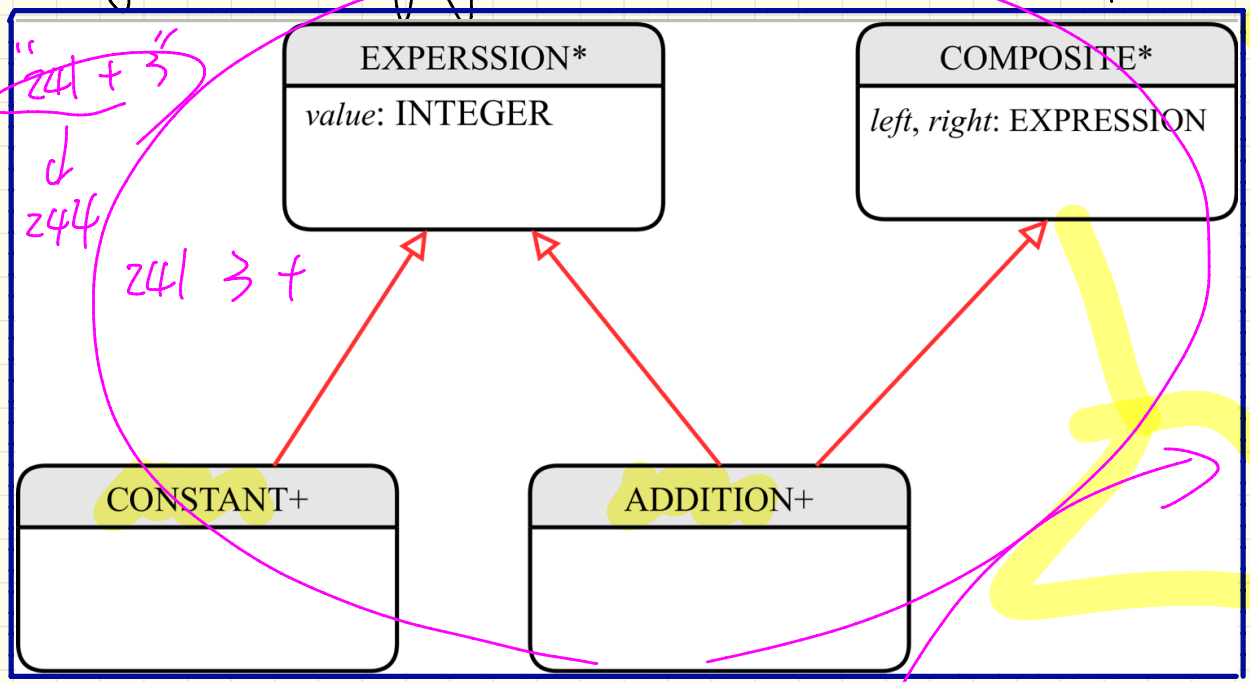
Tuesday Nov. 6

Lecture 16

- Exam: Sunday Dec. 9 7pm
- Midterm results available this Thursday
- Lab 3 (programming) marks available early Friday
- Project released by next Wed.
  (3 weeks)

Design of a Language Application: Open-Closed Principle



Operations: evaluate, print_prefix, print_postfix, type_check

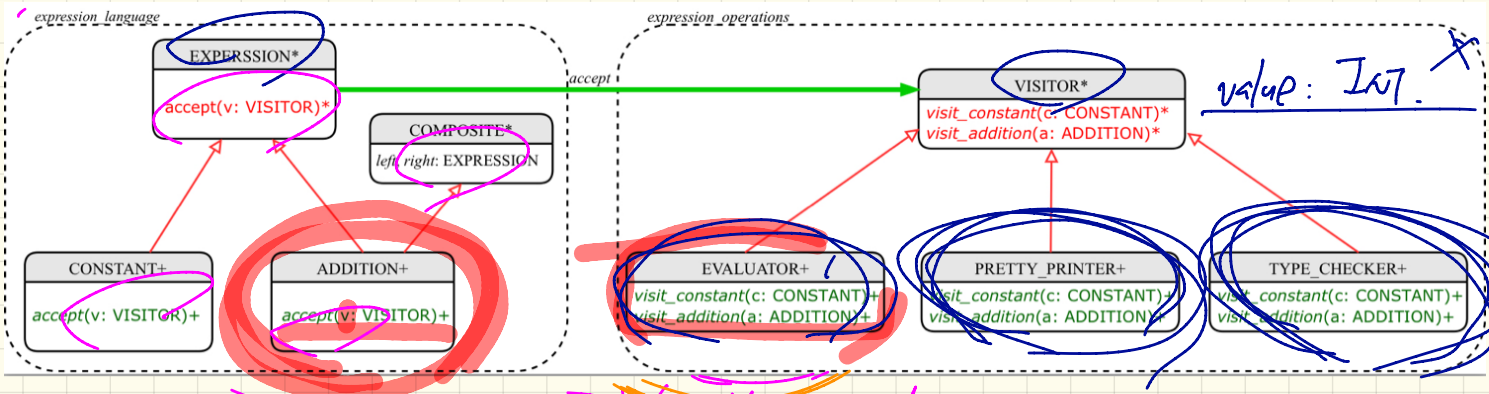
Operations

gene_ [illegible]

	Structure	Operations
Alt. 1	open	closed
Alt. 2	closed	open

Visitor Design Pattern: Architecture

add accept 234 + "a"



effective descendants of EXPRESSION

create a visit_x feature in VISITOR

How to Use Visitors

add.accept(v) ⇒

c1.accept(v) → 1 3

c2.accept(v) → 2

```
1  test_expression_evaluation: BOOLEAN
2    local add, c1, c2: EXPRESSION ; v: VISITOR
3    do
4      create {CONSTANT} c1.make (1) ; create {CONSTANT} c2.make (2)
5      create {ADDITION} add.make (c1, c2)
6      create {EVALUATOR} v.make
7      add.accept (v)
8      check attached {EVALUATOR} v as eval then
9        Result := eval.value = 3
10     end
11   end
```

what you have to write

e: 234 + 2

236

eval.visit_add(e)

PP.visit_add(e)

"234 2 +"

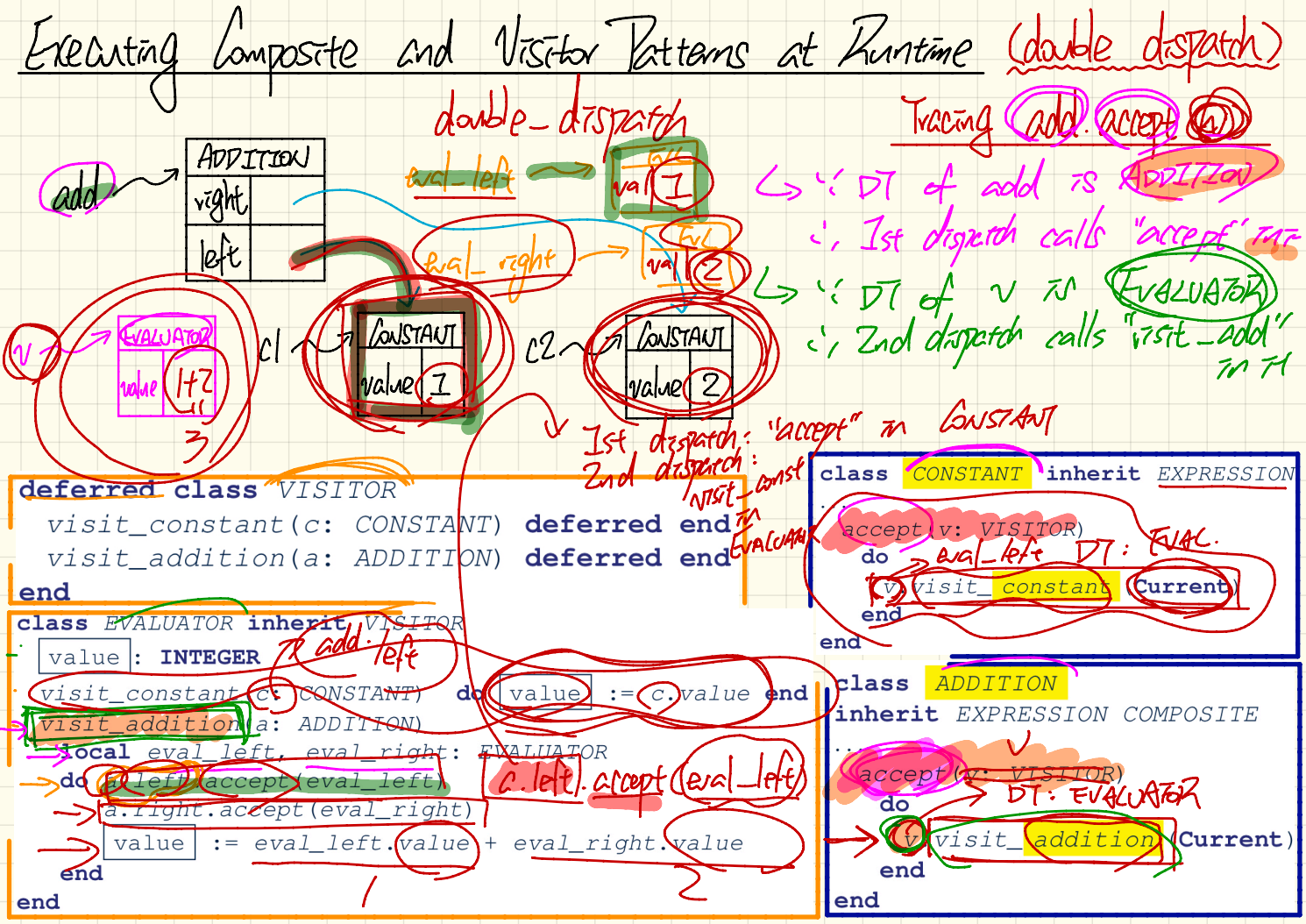
Visitor Design Pattern : Implementation

```
1  test_expression_evaluation: BOOLEAN
2   local add, c1, c2: EXPRESSION ; v: VISITOR
3   do
4    create {CONSTANT} c1.make (1) ; create {CONSTANT} c2.make (2)
5    create {ADDITION} add.make (c1, c2)
6    create {EVALUATOR} v.make
7    add.accept (v)
8    check attached {EVALUATOR} v as eval then
9     Result := eval.value = 3
10   end
11  end
```

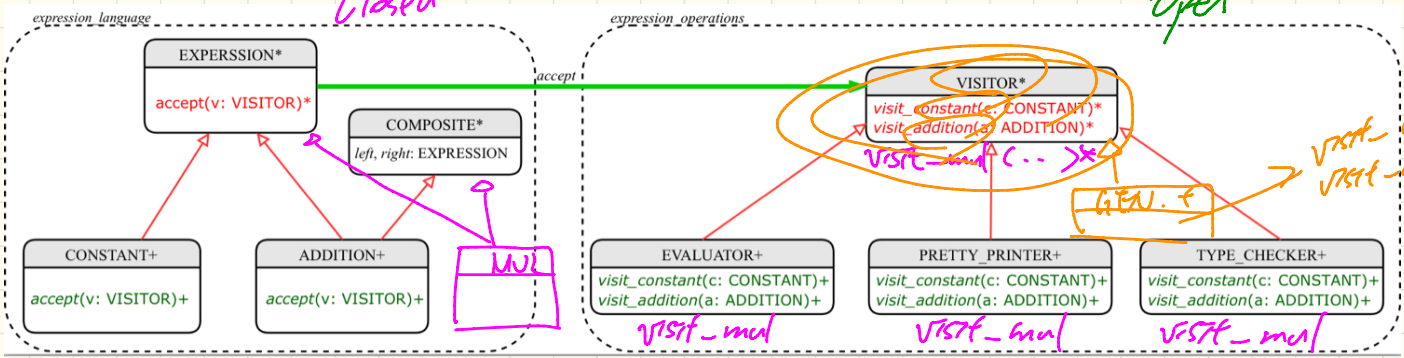
Visualizing Line 4 to Line 7

1 + 2

Executing Composite and Visitor Patterns at Runtime (double dispatch)



Visitor Pattern: Open-Closed and Single Choice Principles



Adding a new language construct? → violates SCP ⇒ this part should be closed

Adding a new language operation? → satisfied SCP ⇒ this part can be open

GEN_ASSEMBLY

Void Safe in Java? (1)

```
1  class Point {
2    double x;
3    double y;
4    Point(double x, double y) {
5      this.x = x;
6      this.y = y;
7  }
```

```
1  class PointCollector {
2    ArrayList<Point> points;
3    PointCollector() { }
4    void addPoint(Point p) {
5  points.add(p); }
6    Point getPointAt(int i) {
7      return points.get(i); } }
```

The above Java code ***compiles***. But anything wrong?

```
1  @Test
2  public void test1() {
3    PointCollector pc = new PointCollector();
4    pc.addPoint(new Point(3, 4));
5    Point p = pc.getPointAt(0);
6    assertTrue(p.x == 3 && p.y == 4); }
```

pc.points null

Void Safe in Java? (2)

```
1  class Point {
2    double x;
3    double y;
4    Point(double x, double y) {
5      this.x = x;
6      this.y = y;
7  }
```

```
1  class PointCollector {
2    ArrayList<Point> points;
3    PointCollector() {
4      points = new ArrayList<>(); }
5    void addPoint(Point p) {
6      points.add(p); }
7    Point getPointAt(int i) {
8      return points.get(i); } }
```

```
1  @Test
2  public void test2() {
3    PointCollector pc = new PointCollector();
4    Point p = null;
5    pc.addPoint(p);
6    p = pc.getPointAt(0);
7    assertTrue(p.x == 3 && p.y == 4); }
```

null

null