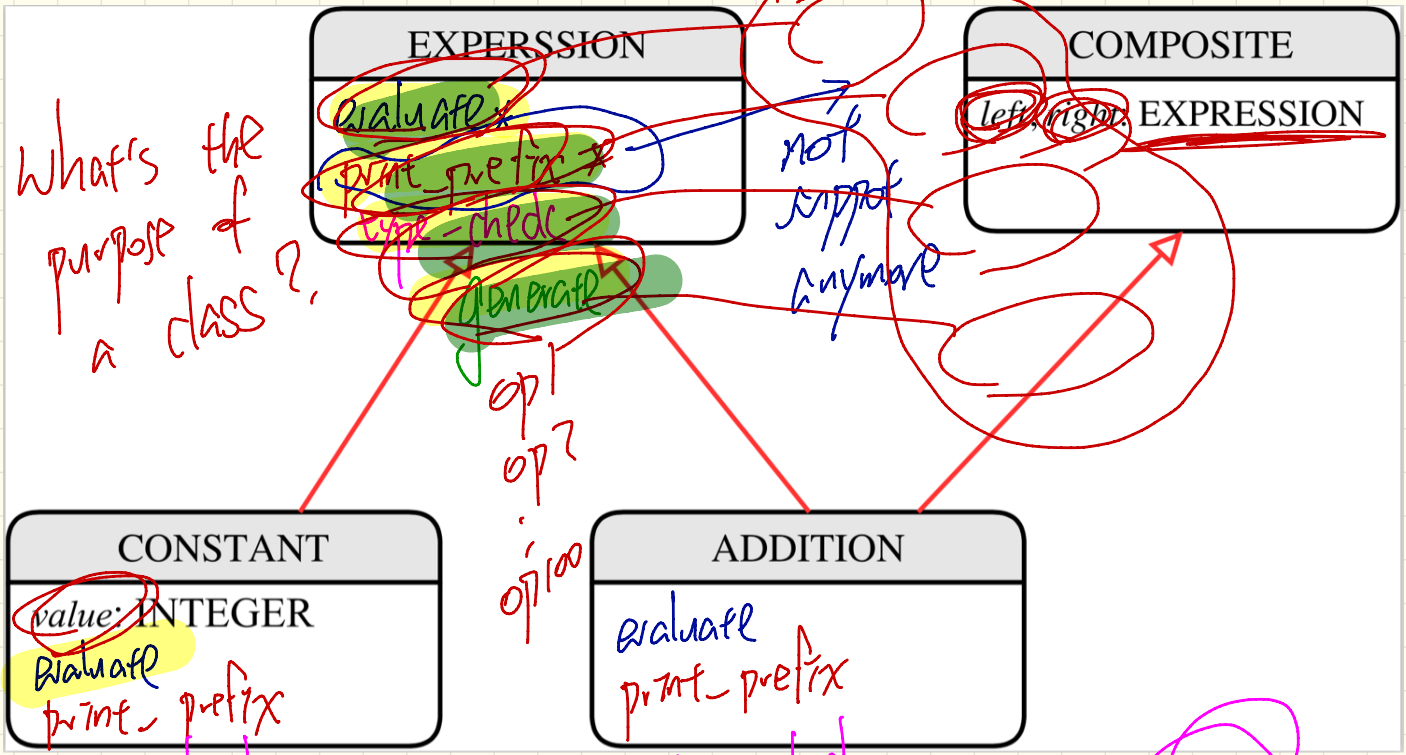
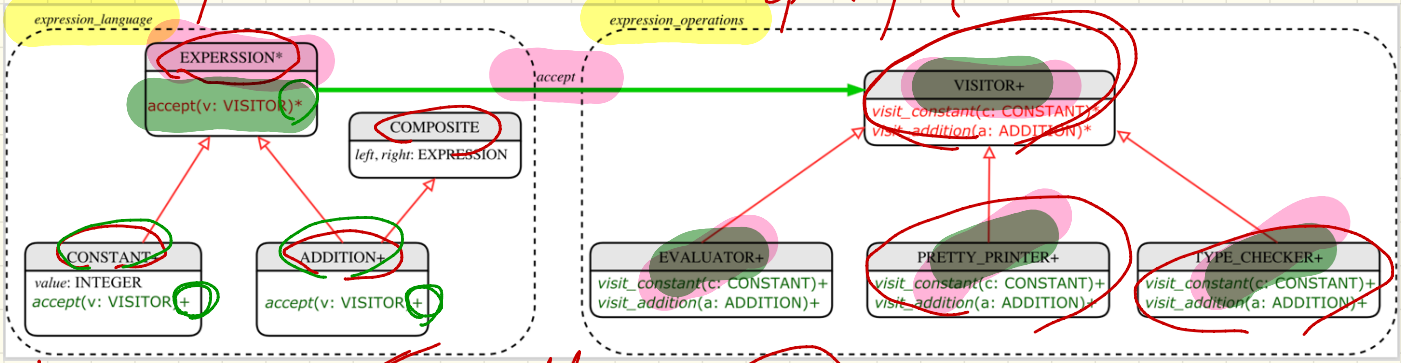
Lecture 17

Wednesday Nov. 8


EXPERSSION
evaluate
print_prefix
type_check
generate
COMPOSITE
left, right: EXPRESSION
CONSTANT
value: INTEGER
evaluate
print_prefix
type_check
generate
ADDITION
evaluate
print_prefix
type_check
generate
What's the purpose of a class?
not support anymore
op1
op2
:
op100
op1
op2
:
op100
+
3
4



add.accept(v)

add

+

3

4

ADDITION.

accept (v: VISITOR) *

CONSTANT

accept (v: VISITOR)

do

v.visit_constant(Current)

end

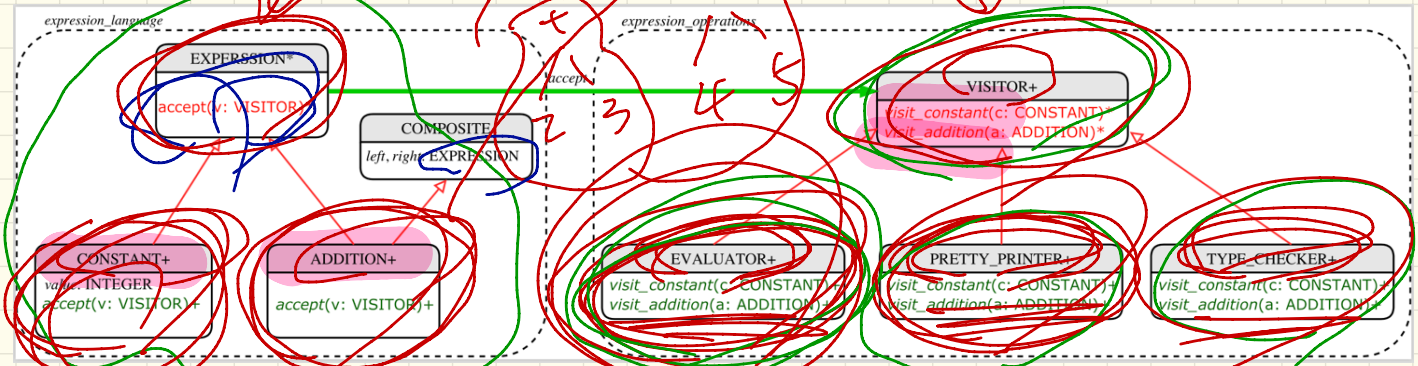
ADDITION

accept (v: VISITOR)

do

v.visit_addition(Current)

end



VISITOR

[visit_constant (c: CONSTANT)
visit_addition (a: ADDITION)] correspond to the list of descendats of EXPRESSION

PRETTY_PRINTER → for pretty printing

visit_constant (c: CONSTANT)
do to.print (c.value) end

(effective) → for pretty printing.

visit_addition (a: ADDITION)
do a.left.accept (current)
to.print ("+")
a.right.accept (Cur.)
end

class PRETTY_PRINTER

visit_addition (a: ADDITION)

:

class EVALUATOR

value: INTEGER

visit_addition (a: ADDITION)

local

left_eval: EVALUATOR

right_eval: EVALUATOR

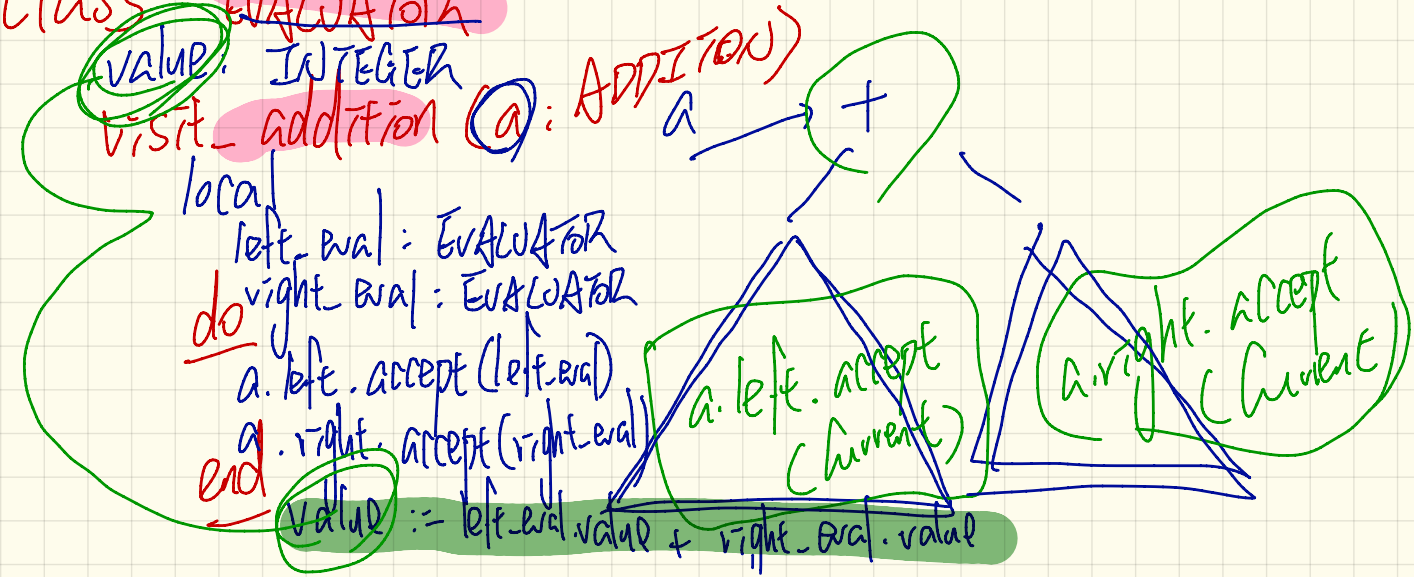
do

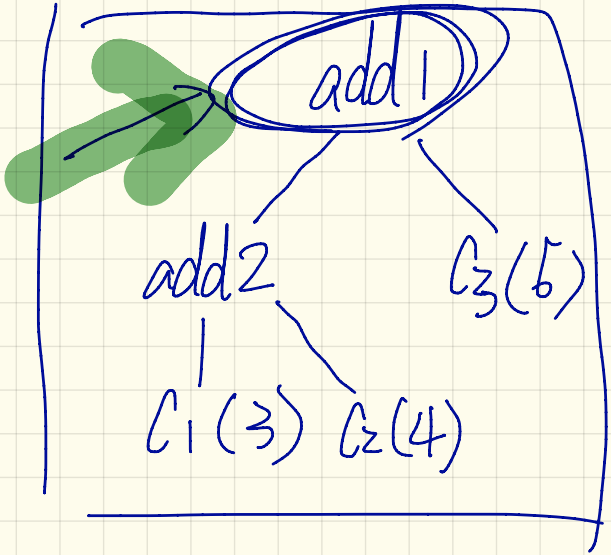
a.left.accept (left_eval)

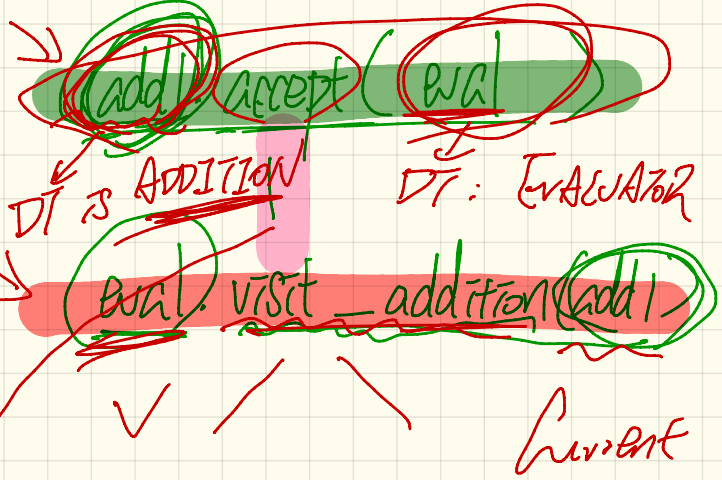
a.right.accept (right_eval)

end

value := left_eval.value + right_eval.value







eval: EVALUATOR

create {EVALUATOR} eval. make

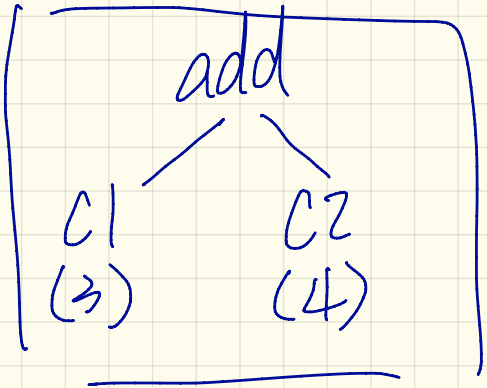
1st patch:
DT of add1 is ADDITION
⇒ version of accept
in ADDITION
is called.

Double dispatch

add.accept(v)

DT: ADDITION

1st patch: call the version of accept in ADDITION.



add: EXPRESSION

create {ADDITION} add.make

v.visit_addition(add)

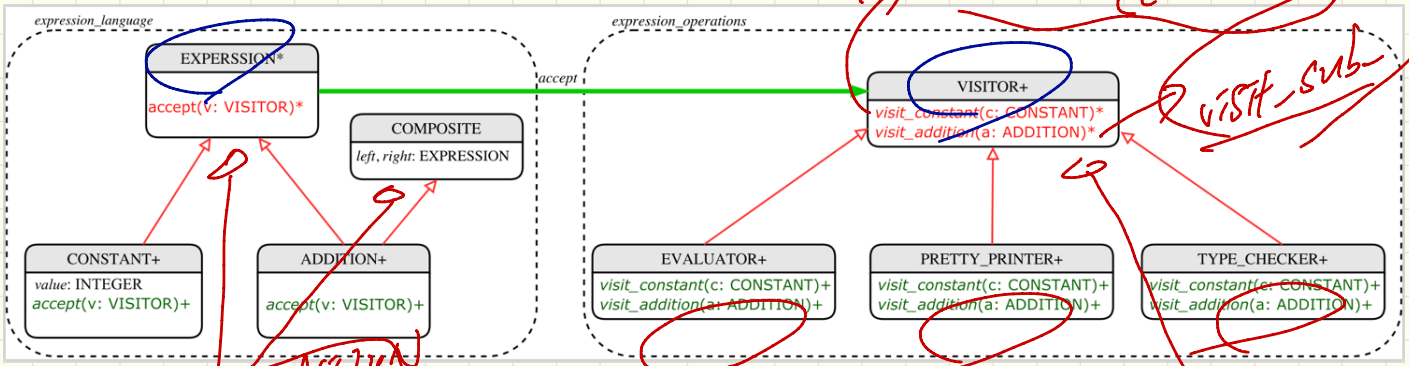
DT: EVALUATOR

2nd patch: call version of visit_addition in EVALUATOR

v: VISITOR

create {EVALUATOR} v.make

add.left.accept(left_eval) add.right.accept(right_eval)



visit_expression (e:

visit_sub-

SUBTRACTION

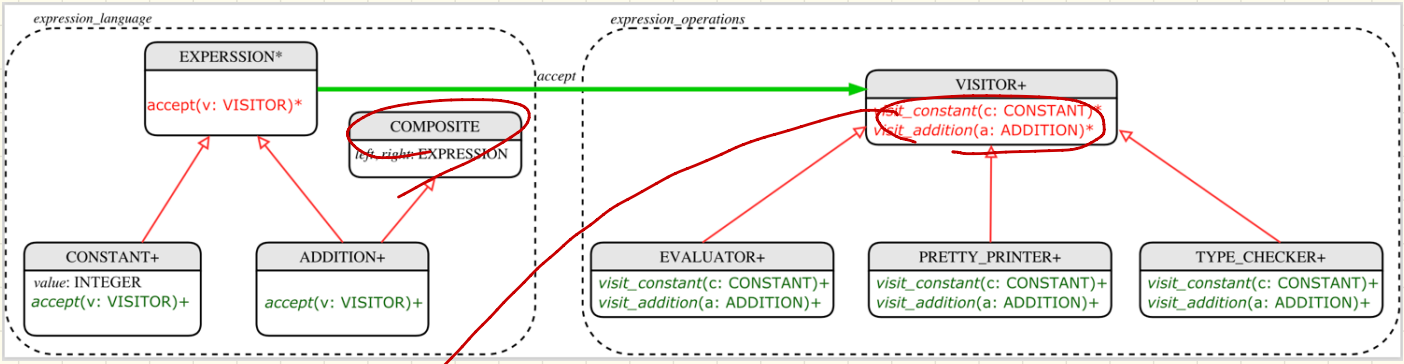
Extensions ?

visit_expression

SIMPLIFIER

- add a new EXPRESSION (e.g. SUBTRACTION)

- add a new VISITOR (e.g. SIMPLIFIER)



visit_expression(e: EXPRESSION) X

EVALUATOR

visit_expression(e: EXPRESSION)
do
[e.left]
end