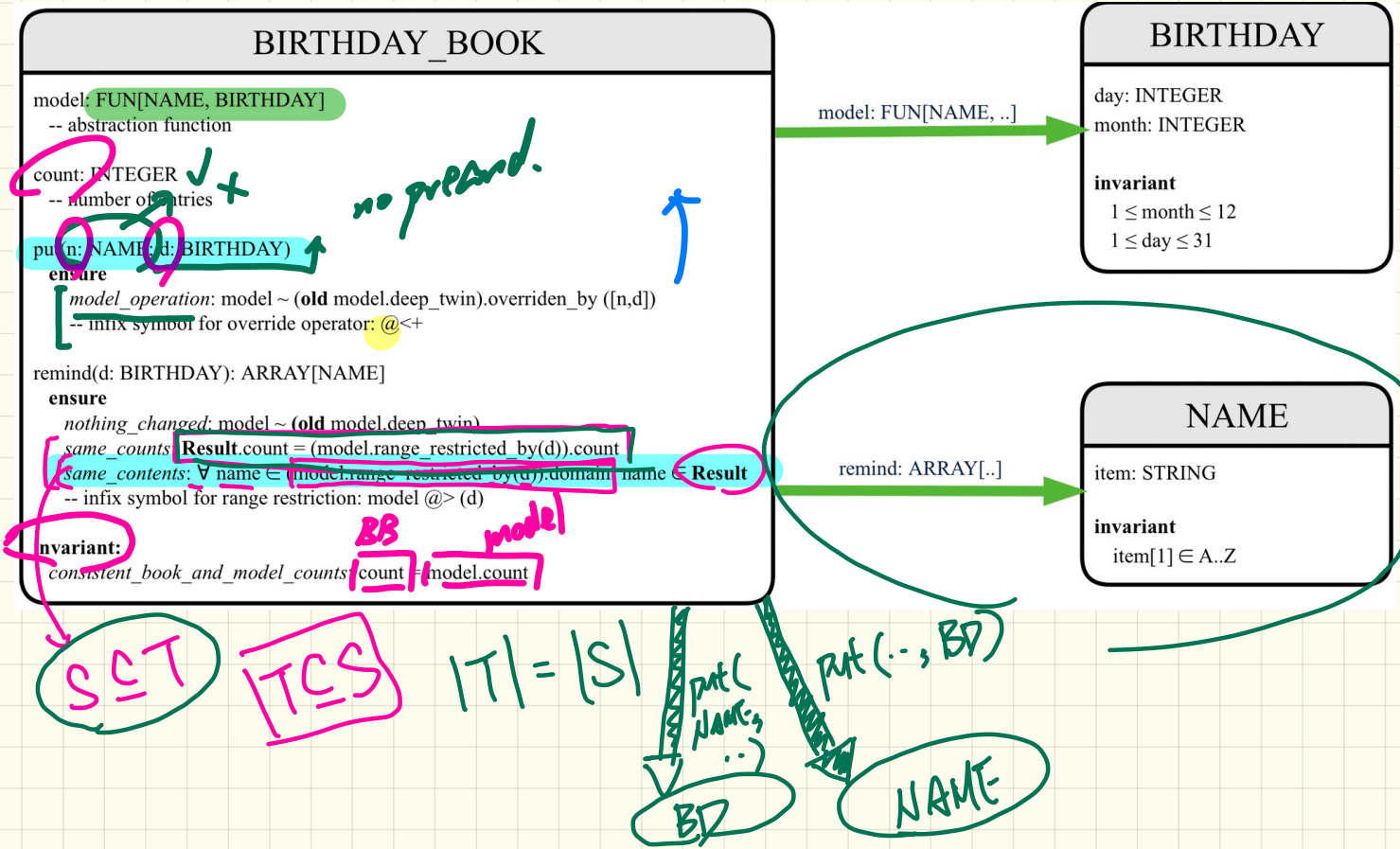
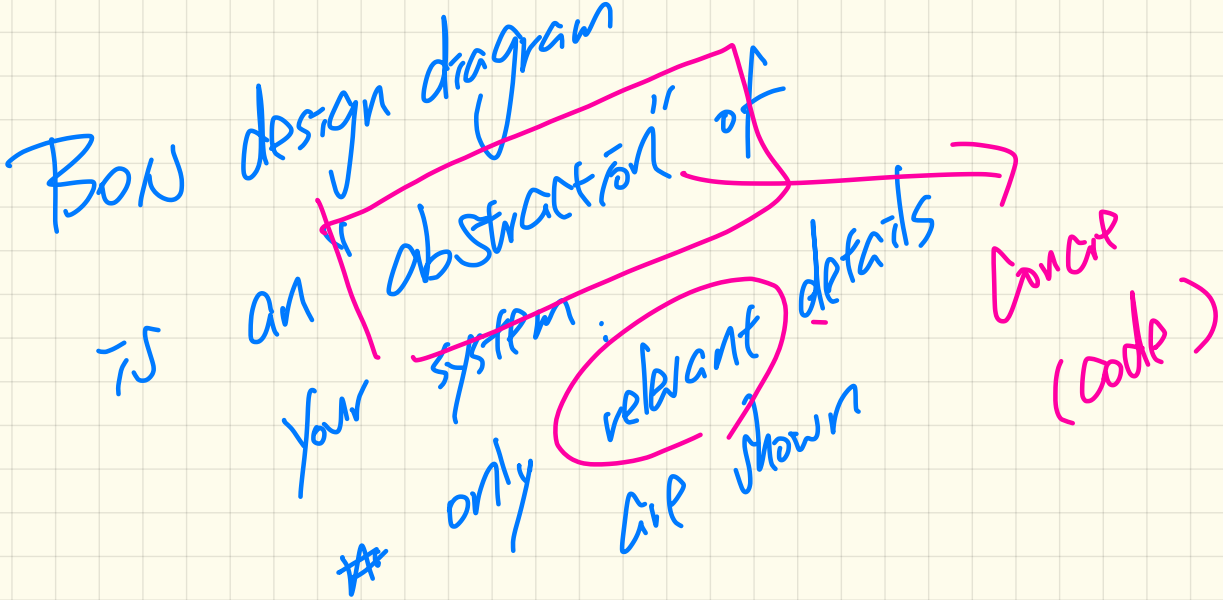
Lecture 10
Wednesday February 5

- Labtest 1:
  * Birthday Book
  * MATHMODELS
  * Iterator Patterns: Two Tutorial Series

Birthday Book: Design

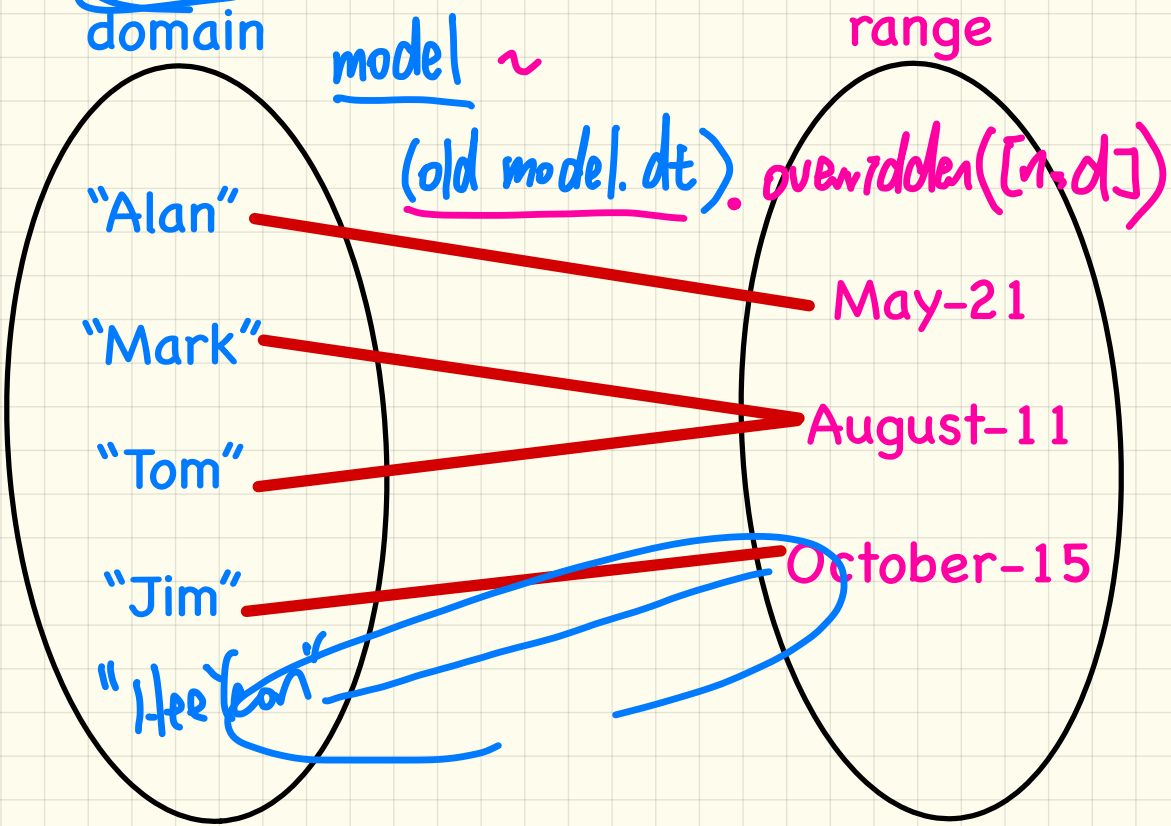



BON design diagram
is an "abstraction" of
your system:
only relevant details
are shown
Concrete
(code)

Birthday Book: Model Operation (1.1)

book.put("Heeyeon", October-15)

override



model ~ (old model1.dt) . overridden([n,d])

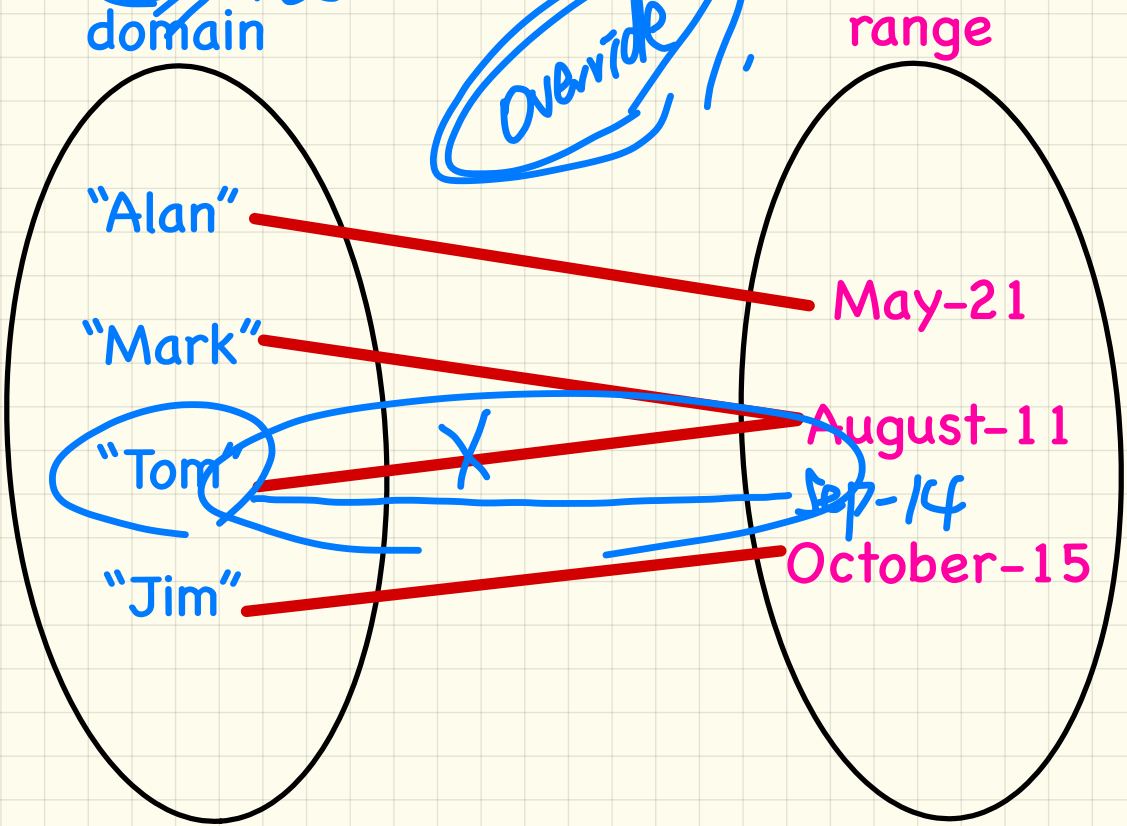
model ~ (old model1.dt) @<+ [n,d]

⊕

FUN
↳ extend cmd.
extended gv.

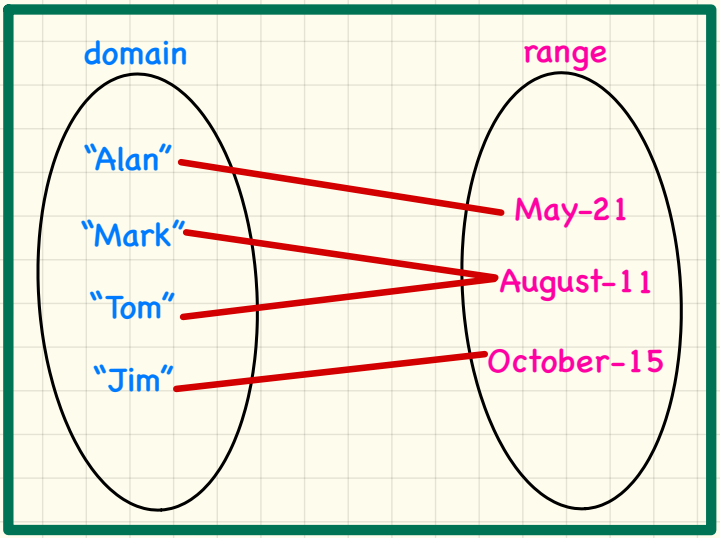
Birthday Book: Model Operation (1.2)

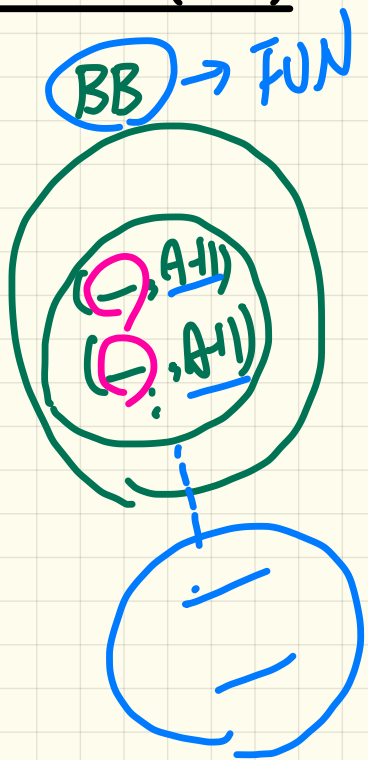
book.put("Tom", September-14)

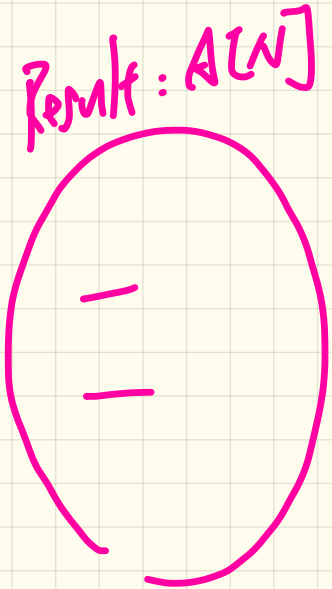


Birthday Book: Model Operation (2.1)

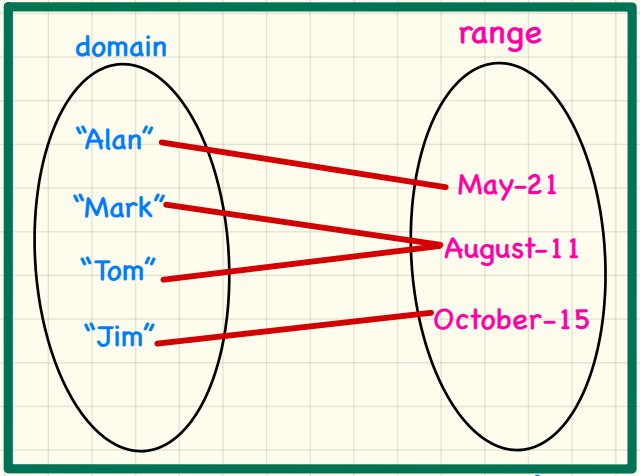
book.remind(August-11)







book.remind(August-11)

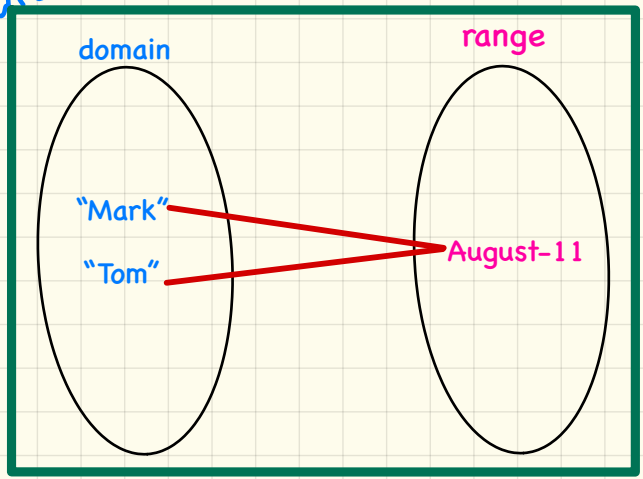


FUN[NAME, BD]
d. r.

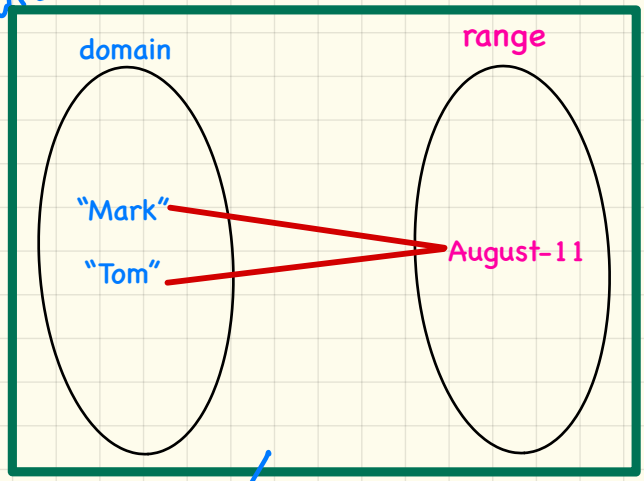
model.range_restricted(Aug-11)

domain restric.
domain sub.

range restric.
range sub.



model.range_restricted(Aug-11)



model.range_restricted(Aug-11)

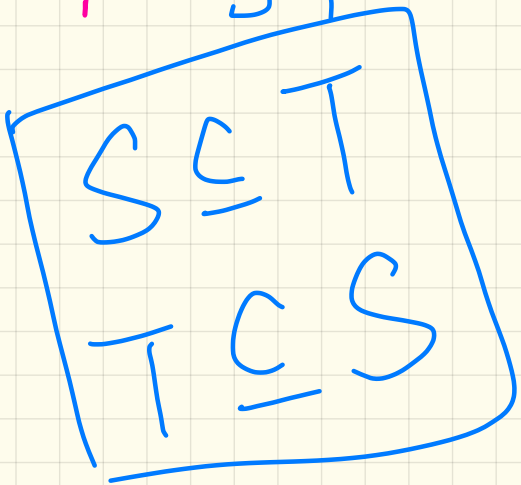
domain

remind(d) : A[N]

model. d_r(d). domain] S

||

Result] I



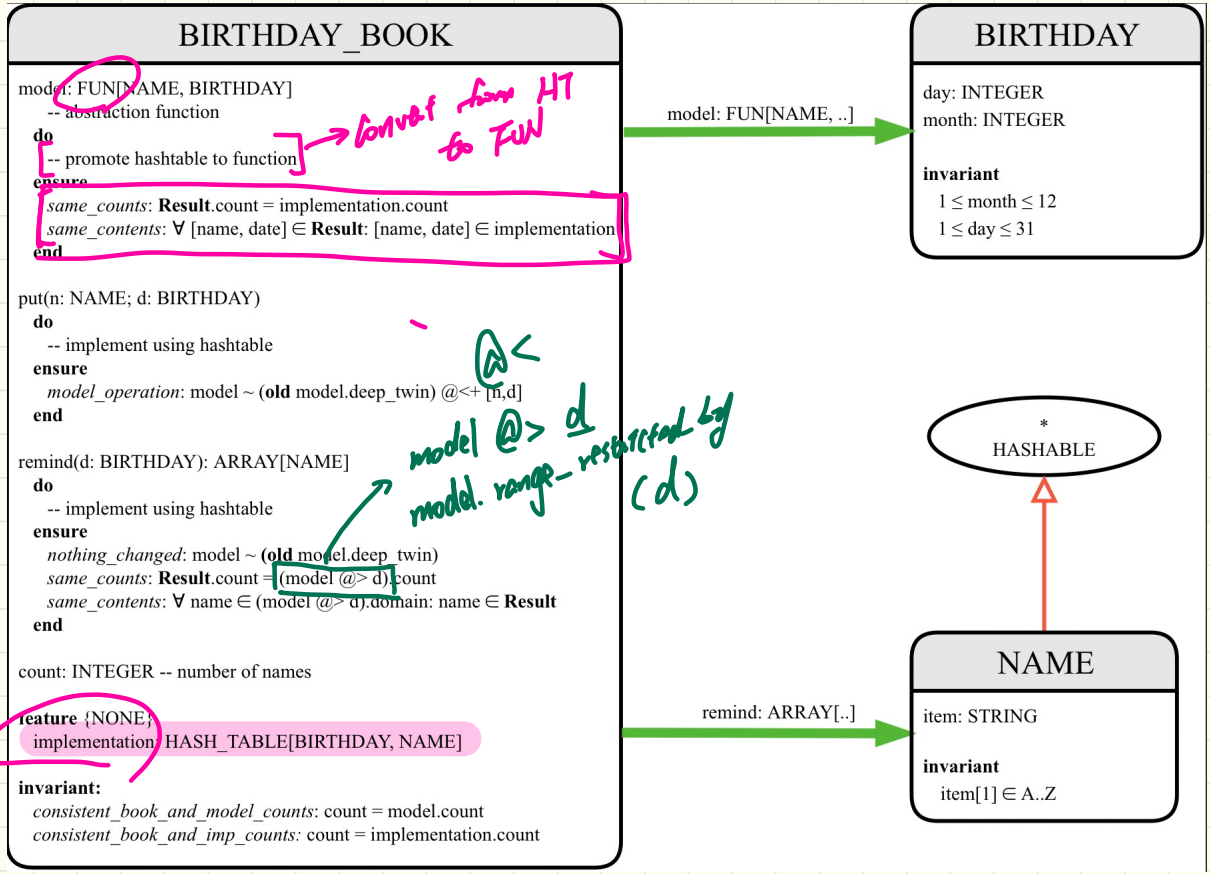
set of tuples

range_restricted ( )

range_restricted_by ( )

single tuple

Birthday Book: Implementation



Stack of Strings vs. Stack of Accounts

```
class STRING_STACK
feature {NONE} -- Implementation
  imp: ARRAY[ STRING ] ; i: INTEGER
feature -- Queries
  count: INTEGER do Result := i end
     -- Number of items on stack.
  top: STRING do Result := imp [i] end
     -- Return top of stack.
feature -- Commands
  push (v: STRING ) do imp[i] := v; i := i + 1 end
     -- Add 'v' to top of stack.
  pop do i := i - 1 end
     -- Remove top of stack.
end
```

```
class ACCOUNT_STACK
feature {NONE} -- Implementation
  imp: ARRAY[ ACCOUNT ] ; i: INTEGER
feature -- Queries
  count: INTEGER do Result := i end
     -- Number of items on stack.
  top: ACCOUNT do Result := imp [i] end
     -- Return top of stack.
feature -- Commands
  push (v: ACCOUNT ) do imp[i] := v; i := i + 1 end
     -- Add 'v' to top of stack.
  pop do i := i - 1 end
     -- Remove top of stack.
end
```

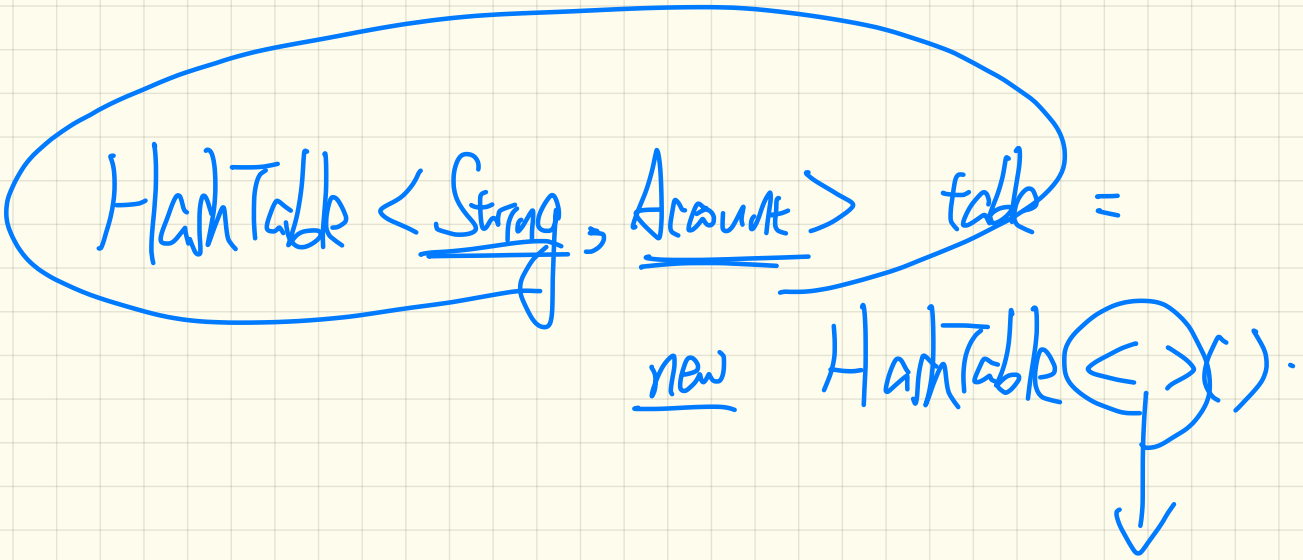
A Generic Stack

Supplier

```
class STACK [G]
feature {NONE} -- Implementation
  imp: ARRAY[G] ; i: INTEGER
feature -- Queries
  count: INTEGER do Result := i end
      -- Number of items on stack.
  top: G do Result := imp [i] end
      -- Return top of stack.
feature -- Commands
  push (v: G) do imp[i] := v; i := i + 1 end
      -- Add 'v' to top of stack.
  pop do i := i - 1 end
      -- Remove top of stack.
end
```

Client

```
1  test_stacks: BOOLEAN
2    local
3      ss: STACK[STRING] ; sa: STACK[ACCOUNT]
4      s: STRING ; a: ACCOUNT
5    do
6      ss.push("A")
7      ss.push(create {ACCOUNT}.make ("Mark", 200))
8      s := ss.top
9      a := ss.top
10     sa.push(create {ACCOUNT}.make ("Alan", 100))
11     sa.push("B")
12     a := sa.top
13     s := sa.top
14   end
```


HashTable<String, Account> table =
new HashTable<>();

Principle of Information Hiding

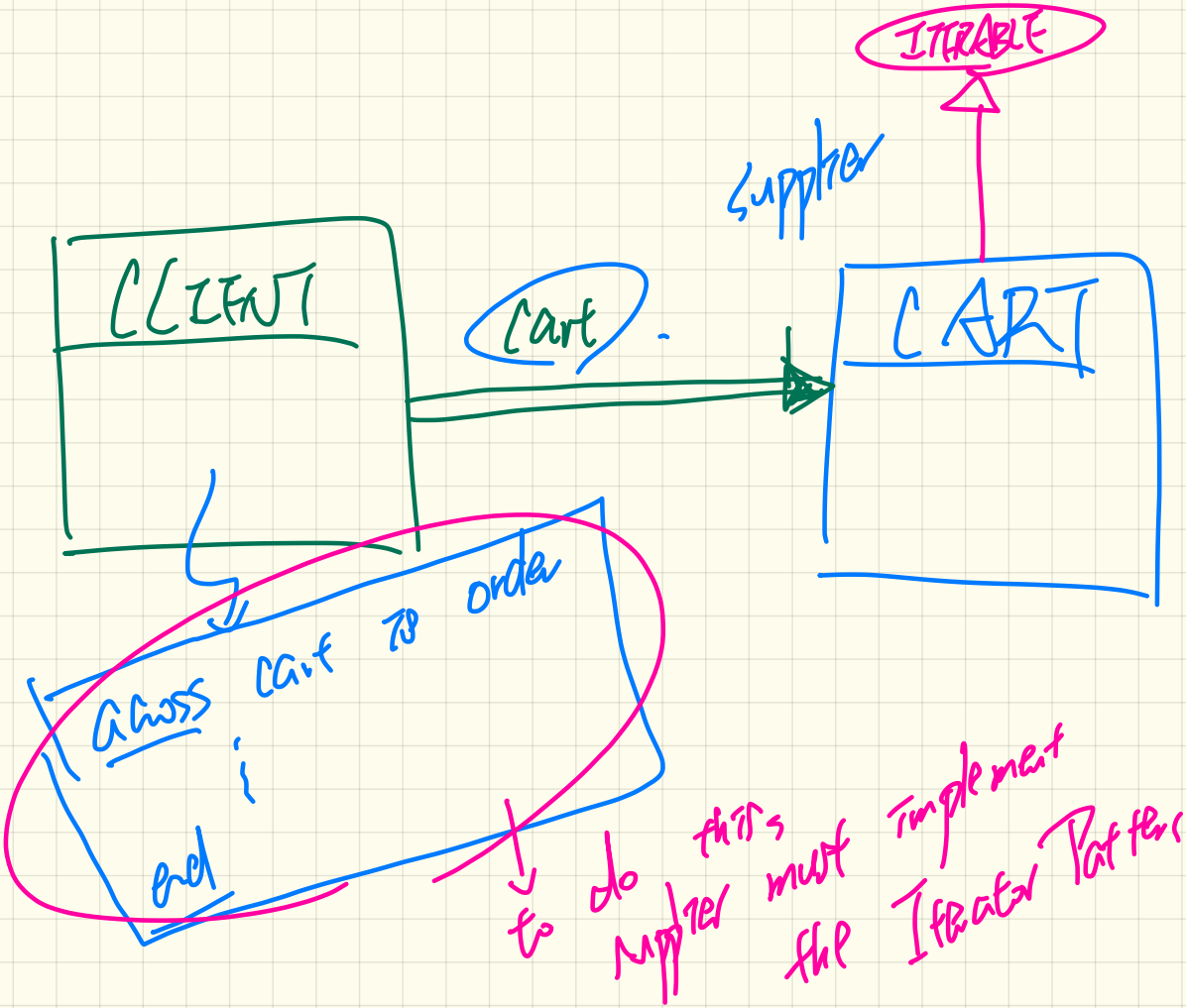
Supplier:

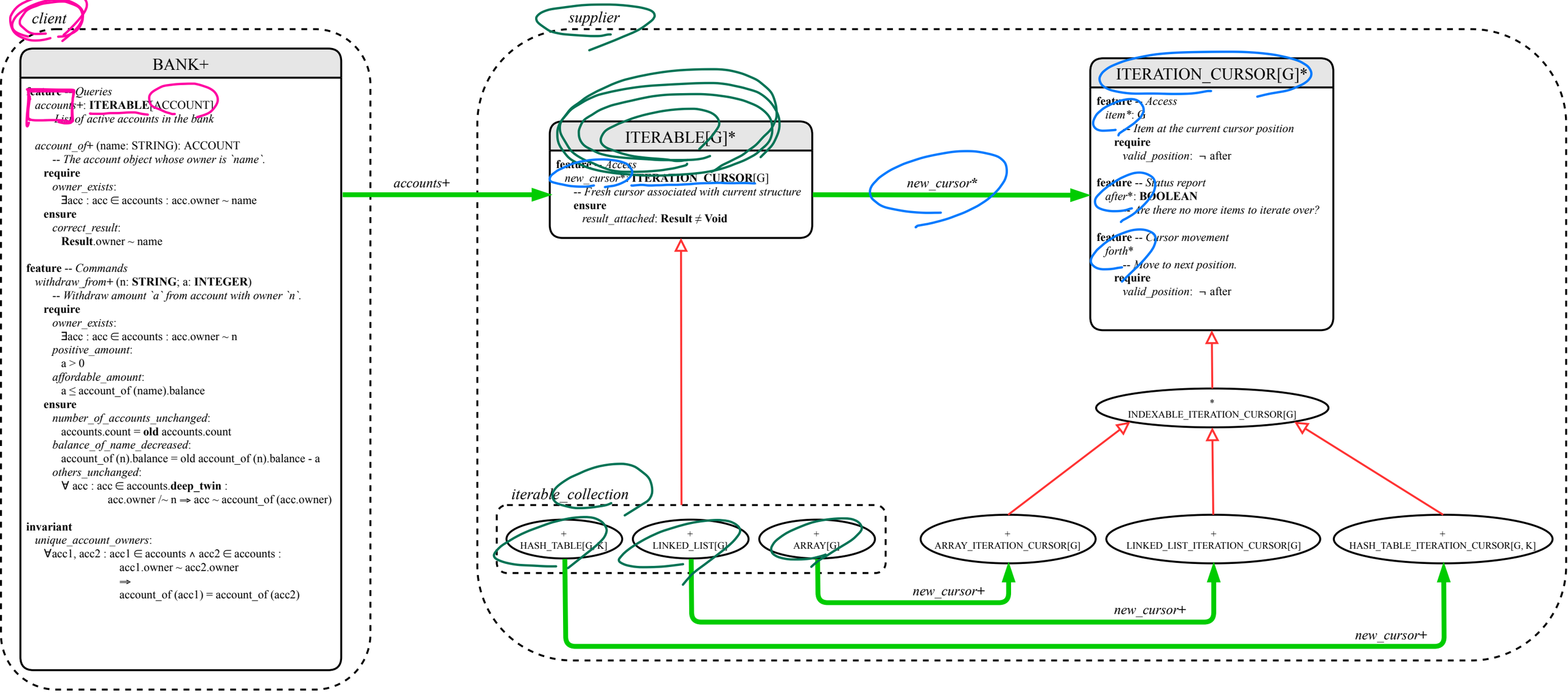
```
class
  CART
feature
  orders: ARRAY[ORDER]
end

class
  ORDER
feature
  price: INTEGER
  quantity: INTEGER
end
```

Client:

```
class
  SHOP
feature
  cart: CART
  checkout: INTEGER
    do
      from
        i := cart.orders.lower
      until
        i > cart.orders.upper
      do
        Result := Result +
          cart.orders[i].price
          *
          cart.orders[i].quantity
        i := i + 1
      end
    end
end
```

Problems?


ITERABLE
Supplier
CLIENT
Cart
CART
Guess cart to order
i
end
to do this supplier must implement the Iterator Pattern


client
supplier
BANK+
feature -- Queries
accounts+: ITERABLE[ACCOUNT]
List of active accounts in the bank
account_of+ (name: STRING): ACCOUNT
-- The account object whose owner is `name`.
require
owner_exists:
∃acc : acc ∈ accounts : acc.owner ~ name
ensure
correct_result:
Result.owner ~ name
feature -- Commands
withdraw_from+ (n: STRING; a: INTEGER)
-- Withdraw amount `a` from account with owner `n`.
require
owner_exists:
∃acc : acc ∈ accounts : acc.owner ~ n
positive_amount:
a > 0
affordable_amount:
a ≤ account_of (name).balance
ensure
number_of_accounts_unchanged:
accounts.count = old accounts.count
balance_of_name_decreased:
account_of (n).balance = old account_of (n).balance - a
others_unchanged:
∀ acc : acc ∈ accounts.deep_twin :
acc.owner /~ n ⇒ acc ~ account_of (acc.owner)
invariant
unique_account_owners:
∀acc1, acc2 : acc1 ∈ accounts ∧ acc2 ∈ accounts :
acc1.owner ~ acc2.owner
⇒
account_of (acc1) = account_of (acc2)
accounts+
ITERABLE[G]*
feature -- Access
new_cursor*: ITERATION_CURSOR[G]
-- Fresh cursor associated with current structure
ensure
result_attached: Result ≠ Void
new_cursor*
ITERATION_CURSOR[G]*
feature -- Access
item*: G
-- Item at the current cursor position
require
valid_position: ¬ after
feature -- Status report
after*: BOOLEAN
-- Are there no more items to iterate over?
feature -- Cursor movement
forth*
-- Move to next position.
require
valid_position: ¬ after
*
INDEXABLE_ITERATION_CURSOR[G]
iterable_collection
+
HASH_TABLE[G, K]
+
LINKED_LIST[G]
+
ARRAY[G]
+
ARRAY_ITERATION_CURSOR[G]
+
LINKED_LIST_ITERATION_CURSOR[G]
+
HASH_TABLE_ITERATION_CURSOR[G, K]
new_cursor+
new_cursor+
new_cursor+