LECTURE 4

TUESDAY SEPTEMBER 17

class BANK

accounts: ARRAY[ACCOUNT]

LL

like

- anchor type -

make (new_accounts: like accounts.)

add_accounts (accs : like accounts)

Single choice principle

end

Copying Collection Objects : Reference Copy & Make Changes

```
1  old_imp := imp
2  Result := old_imp = imp   -- Result = [redacted]
3  imp[2] := "Jim"
4  Result :=
5  across 1 |..| imp.count is j
6    all imp [j] ~ old_imp [j]
7    end -- Result = [redacted]
```

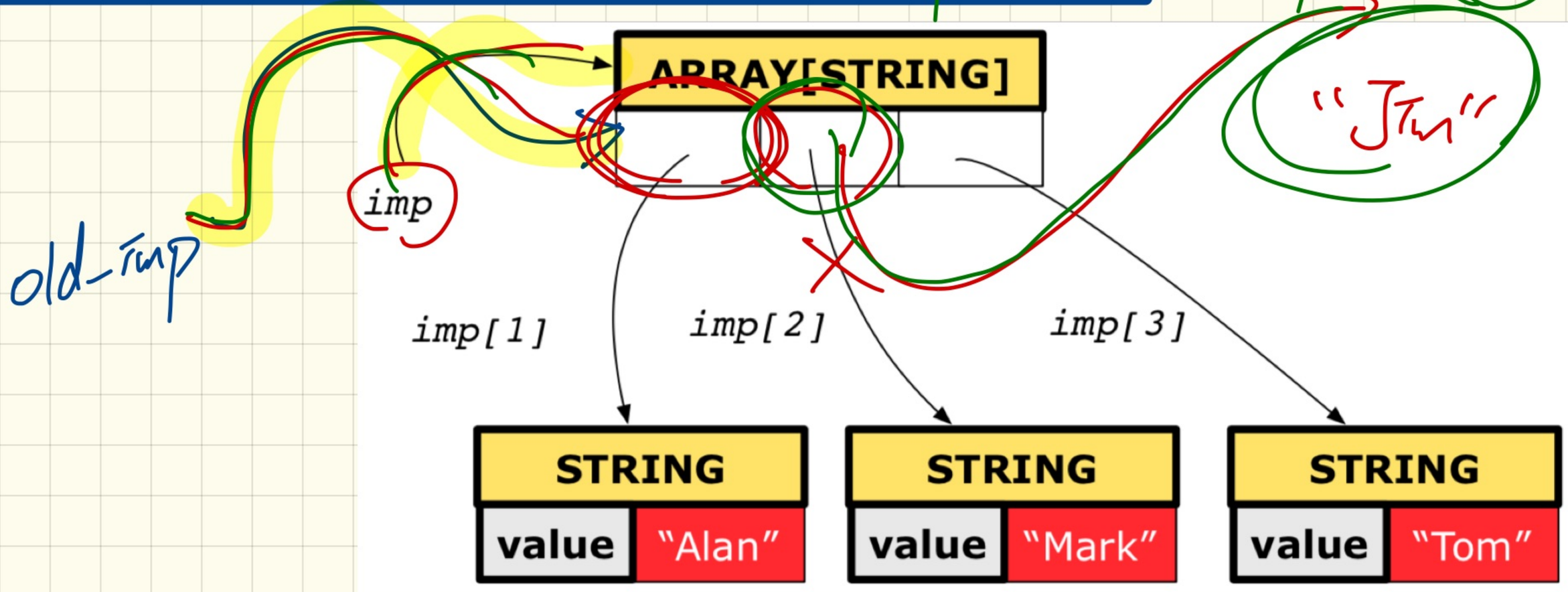
not good :(
we expect to see two versions being diff.

2
1
j

imp[1] =
imp[2] ~

old_imp[1] T
old_imp[2] T

"Jim"



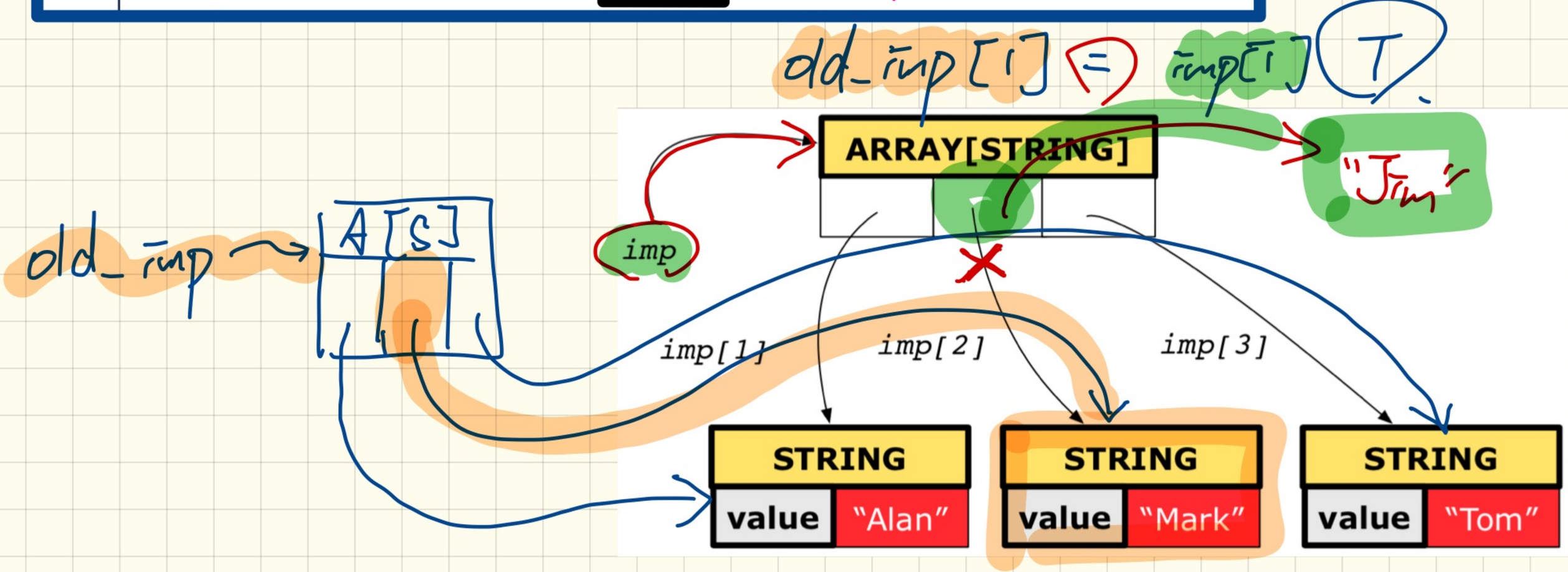
Copying Collection Objects : Shallow Copy & Make 1st-level changes

```
1  old_imp := imp.twin
2  Result := old_imp = imp   -- Result = false
3  imp[2] := "Jim"
4  Result :=
5    across 1 |..| imp.count is j
6    all imp [j] ~ old_imp [j]
7    end -- Result = [redacted]
```

imp[1] ~ old_imp[1] (T)

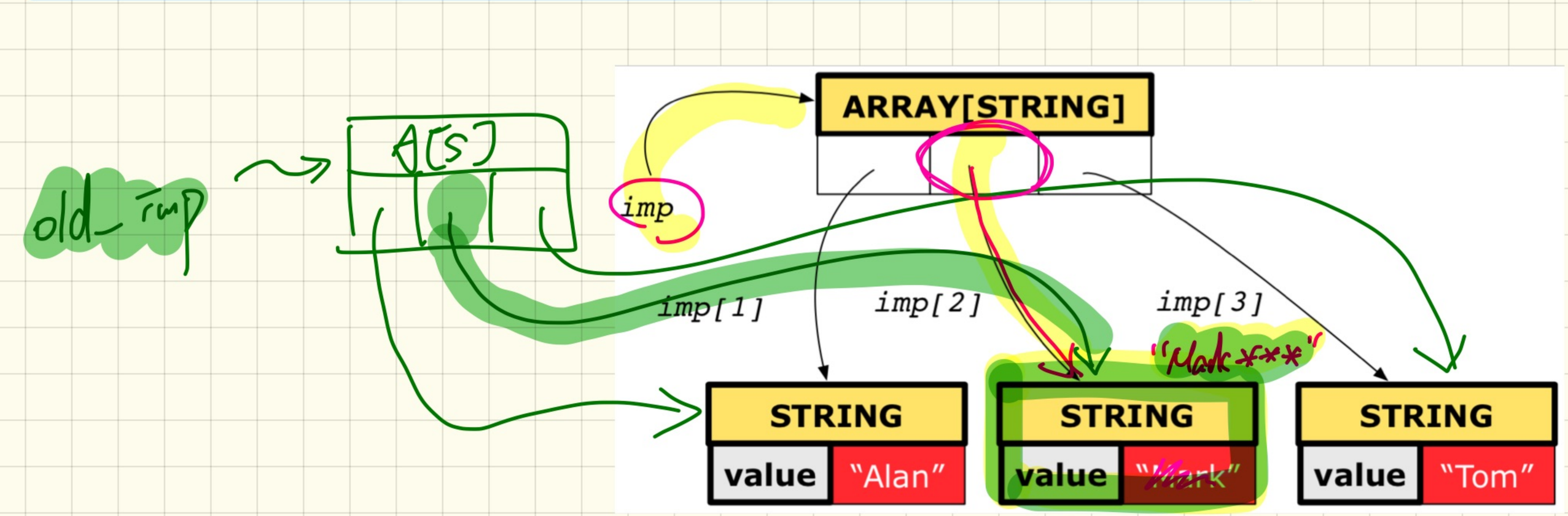
imp[2] ~ old_imp[2] (F)

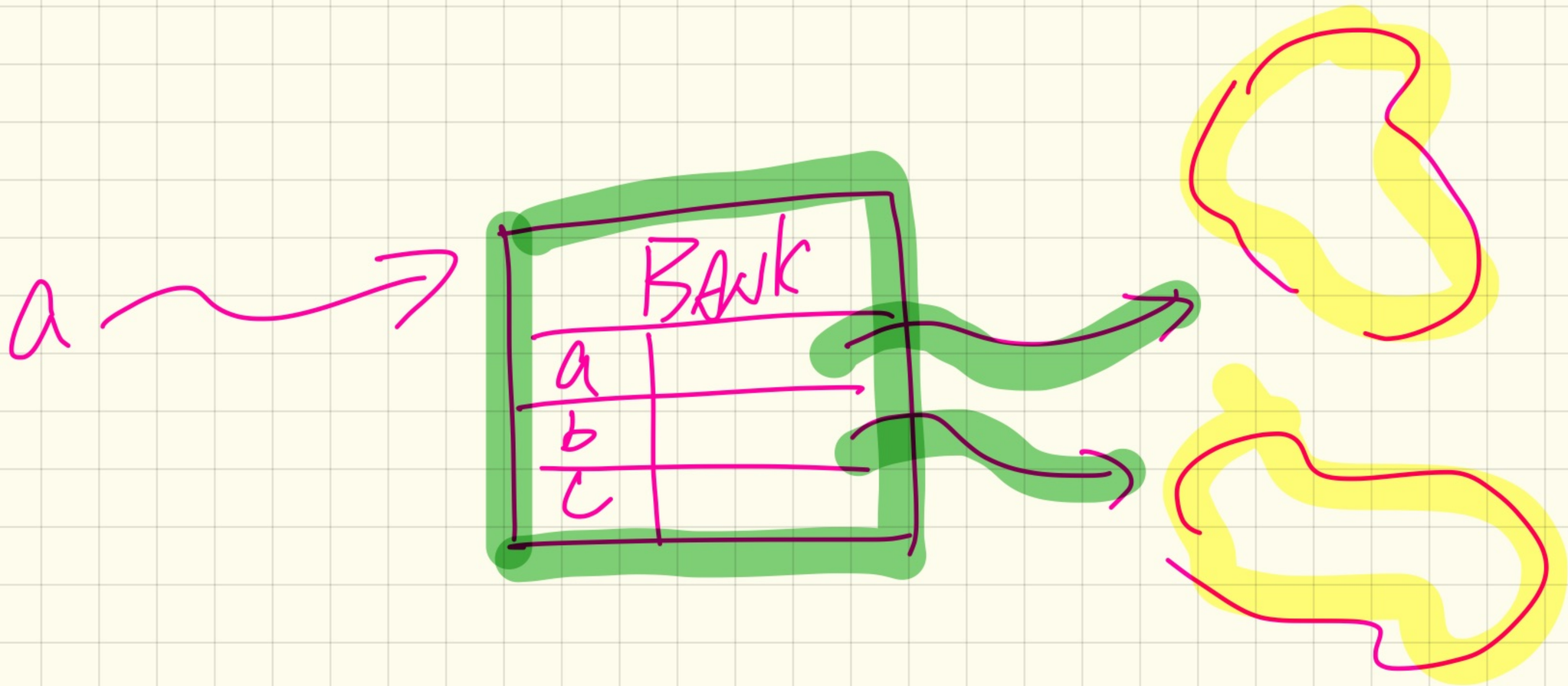
old_imp[1] = imp[1] (T)



Copying Collection Objects : Shallow Copy & Make 2nd-level changes

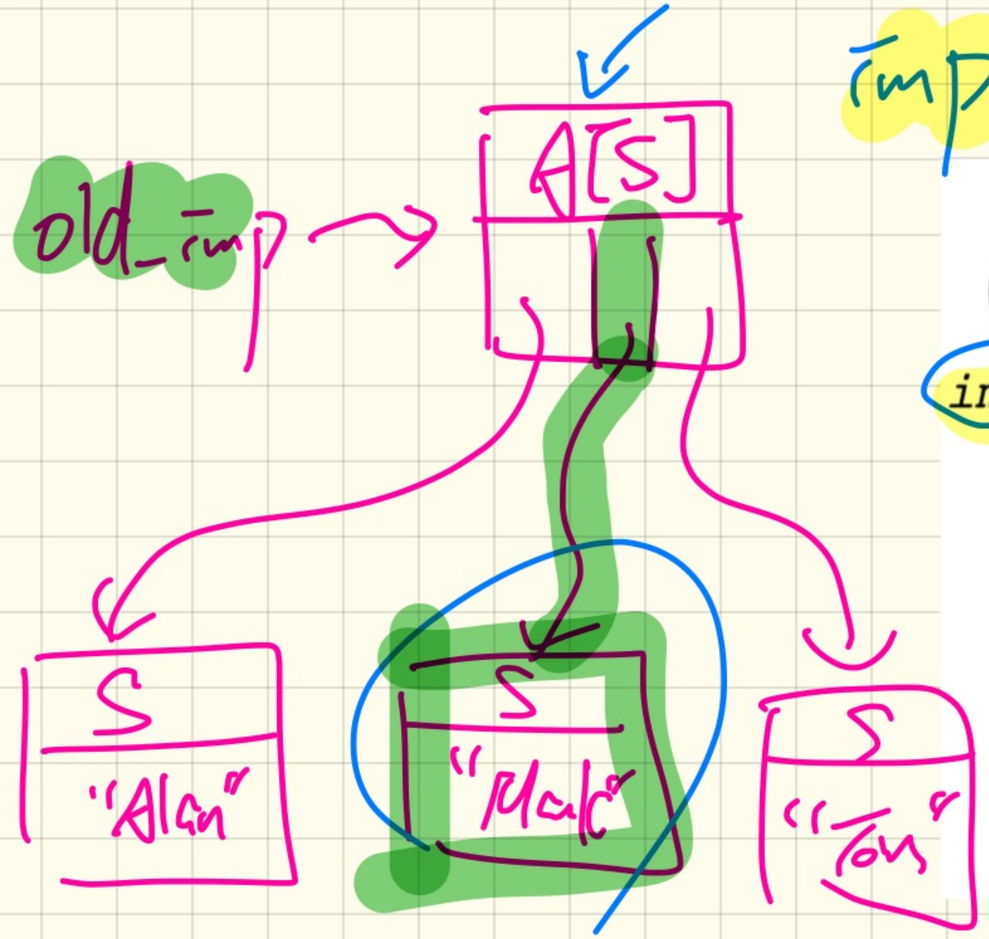
```
1  old_imp := imp.twin
2  Result := old_imp = imp   -- Result = false
3  imp[2].append ("***")
4  Result :=
5   across 1 |..| imp.count is j
6   all imp [j] ~ old_imp [j]
7   end -- Result =
```

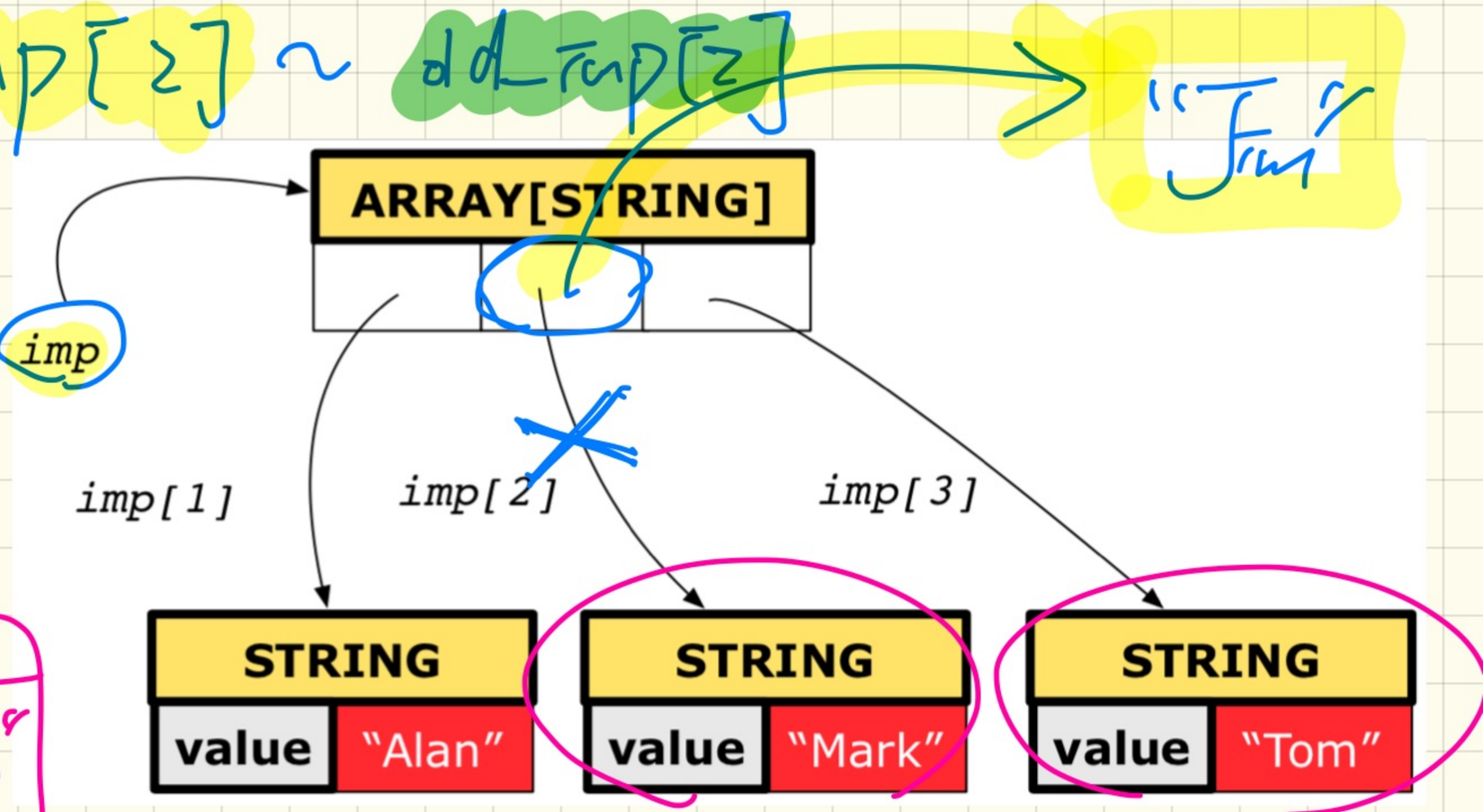



a
BANK
a
b
c

Copying Collection Objects: Deep Copy & Make 1st-level Changes

```
1  old_imp := imp.deep_twin
2  Result := old_imp = imp   -- Result = false
3  imp[2] := "Jim"
4  Result :=
5   across 1 |..| imp.count is j
6   all imp [j] ~ old_imp [j] end -- Result = [redacted]
```





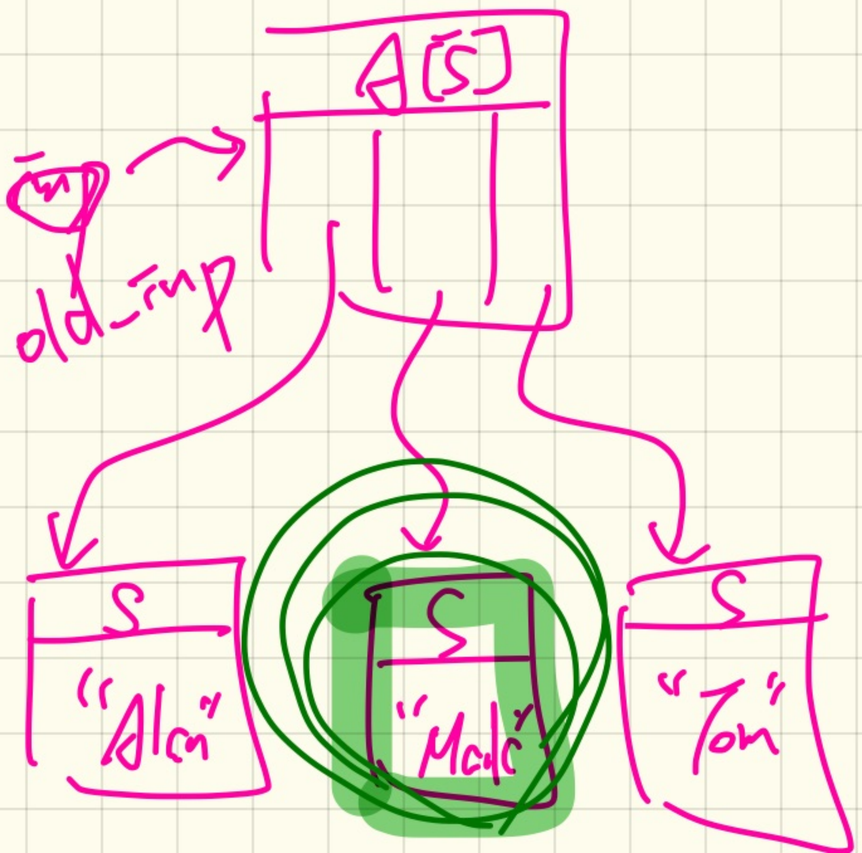
Copying Collection Objects : Deep Copy & Make 2nd-level changes

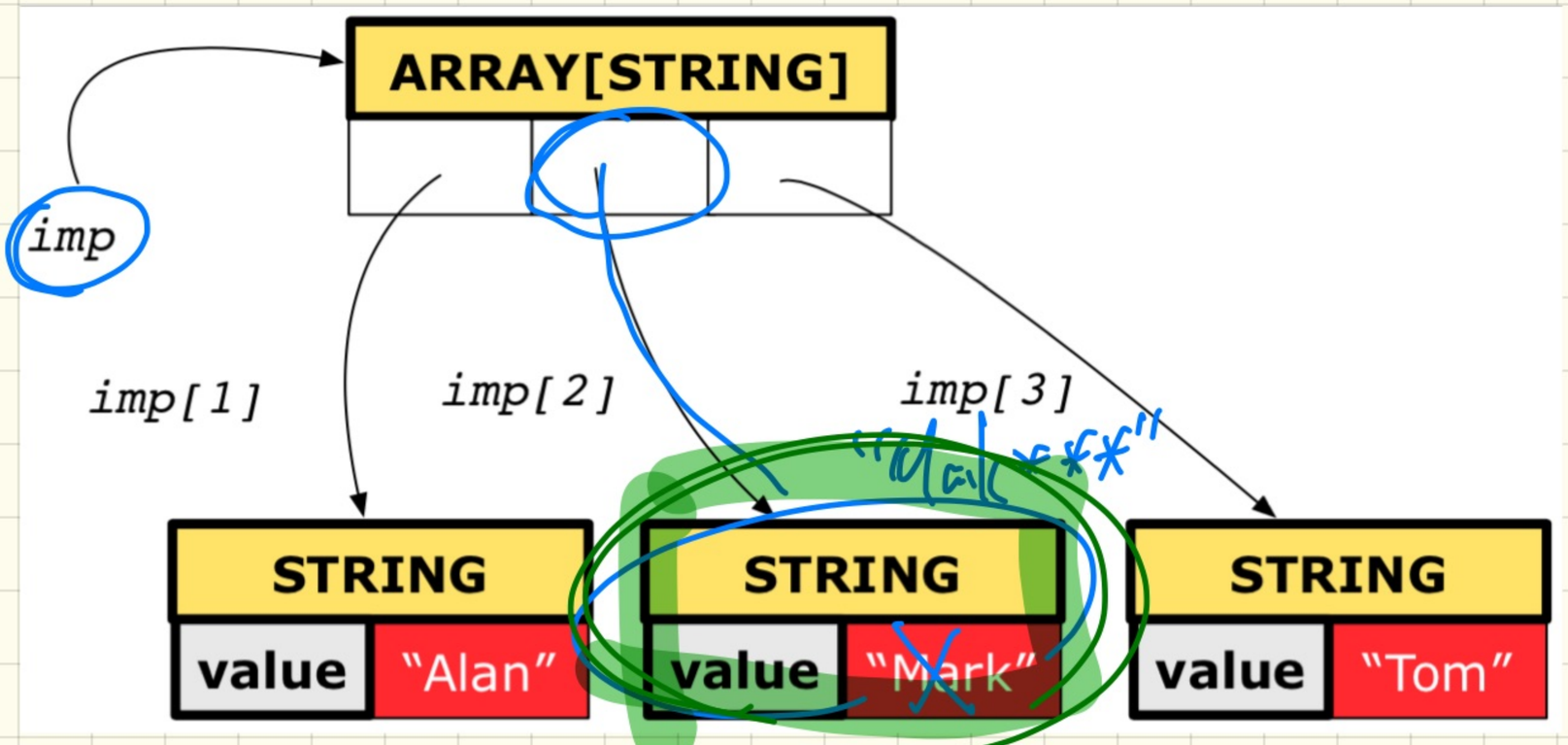
```
1  old_imp := imp.deep_twin
2  Result := old_imp = imp   -- Result = [redacted]
3  imp[2].append ("***")
4  Result :=
5  across 1 |..| imp.count is j
6   all imp [j] ~ old_imp [j] end   -- Result = [redacted]
```

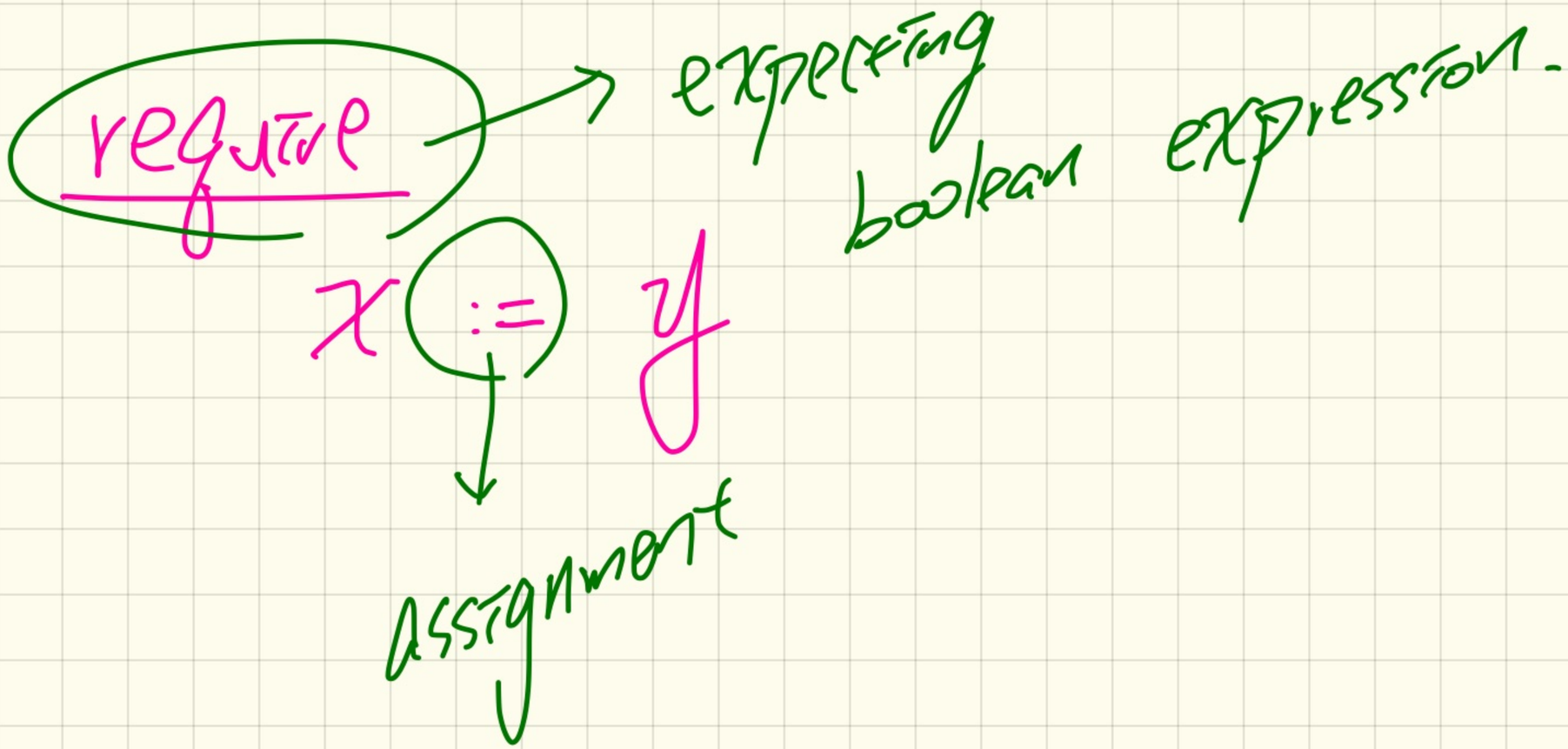
old

z

imp[z] ~ old_imp[z]



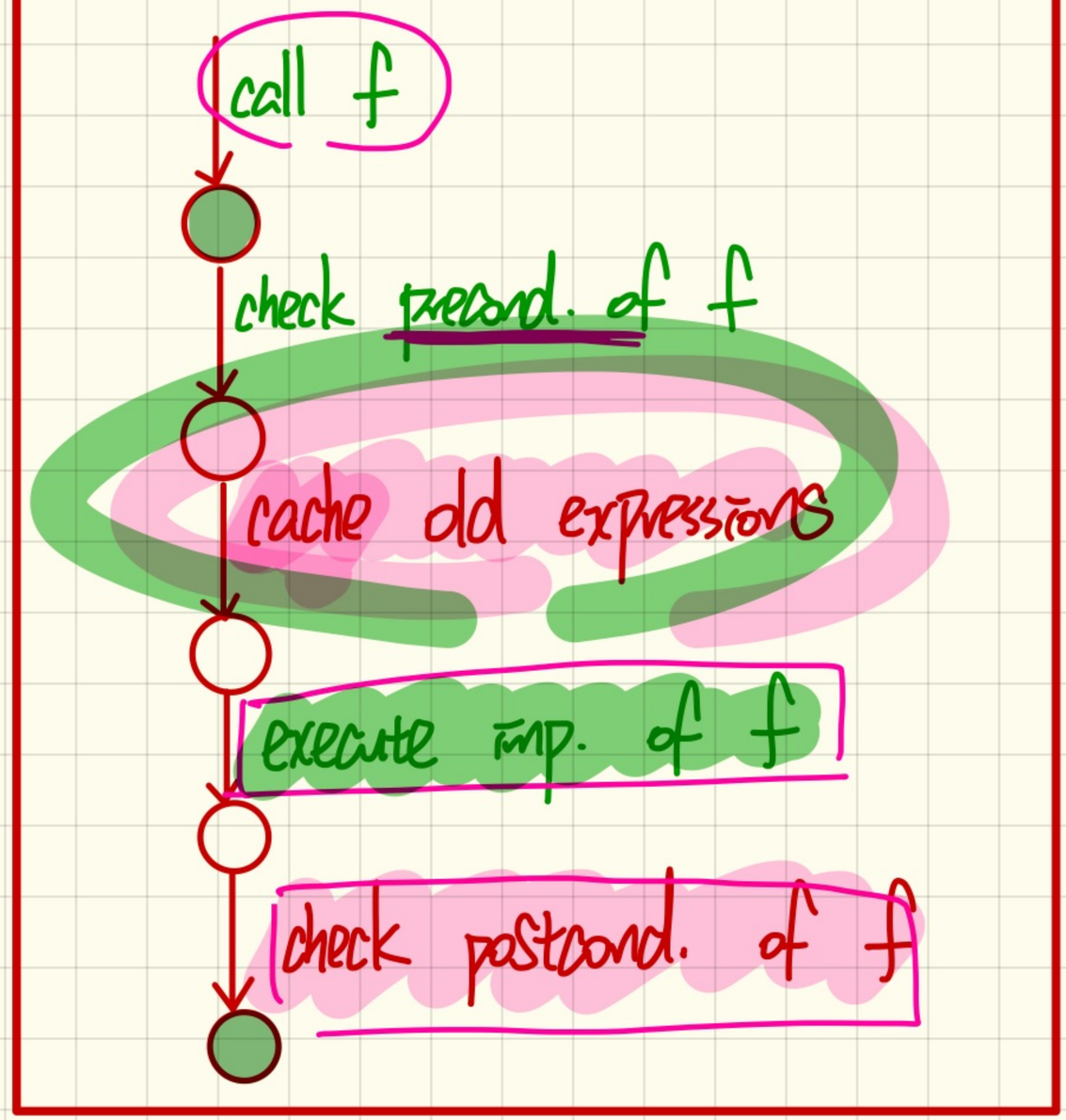



require
expecting
boolean expression.
x := y
assignment

Contract View

```
f
    require
        ...
    do
        ...
    ensure
        ... old expr ...
    end
```

Runtime Contract Checks



Caching Values for old Expressions in Postconditions

ensure (in context of BANK)

① old accounts[i].id

② (old accounts[i]).id

③ (old accounts[i].twin).id

④ (old accounts)[i].id

⑤ (old accounts.twin)[i].id

⑥ (old Current).accounts[i].id

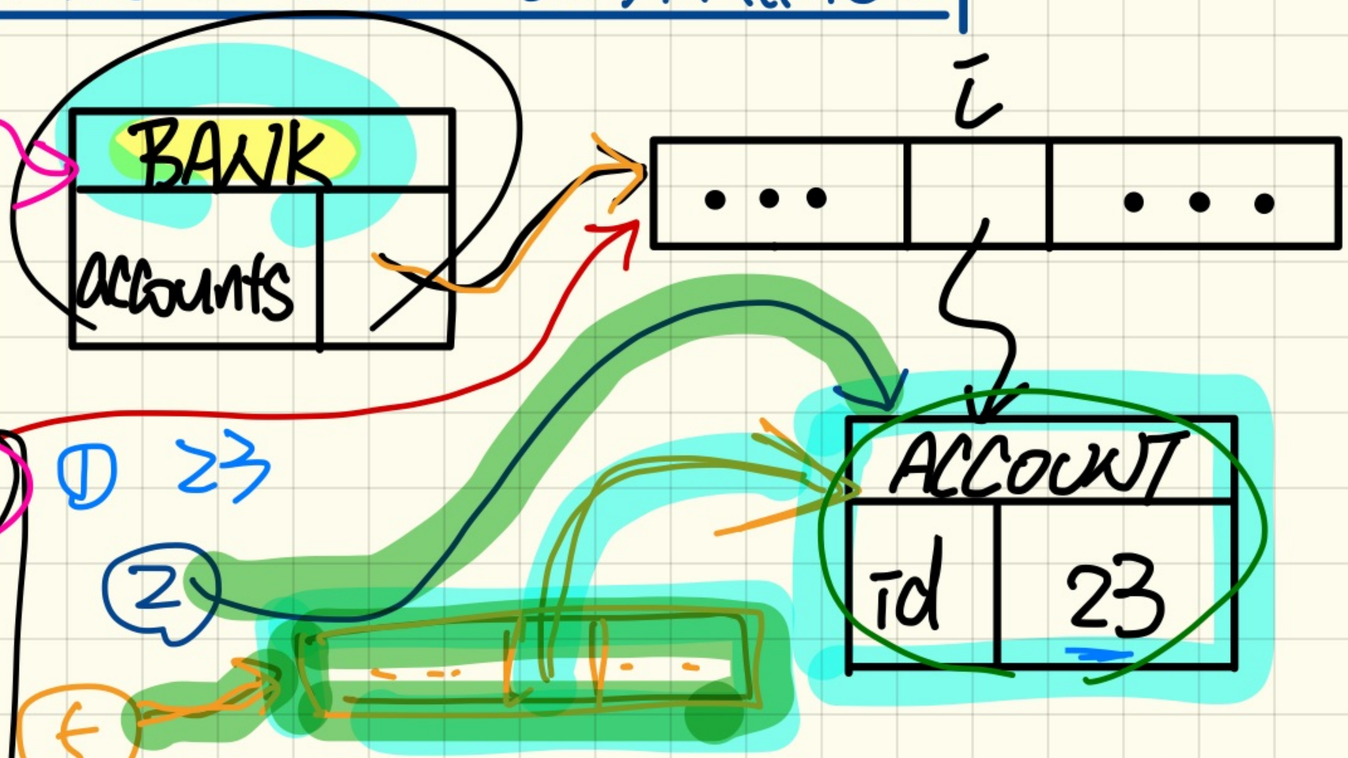
⑦ (old Current.twin).accounts[i].id

INT

shallow copy of account

class BANK
...
ensure
...
end

How to Cache at Runtime?



⑦ → BANK, accs → shallow copy of array

① := accounts[i].id

② := accounts[i]

⑤ := accounts.twin

⑦ := Current.twin

⑥ := Current

Use of old in across expression in Postcondition

```
class LINEAR_CONTAINER
create make
feature -- Attributes
  a: ARRAY[STRING]
feature -- Queries
  count: INTEGER do Result := a.count end
  get (i: INTEGER): STRING do Result := a[i] end
feature -- Commands
  make do create a.make_empty end
  update (i: INTEGER; v: STRING)
    do ...
    ensure -- Others Unchanged
      across
        1 |..| count as j
      all
        j.item /= i implies old get(j.item) ~ get(j.item)
      end
  end
end
```

old_get_j := get(j.item)

old (get) (j.item) X

(old Current).get(j.item)

Hint: What value will be cached at runtime before executing the imp. of update?

Programming Client-Supplier Relation

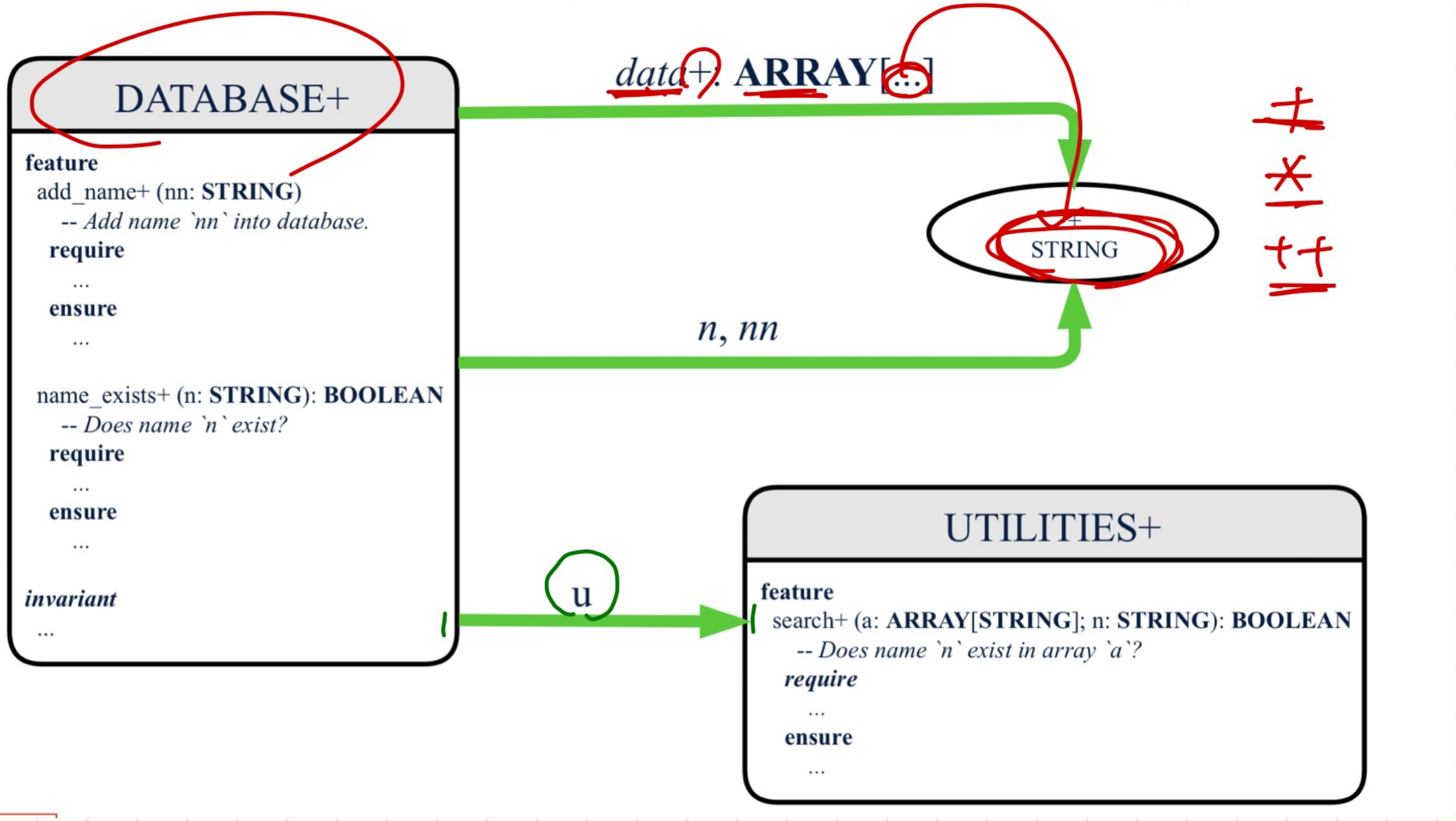
```
class DATABASE
feature {NONE} -- implementation
  data: ARRAY[STRING]
feature -- Commands
  add_name (nn: STRING)
      -- Add name `nn' to database.
    require ... do ... ensure ... end

  name_exists (n: STRING): BOOLEAN
      -- Does name `n' exist in database?
    require ...
    local
      u: UTILITIES
    do ... ensure ... end
invariant
  ...
end
```

```
class UTILITIES
feature -- Queries
  search (a: ARRAY[STRING]; n: STRING): BOOLEAN
      -- Does name `n' exist in array `a'?
    require ... do ... ensure ... end
end
```

data : ARRAY [STRING]

Presenting CS Relation in Diagram: Approach 1

If `STRING` is to be emphasized, label is `data: ARRAY[...]`, where ... denotes the supplier class `STRING` being pointed to.



Presenting CS Relation in Diagram: Approach 2

If `ARRAY` is to be emphasized, label is `data`.

The supplier's name should be complete: `ARRAY[STRING]`