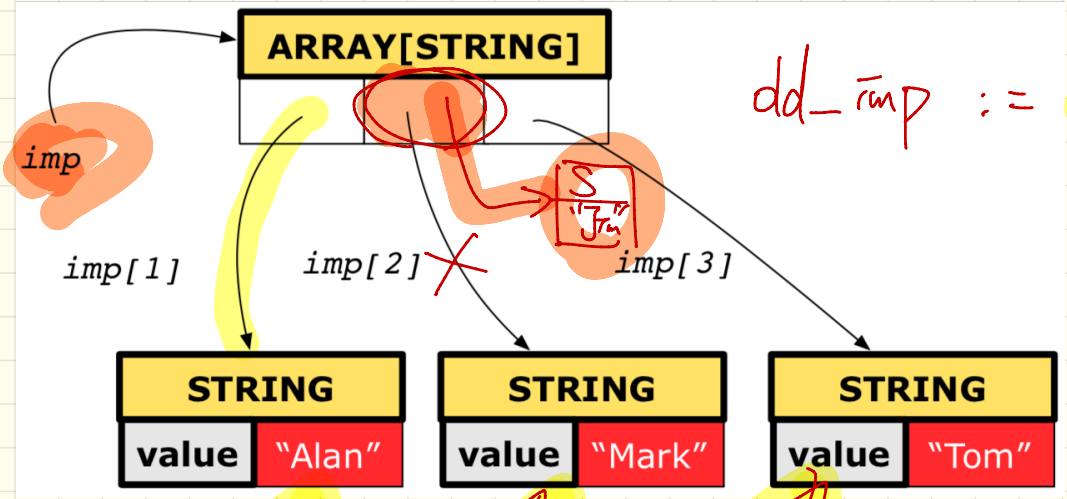
Thursday Sep. 20

Lecture 5

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



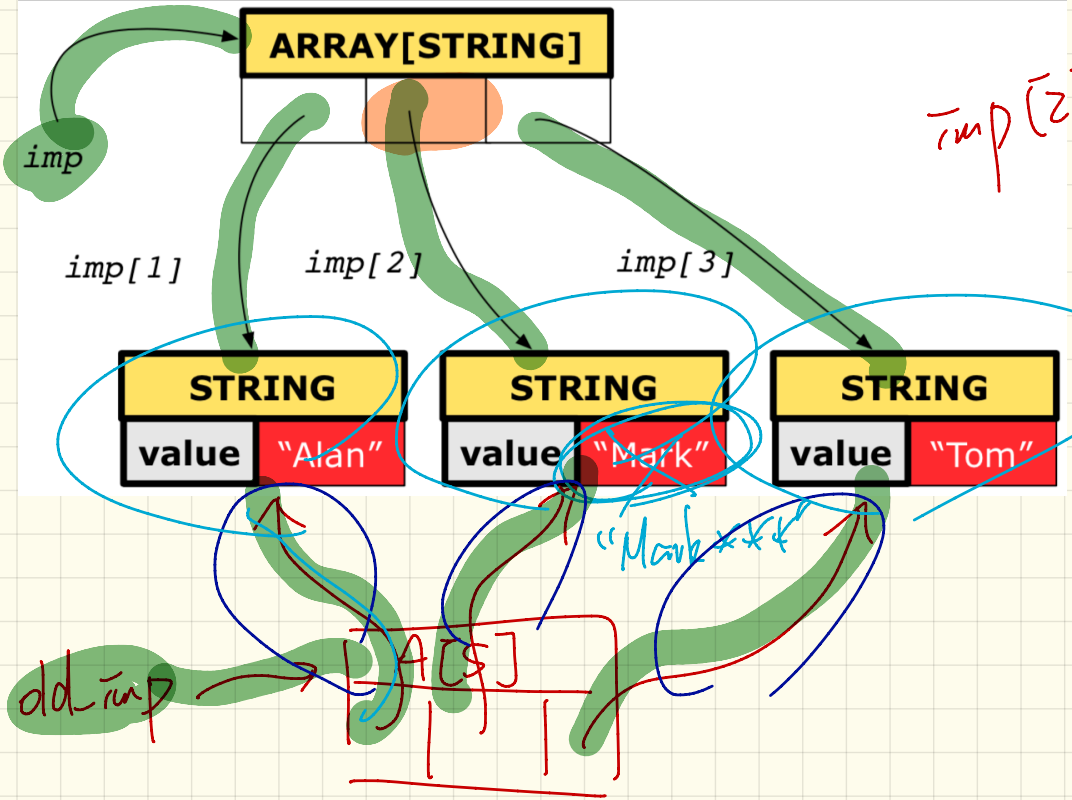
dd_imp := imp.twin

old_imp → A[S]

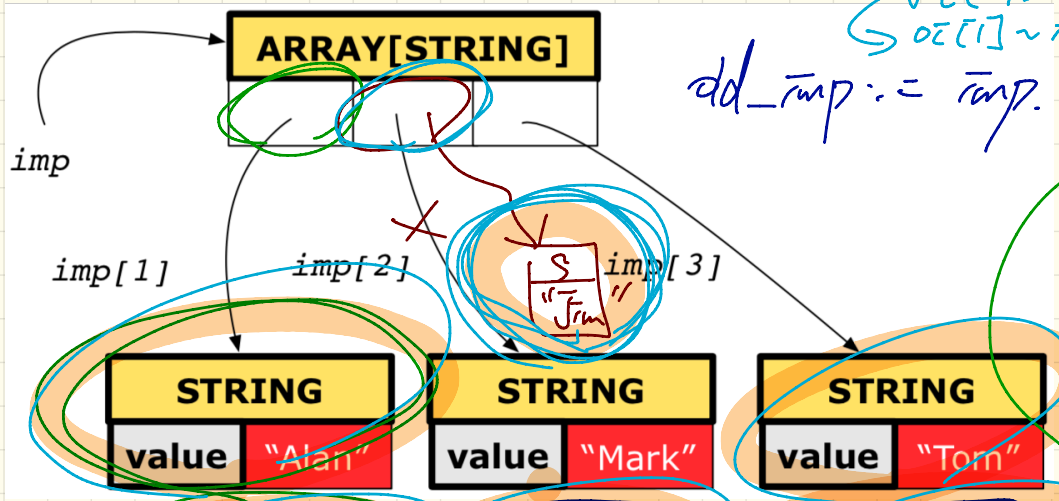
imp[1] = dd_imp[1] T

imp = dd_imp T.

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



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



$\forall i \in 1..3 \cdot old\_imp[i] \sim imp[i]$

$\hookrightarrow or[1] \sim imp[1]$ and $or[2] \sim imp[2]$ and $or[3] \sim imp[3]$

old_imp := imp.deep_twin

T
F
T

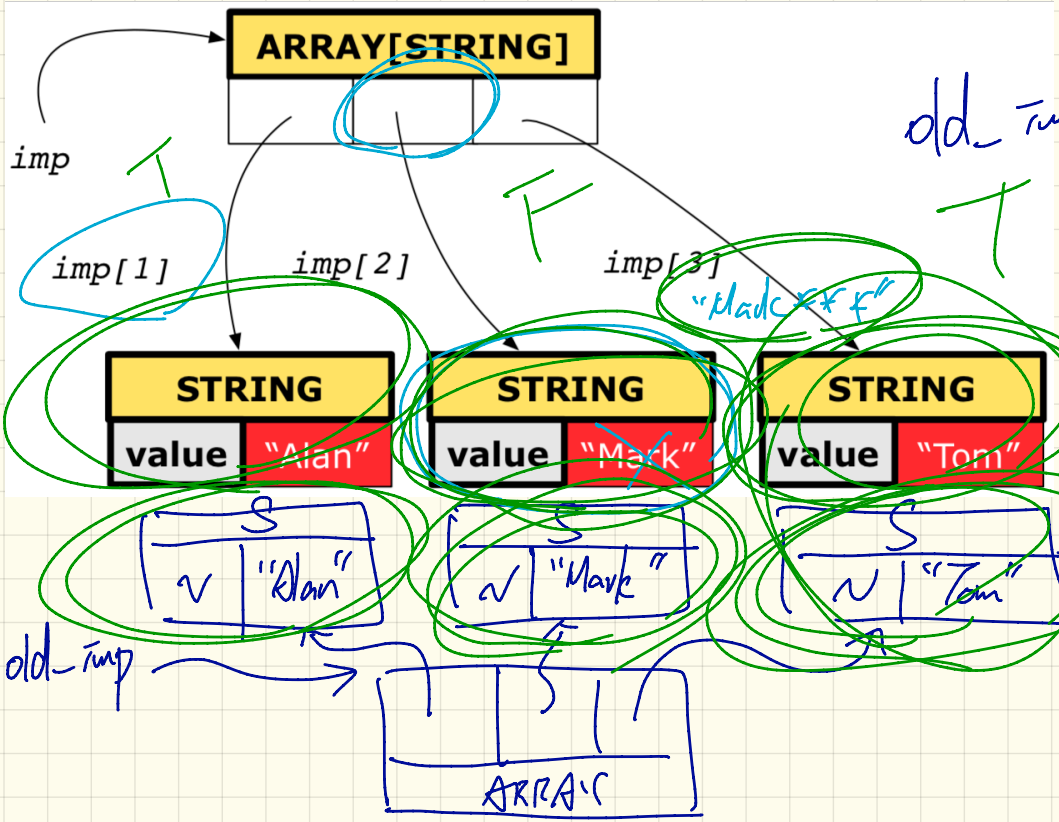
F

imp = old_imp F

imp[1] = old_imp[1]

imp[1] ~ old_imp[1] T

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



Caching Values for old Expressions in Postconditions

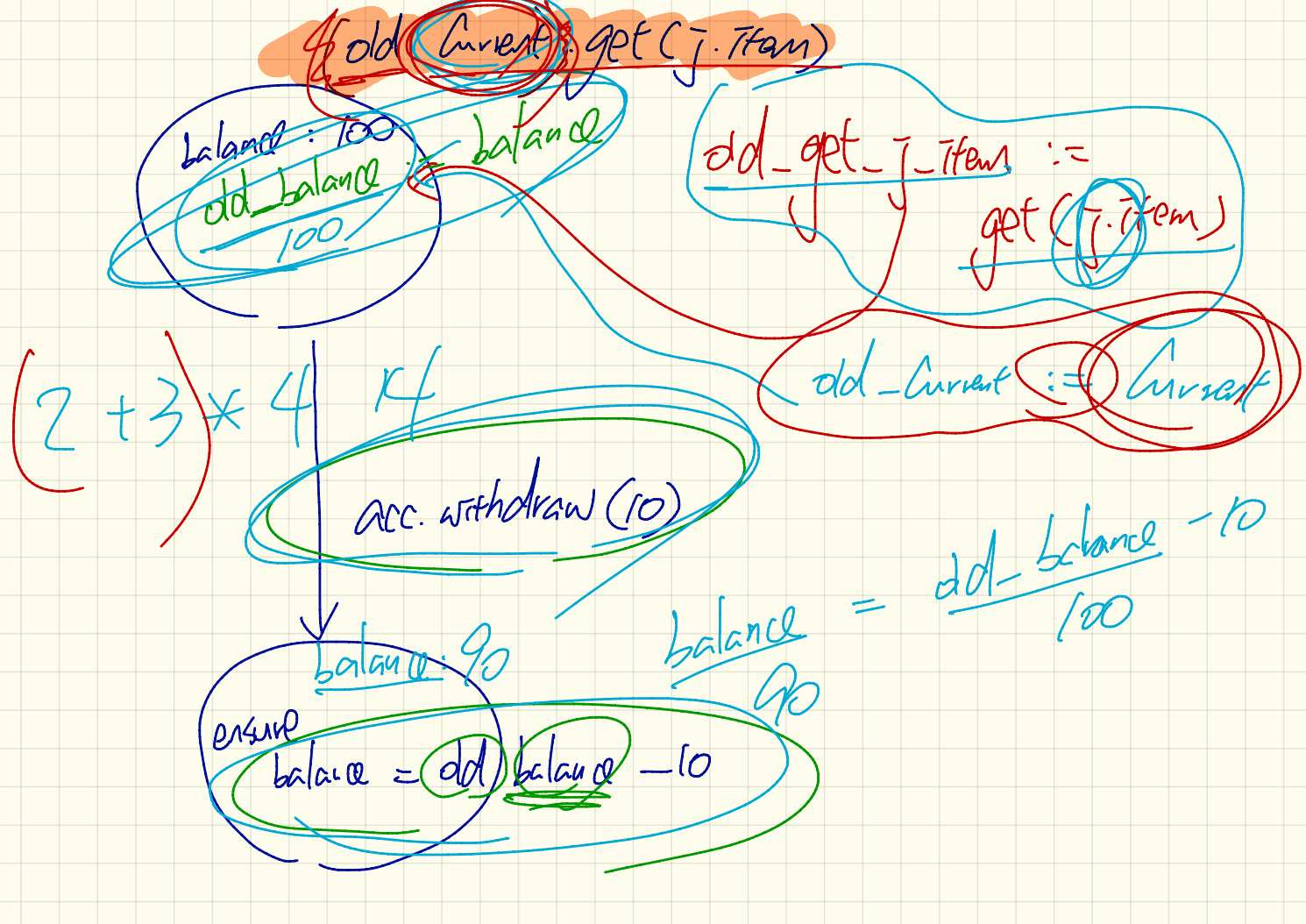
do x old ensure

ensure (current class Bank)	How to cache?
old balance = balance - a	old_balance := balance
old accounts[i].id	
(old accounts[i]).id	
(old accounts)[i].id	
(old current).accounts[i].id	

old accounts[i].id.item(2)

old_current := current
old_c_t := current.twin
old_c_d_t := current.dt

① old current
② old current.twin
③ old current.deep_twin


old (current).get (j.item)
balance : 100
old_balance
100
balance
old_get_j_item :=
get (j.item)
old_current := current
(2+3)*4
K
acc.withdraw (10)
balance : 90
balance
90
balance = old_balance - 10
100
ensure
balance = old balance - 10

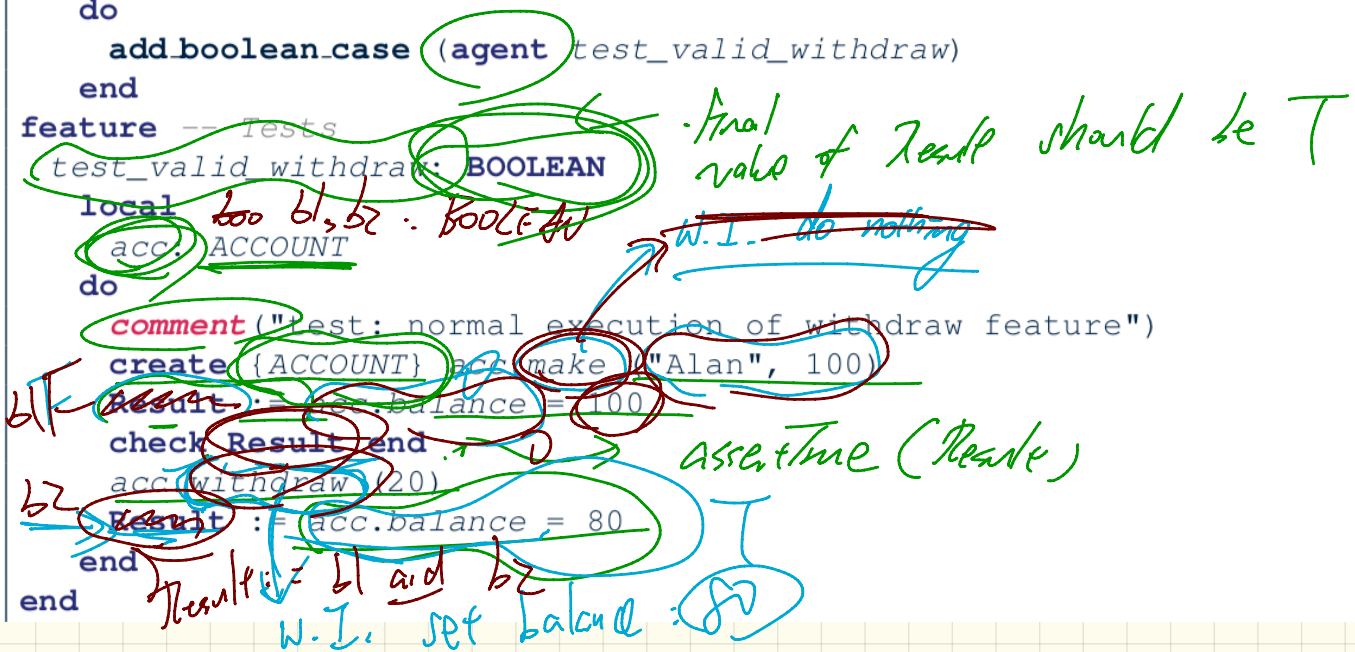
Use of old in across expression in Postcondition

```
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
```

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

Test for Success

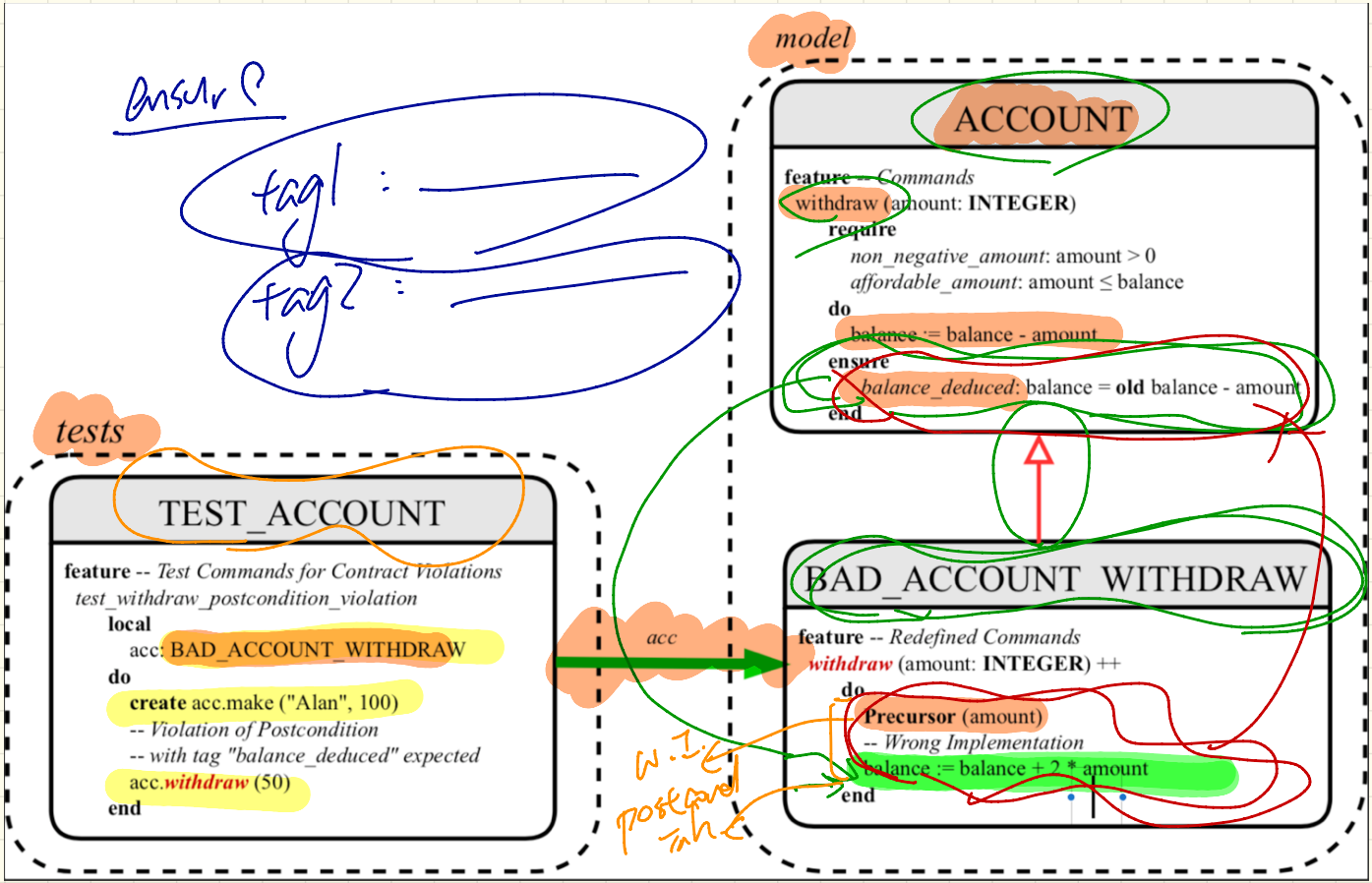
```
class TEST_ACCOUNT
inherit ES_TEST
create make
feature -- Add tests in constructor
  make
    do
      add_boolean_case (agent test_valid_withdraw)
    end
feature -- Tests
  test_valid_withdraw: BOOLEAN
    local
      acc: ACCOUNT
    do
      comment("test: normal execution of withdraw feature")
      create {ACCOUNT} acc.make ("Alan", 100)
      Result := acc.balance = 100
      check Result end
      acc.withdraw (20)
      Result := acc.balance = 80
    end
end
```



Test for Precondition Violation

```
class TEST_ACCOUNT
inherit ES_TEST
create make
feature -- Add tests in constructor
  make
    do
      add_violation_case_with_tag ("non_negative_amount",
        agent test_withdraw_precondition_violation)
    end
feature -- Tests
  test_withdraw_precondition_violation
    local
      acc: ACCOUNT
    do
      comment("test: expected precondition violation of withdraw")
      create {ACCOUNT} acc.make ("Mark", 100)
      -- Precondition Violation
      -- with tag "non_negative_amount" is expected.
      acc.withdraw (-1000000)
    end
end
```

Test for Postcondition Violation : Architecture



Test for Postcondition Violation: Code

```
class ACCOUNT
create
	make
feature -- Attributes
	owner : STRING
	balance : INTEGER
feature -- Constructors
	make(nn: STRING; nb: INTEGER)
		require -- precondition
			positive_balance: nb > 0
		do
			owner := nn
			balance := nb
		end
feature -- Commands
	withdraw(amount: INTEGER)
		require -- precondition
			non_negative_amount: amount > 0
			affordable_amount: amount <= balance -- problema
		do
			balance := balance - amount
		ensure -- postcondition
			balance_deducted: balance = old balance - amount
		end
invariant -- class invariant
	positive_balance: balance > 0
end
```

```
class TEST_ACCOUNT
inherit ES_TEST
create make
feature -- Constructor for adding tests
  make
    do
      add_violation_case_with_tag ("balance_deducted",
        agent test_withdraw_postcondition_violation)
    end
feature -- Test commands (test to fail)
  test_withdraw_postcondition_violation
    local
      acc: BAD_ACCOUNT_WITHDRAW
    do
      comment ("test: expected postcondition violation of withdraw")
      create acc.make ("Alan", 100)
      -- Postcondition Violation with tag "balance_deduced" to occur.
      acc.withdraw (50)
    end
end
```

```
class
  BAD_ACCOUNT_WITHDRAW
inherit
  ACCOUNT
    redefine withdraw end
create
  make
feature -- redefined commands
  withdraw(amount: INTEGER)
    do
      Precursor(amount)
      -- Wrong implementation
      balance := balance + 2 * amount
    end
end
```