Classes & Objects

Today:

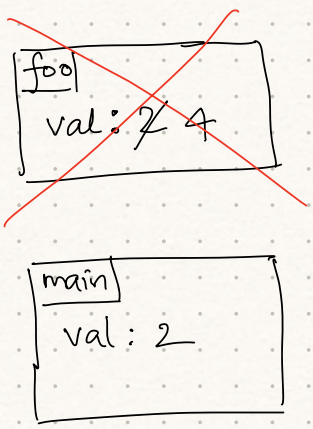
- What are Classes & Objects
- Why ?
- Java Syntax
- Constructors
- Instance Methods
- Problems.

1. What is the output of the following code?

```
public class Main {

    public static void foo(int val) {
        val = val * val;
    }

    public static void main(String[] args) {
        int val = 2;
        foo(val);
        System.out.println(val);
    }
}
```



```java
public class Main {

    public static double arrAverage(int[] arr) {
        int sum = 0;
        for(int i=0; i<arr.length; i++)
          sum += arr[i];

        return  (double) sum / arr.length;
    }

    public static void main(String[] args) {
        int[] arr = {5,3,4,1,6};
        System.out.println(arrAverage(arr));
    }
}
```

Object Oriented Programming (OOP)

Create our own (custom) Data type.

int, float, boolean, double

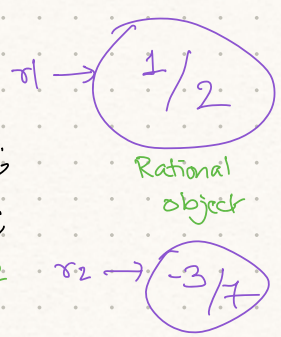
String, int[], String[] . . .

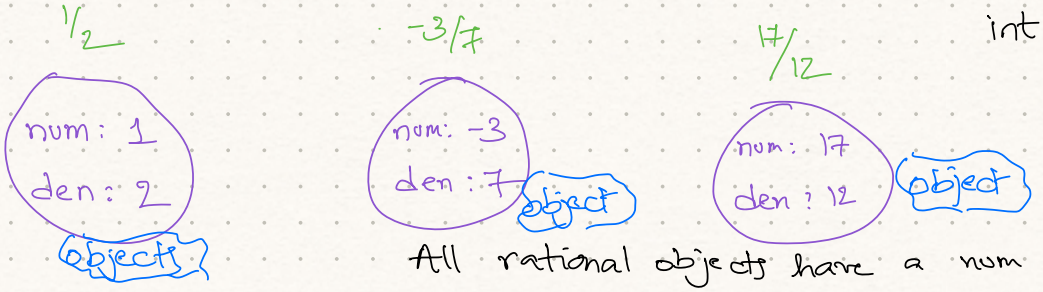
Ex 1: Rational Numbers = p/q ← numerator, ← denominator; p, q are integers, $q \neq 0$

e.g. $1/2$, $-3/7$, $17/12$, ...

We'd like to:

```
Rational r1 = new Rational(1,2);
Rational r2 = new Rational(-3,7);
S.O.P(r1.add(r2)); // Add rational r2 to rational r1
```



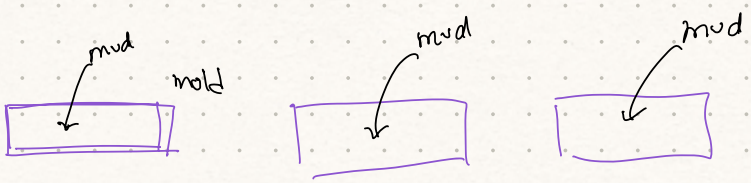


int a = -10;

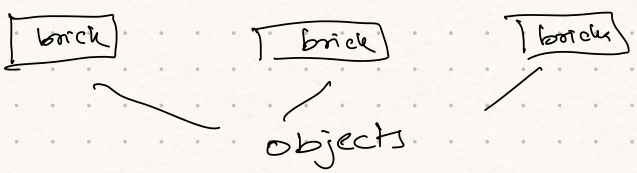
All rational objects have a num & a den.

```
class Rational {
    int num;
    int den;
}
```

Instance Variables (fields) — what is common to all objects of that type



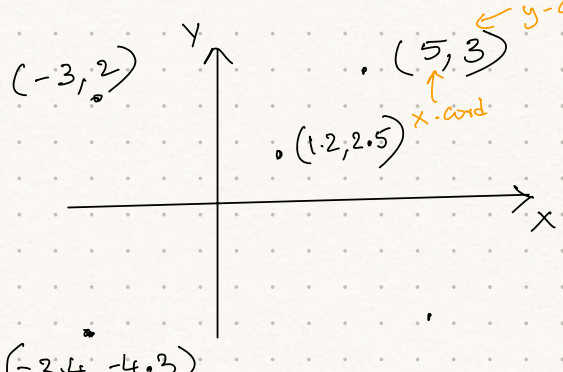
: class

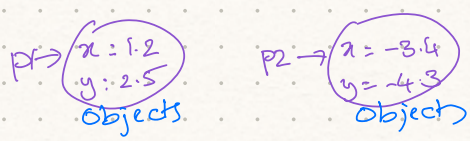


objects

classes are like 'blueprints' of objects.
'Template'

Ex 2: Points in 2D plane.






```
Point p1 = new Point(1.2, 2.5);
Point p2 = new Point(-3.4, -4.3);
S.o.p(p1.distanceTo(p2));
```

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

instance variables

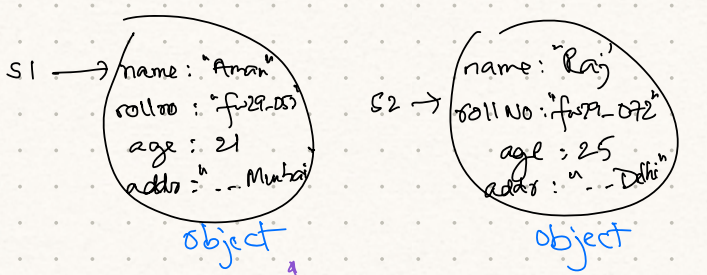
Ex 3:

```
class Student {
    String name;
    String rollno;
    int age;
    String address;
}
```

instance variables

once

```
Student s1 = new Student("Aman", "fw29_053", 21, "... Mumbai");
Student s2 = new Student("Raj", "fw29_072", 25, "... Delhi");
```



Structure of a class Definition

```
class <class_name> {
```

- instance variables ✓
- Constructors ✓
- Instance Methods ✓
- static main (optional)

```
}
```

Constructors: special method that creates an object (instantiate) & returns a reference to the object.

```
class Point {
    double x;
    double y;
    public Point(double x, double y) {
        this.x = x;   // x = 2.1
        this.y = y;   // y = 3.5
    }
    public Point() {
        this.x = 1;
        this.y = 1;
    }
```

```
Point p1 = new Point(2.1, 3.5);
int x = 10;
```

p1 → x: 2.1, y: 3.5

```
Point p2 = new Point();
```

p2 → x: 1, y: 1

this : reserved keyword.
refers to the current object

Instance Methods

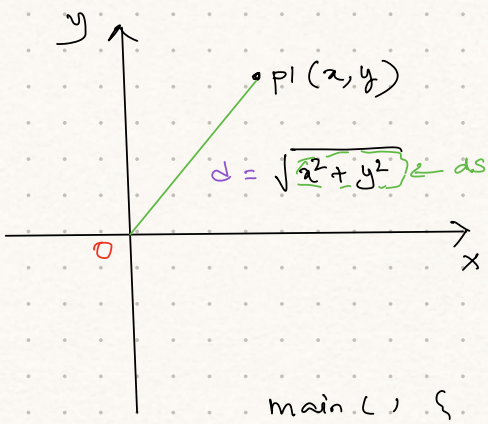
(very similar to functions)

Method ≡ what are all the (actions) operation I can perform on an object ?

Func. Definition .... Method Definition

Func. Call .... Method Calling

(Static) — No static keyword



p1 → x = 3.0, y = 4.0

```
class Point {
    double x;
    double y;
    Constructn

    // Method Definition
    public void distanceFromOrigin(){
        double dsq = x*x + y*y;
        S.o.P(Math.sqrt(dsq));
    }
}
```

```
main() {
    Point p1 = new Point(3.0, 4.0);
    p1.distanceFromOrigin();
    Point p2 = new Point(5, 6);
    p2.distanceFromOrigin();
}
```

Method call ←

p1.distanceFromOrigin() — object | method name | (arguments) values to parameters

```
public void distanceFromOrigin() {
    double dsq = (this.x) * (this.x) +
                 (this.y) * (this.y);
    S.o.P(Math.sqrt(dsq));
}
// this refers to the object that the method was called on
```

p1 → x: 3.0, y: 4.0

p2 → x: 5.0, y: 6.0

```
main( )                {
  Point p1 =  new Point(3.0,4
                                  .0)

}
```

Method call

```
public double distanceFromOrigin() {
   double dsq = (this.x) * (this.x) +
                (this.y) * (this.y);
   return (Math.sqrt (dsq));
}
```

Method Defn