Lecture 2: Objects & Values

Programming language describes computations using values

primitive values | constructed values (objects)

Ex. 1 2.0 true null 'x'

OO language

Values have run-time type (or run-time class, for objects)

2 is an int. 2.0 is a double (precision floating-point number) ~17 decimal places

2.0f is a float (~7 decimal places)

'x' is a char (character) - represents one of 65,000 Unicode characters.

true is a boolean

2+3 is a computation that computes value 5.

↑ (binary) operator

↑ 2 operands

!true computes false

↑ unary operator

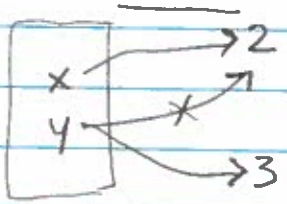
2+true : runtime type error (not possible in Java: strongly typed)

Variables Languages let us assign values to variables.

```
x = 2;
y = 2;
y = y+1;
```

↑ "gets" ≠ "equal"

view 1



view 2

x = 2

y = ~~2~~ 3

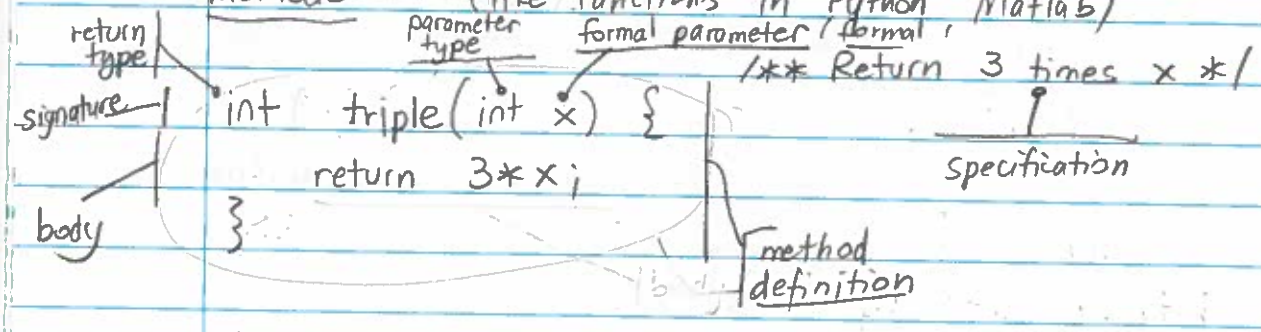
(closer to how it's implemented)

Types

Variables must be declared with their type. (Unlike Python)

```
int x;          (declaration)
x = 2;          (assignment)
int y = x+1;    (declaration + initialization)
y = true;       (static (compile-time) type error)
double z = y;   (ok: int is promoted to double)
```

Methods (like functions in Python, Matlab)



Use:

```
int y = 1000;
int z = triple(y+1);
```

y → 1000
z → 3003
x → 1001

Objects

- created at run time
- instances of classes
- contain instance variables (fields)
- and methods

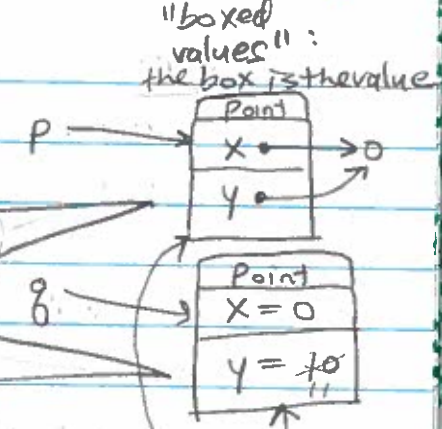
```
class Point {
    int x = 0;
    int y = 0;
}
```

```
Point p = new Point();
Point q = new Point();
```

"new" makes a new object

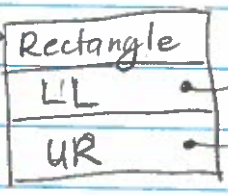
different objects, same state

"boxed values": the box is the value




```
q.y = 10        (same object, new state)
q.y = q.y + 1   (using an ivar)

class Rectangle {
    Point LL;
    Point UR;
}
Rectangle r = new Rectangle();
r.UL = p;
r.LR = q;
```



Instance Methods

Classes can define operations (methods).

```
class Rectangle {
    :
    int area() { return (UR.x - LL.x)
                        *(UR.y - LL.y); }
}
```

can use instance variables

```
q.x = 11;
r.area()
```

method invocation (evaluates to 121)

must have an object to invoke method: "receiver object"

procedure (methods): return type = void.
useful for side effects

```
class Point {
    :
    void setOrigin() { x=0; y=0; }
}
q.setOrigin();
```
changes q obj.

Constructors

Methods for initializing object to a "good" state.

```
class Point {
    Point(int xx, int yy) {
        x = xx;
        y = yy;
    }
}
```
no return: value is new object ("this")

```
Point p2 = new Point(3,5)
```

p2 → Point { x=3, y=5 }

Static methods

- Attached to class
- Can't use ivars

array of arguments

```
(public) class Program {
    public static void main(String[] args) {
        System.out.println("Hello World");
    }
}
```

Program.main(new String[0])
prints Hello World

what happens if run "java Program"

Recitat

Arrays

Arrays

int

new

Point[]
pa = new
pa. leng

initiali

Can use

Point[]
for (i

po

Aliasing

Point [
pa