Thursday Sept. 21

Lecture 5

Why OO

template

Observe

p1 . (x1, y1)
2 3

p2 . (x2, y2)
2 -3

Model

```
class {
    Point
    int x;
    int y;
}
```

Execute

p1 → Point: x | 2, y | 3

p2 → Point: x | 2, y | -3

p1 p2

context object

p1.getY()

method

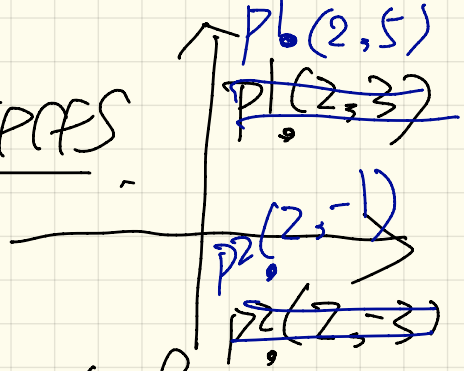
dot notation

p1.moveUp(2)

p2.getY()

p2.moveUp(2)

point objects



3

calling the same method on different context objects may give you different results.

-3

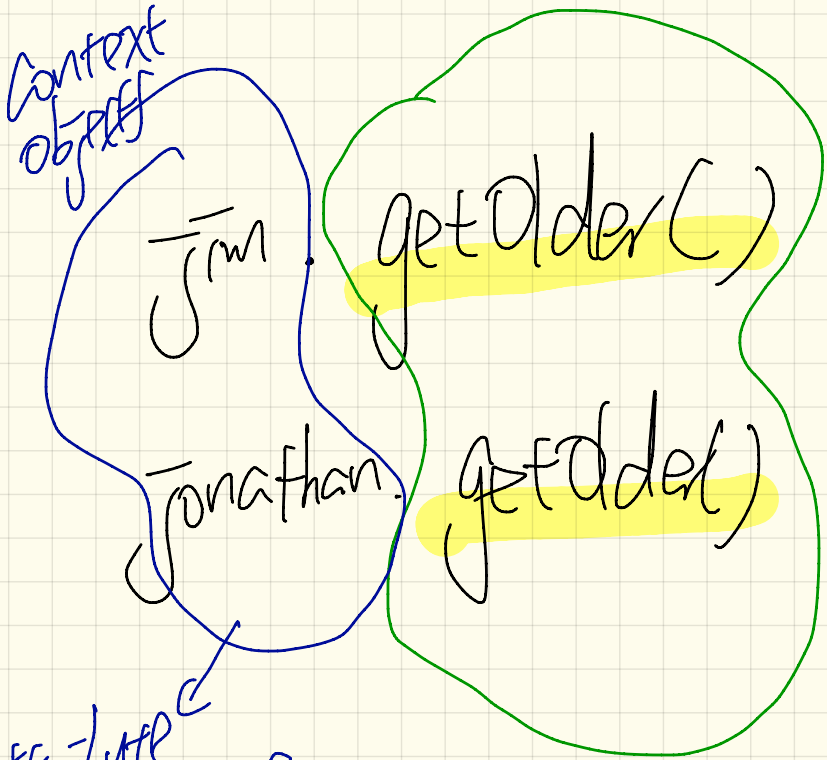
Given some natural language requirements,

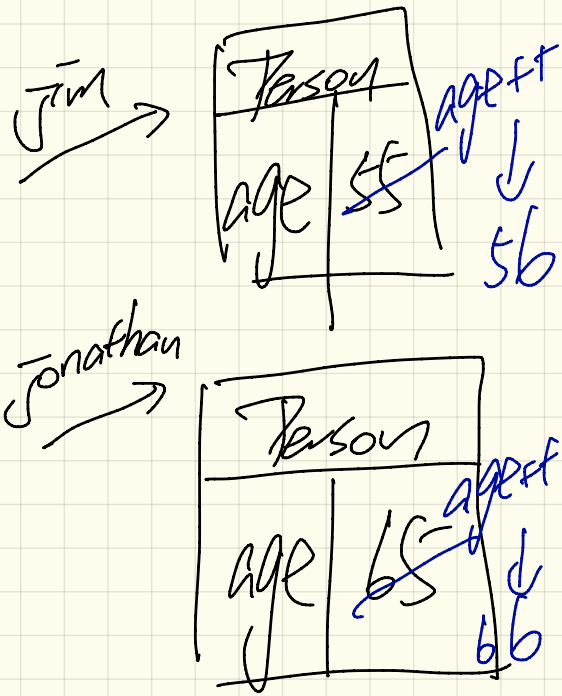
we can:

1. Identify nouns as either classes or attributes

2. Identify verbs as methods.

- change attributes
- ask something about attributes



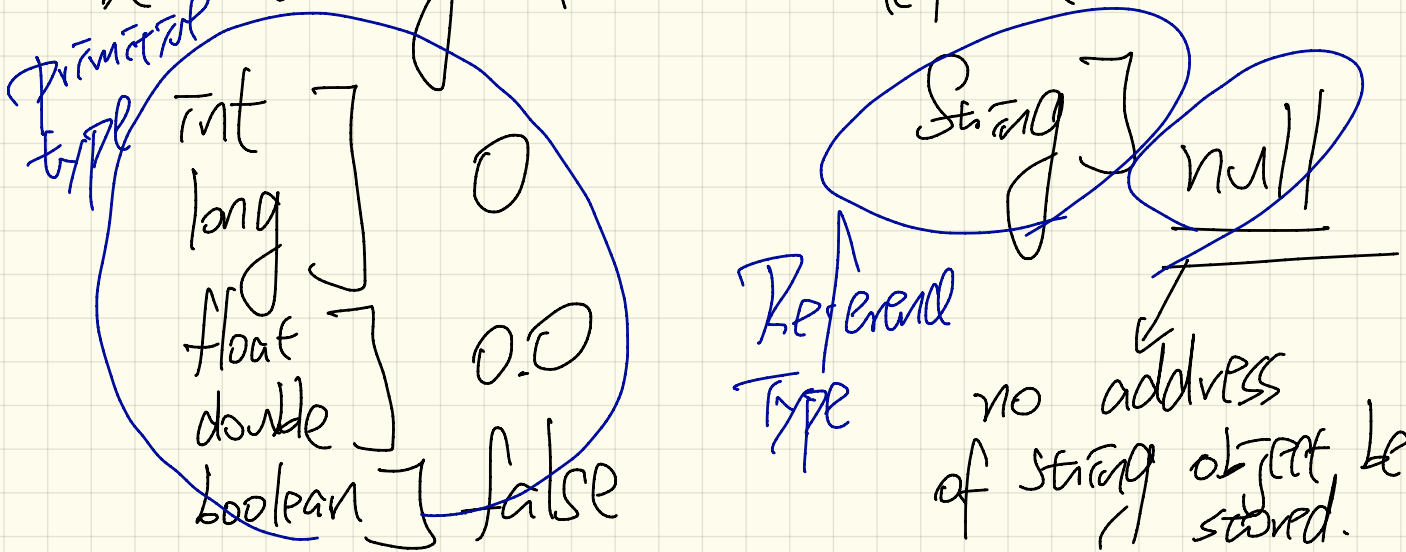


These two method calls share the same definition:

```
age++;
```

default values

When uninitialized, variables can be assigned their default values



```
class Point {
    double x;
    double y;
    Point() {
        // default values
    }
}
```

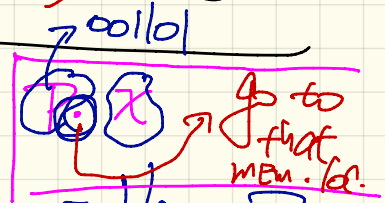
③ Store 00110l into P.

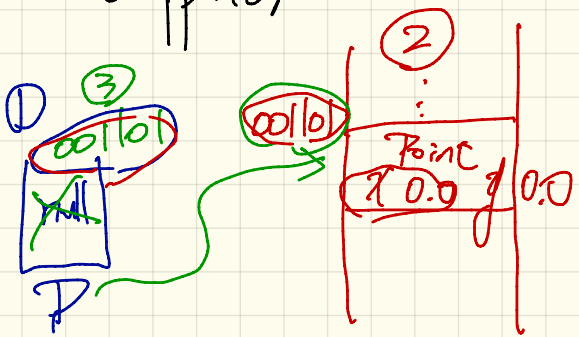
Supplier

```
class PointApp {
    ——— main(--) {
        Point p = new Point();
    }
}
```

① ③ ②

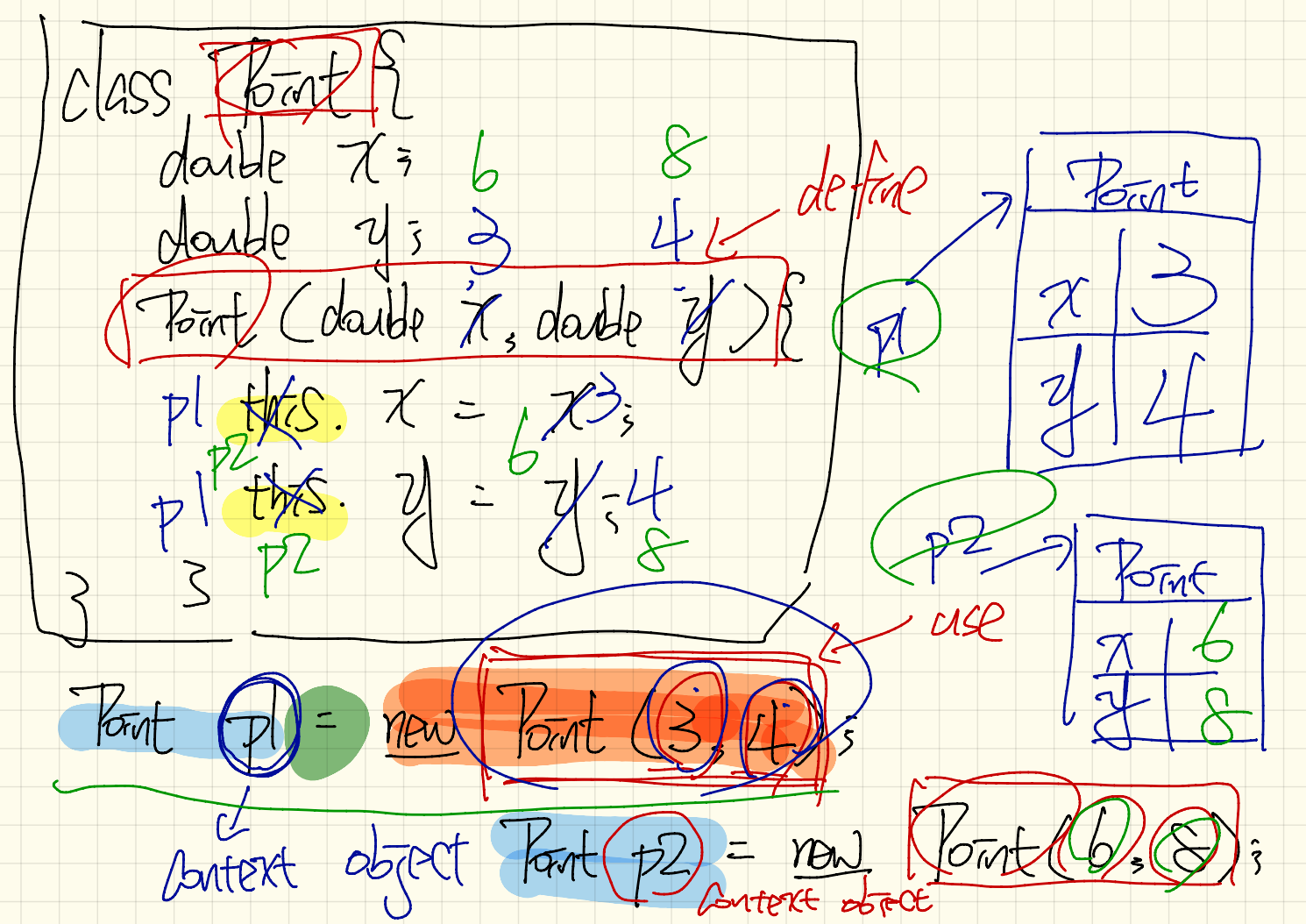
User/Client





① We declare a variable p. p can only store addresses of a memory portion that store Point information. Point info.

② Allocate a memory portion for storing


class Point {
double x;
double y;
Point (double x, double y) {
p1 this. x = x;
p2 this. y = y;
}
define
p1
Point
x 3
y 4
p2
Point
x 6
y 8
use
Point p1 = new Point (3, 4);
Point p2 = new Point (6, 8);
context object
context object

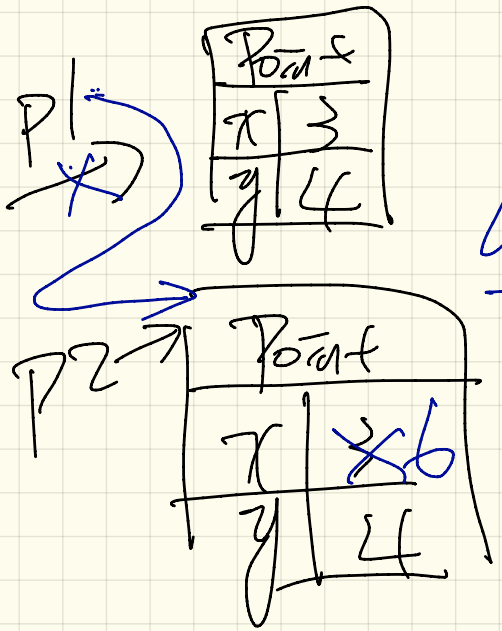
Point p1 = new . . . .

Point p2 = new . . .

p1 = p2 ;

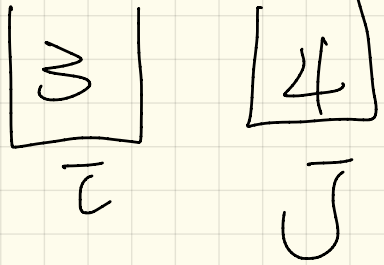
p1.x = 6

p2.x ?



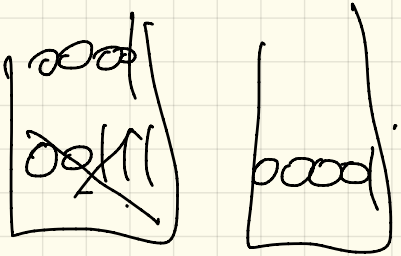
aliasing

int i = 3;
int j = 4;



i == j F

Point p1 = new Point(3,4);
Point p2 = new Point(3,4);



p1 = p2, p1 == p2