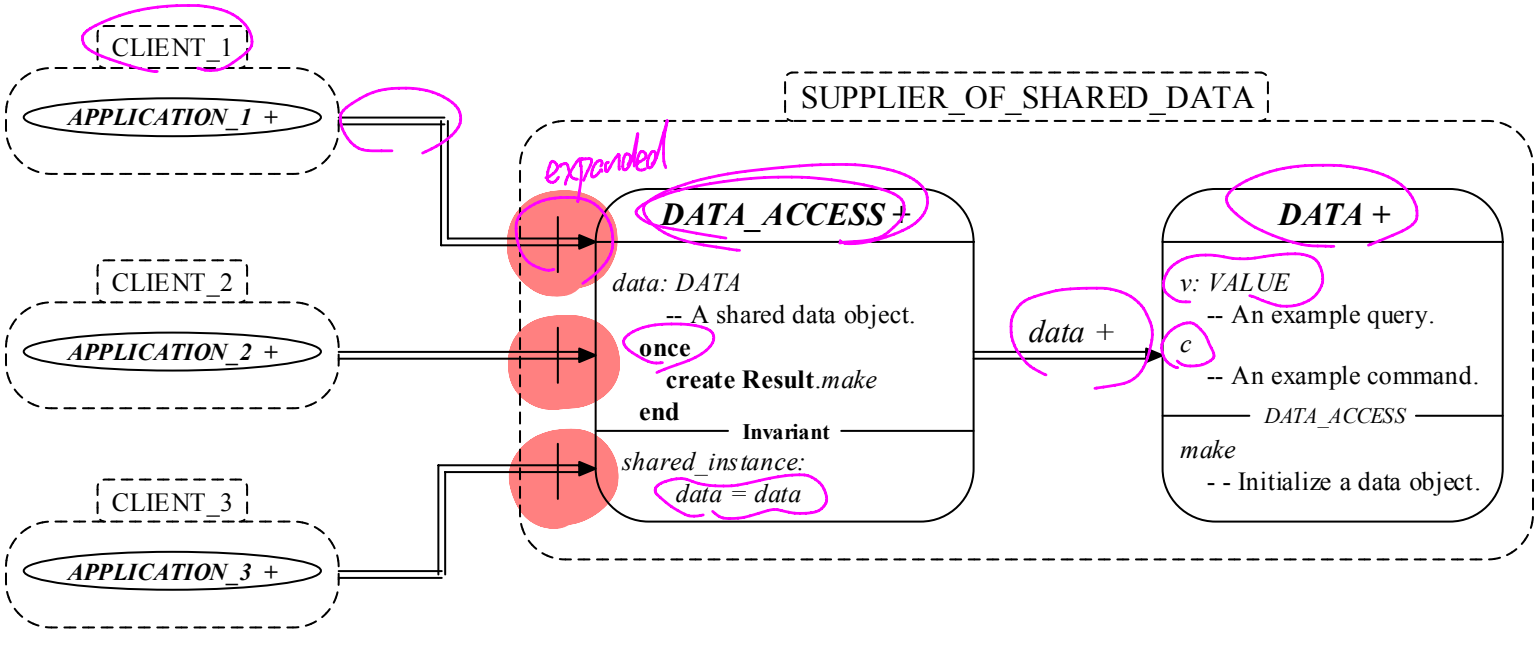
data : DATA
do
create data.make
end

create d1.make ✗ ①

d1.make ✗ ②

data
a call to q. 'data' v3 v2 Data

a separate call to q. 'data'


CLIENT_1
APPLICATION_1 +
CLIENT_2
APPLICATION_2 +
CLIENT_3
APPLICATION_3 +
SUPPLIER_OF_SHARED_DATA
expanded
DATA_ACCESS +
data: DATA
-- A shared data object.
once
create Result.make
end
Invariant
shared_instance:
data = data
data +
DATA +
v: VALUE
-- An example query.
c
-- An example command.
DATA_ACCESS
make
- - Initialize a data object.

# Singleton Pattern: Code (2)

## Supplier:

```
class BANK_DATA
create {BANK_DATA_ACCESS} make
feature {BANK_DATA_ACCESS}
  make do ... end
feature -- Data Attributes
  interest_rate: REAL
  set_interest_rate (r: REAL)
  ...
end
```

```
expanded class
  BANK_DATA_ACCESS
feature
  data: BANK_DATA
    -- The one and only access
    once create Result.make end
invariant data = data
```

## Client:

```
class
  ACCOUNT
feature
  data: BANK_DATA
  make (...)
      -- Init. access to bank data.
    local
      data_access: BANK_DATA_ACCESS
    do
      data := data_access.data
      ...
    end
end
```

create data.make ✗

# Testing of Singleton Pattern

```
test_bank_shared_data: BOOLEAN
    -- Test that a single data object is manipulated
  local acc1, acc2: ACCOUNT
  do
    comment("t1: test that a single data object is shared")
    create acc1.make ("Bill")
    create acc2.make ("Steve")
    Result := acc1.data = acc2.data
    check Result end
    Result := acc1.data ~ acc2.data
    check Result end
    acc1.data.set_interest_rate (3.11)
    Result :=
          acc1.data.interest_rate = acc2.data.interest_rate
      and acc1.data.interest_rate = 3.11
    check Result end
    acc2.data.set_interest_rate (2.98)
    Result :=
          acc1.data.interest_rate = acc2.data.interest_rate
      and acc1.data.interest_rate = 2.98
  end
```

1st time data in DATA_Access is called

2nd time

# Violation of 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
```

STUDENT

resident: BOOLEAN

```
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
```

change or register