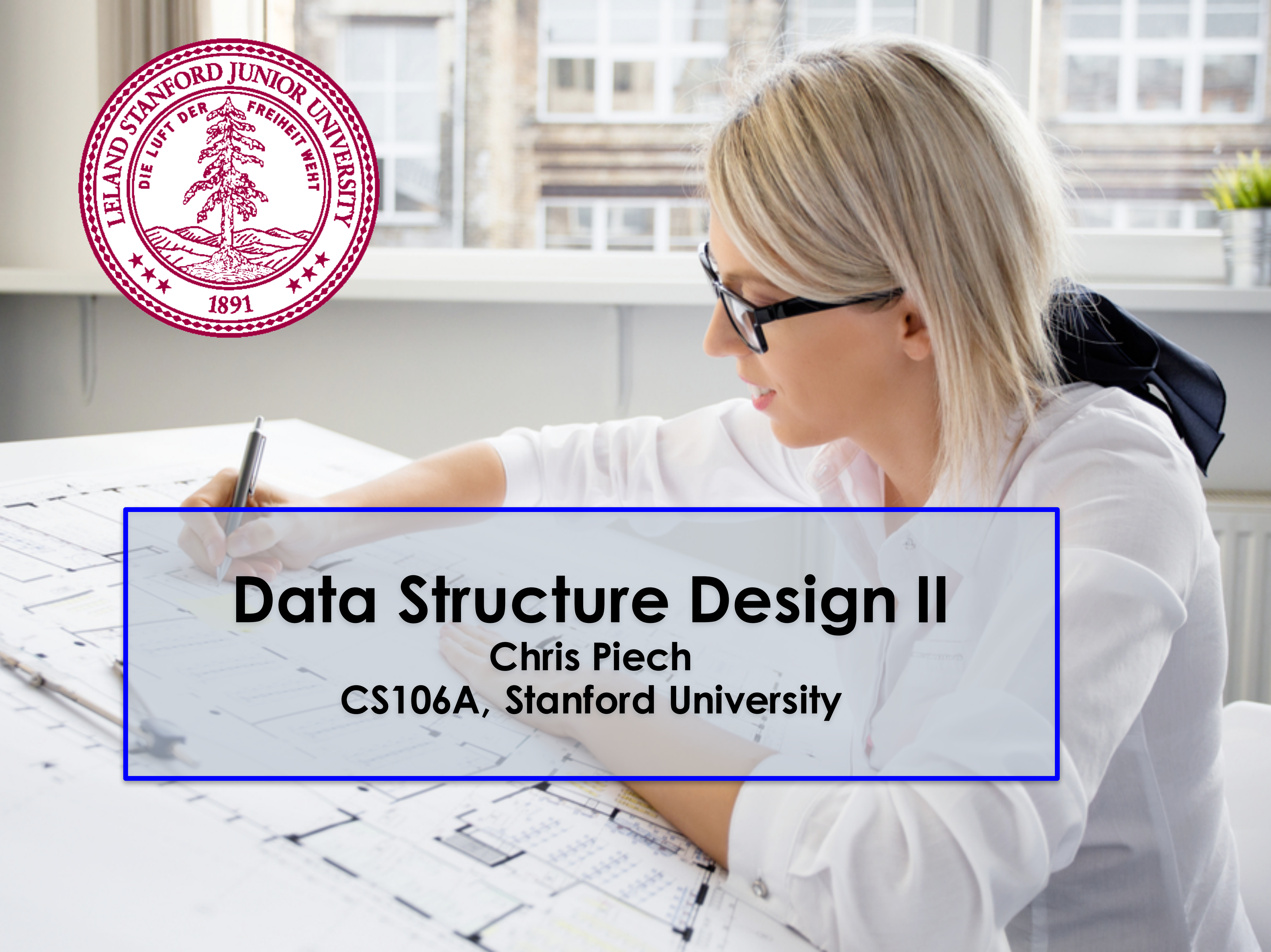
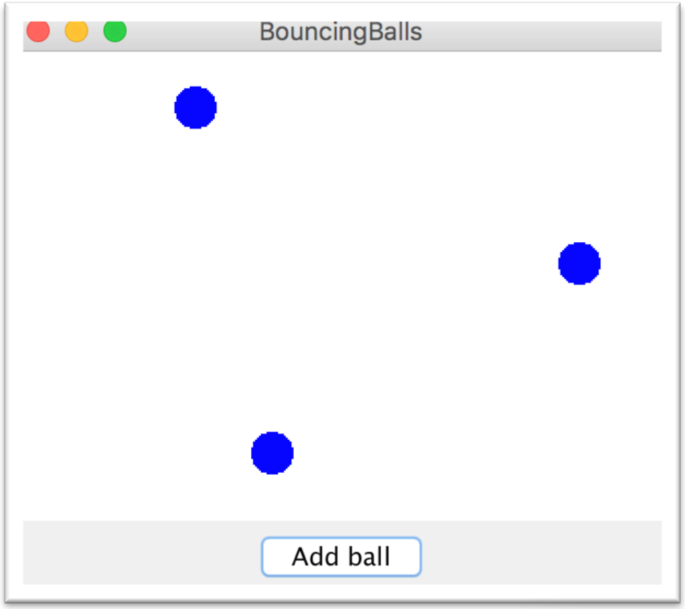


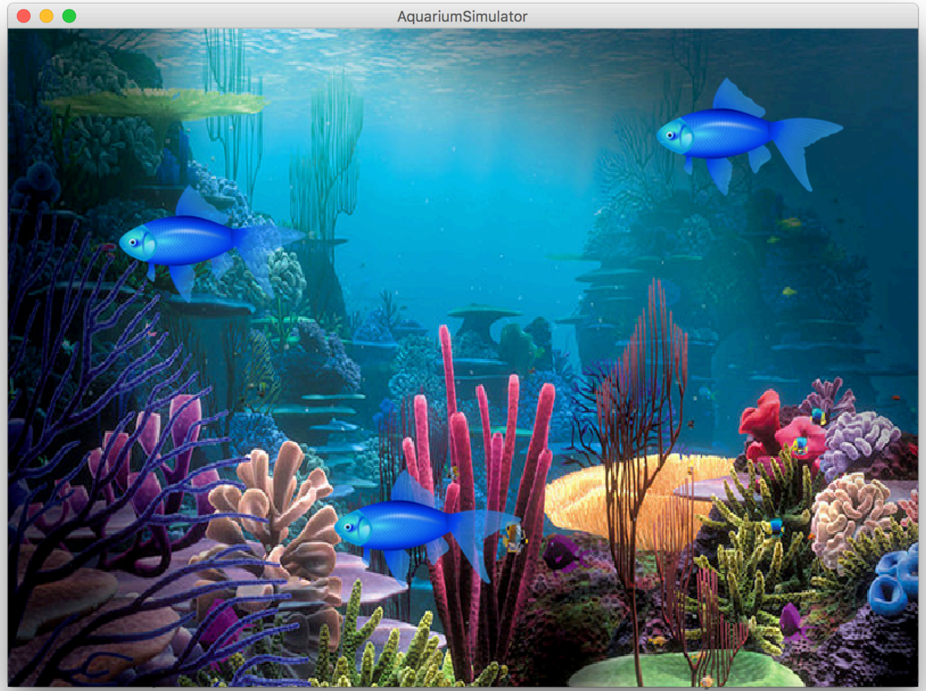
Data Structure Design II

Chris Piech

CS106A, Stanford University

Today in lecture





We have *used* many variable types

E.g. `GRect`

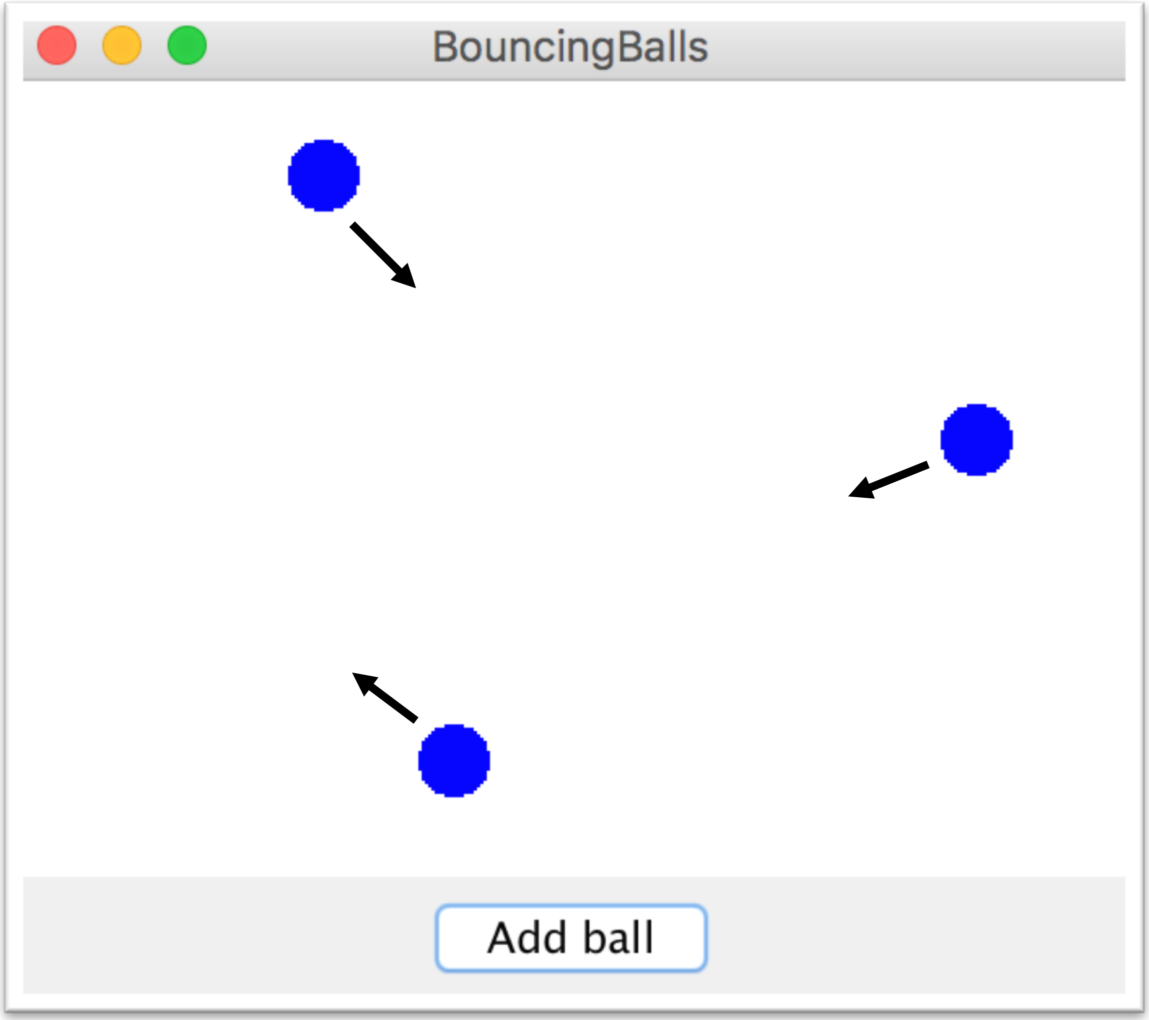
E.g. `String`

E.g. `AudioSample`

Today we learn how to define our own

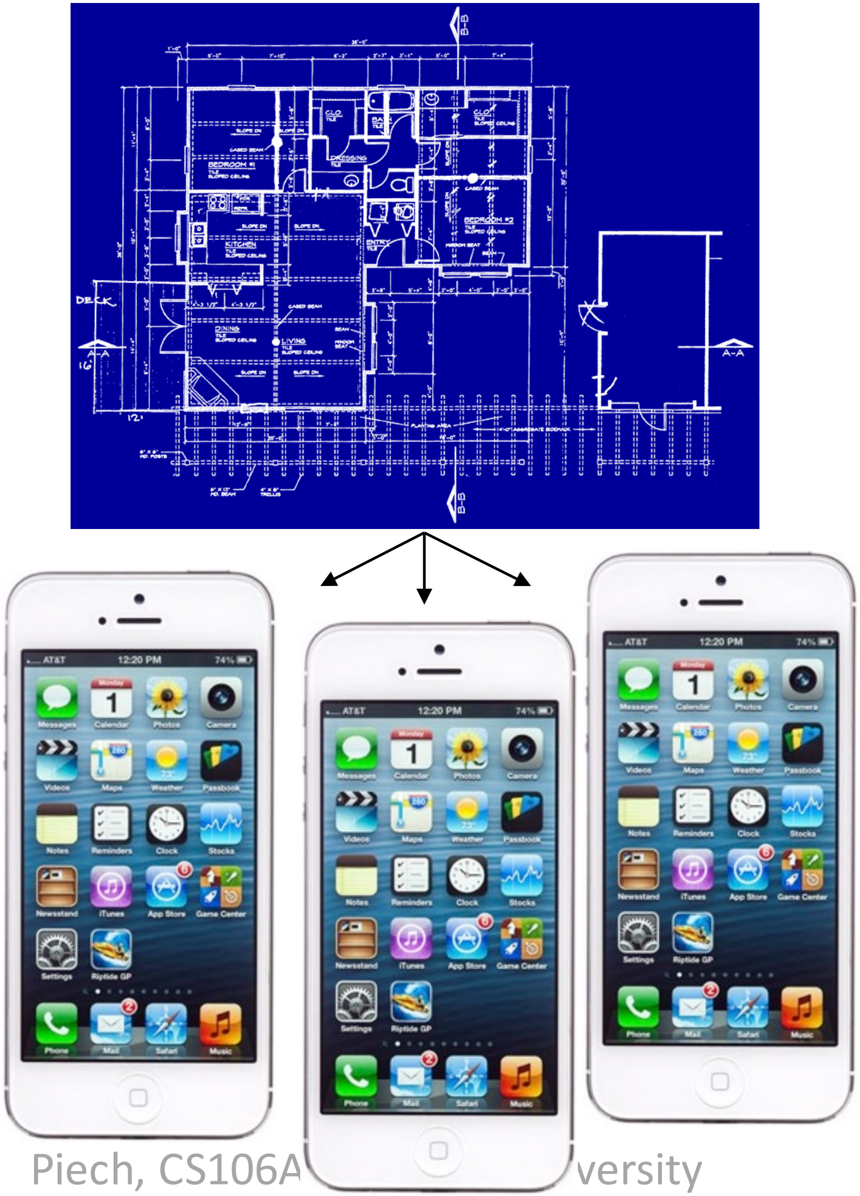
We use new Classes (written in new files) to define new variable types

Bouncing Balls



Classes are like blueprints

class: A template for a new type of variable.



You must define three things

1. What **variables** does each instance store?

2. What **methods** can you call on an instance?

3. What happens when you make a **`new`** one?

*details on how to define these three things coming soon

A Ball Variable Type

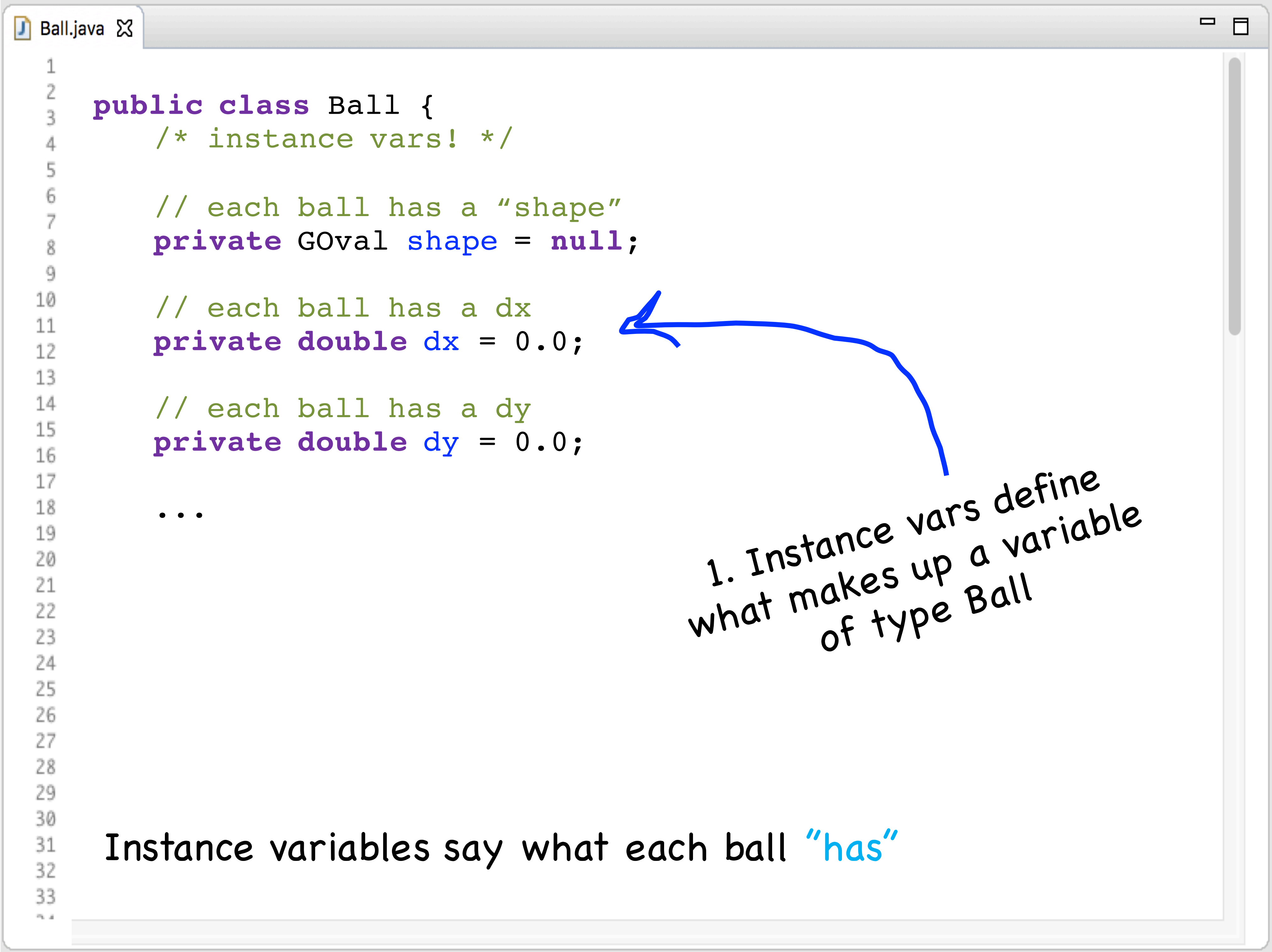
The Ball class

1. What **variables** does each instance store?
   - Each ball has its own Goval (lets call it shape)
   - Each ball has its own dx
   - Each ball has its own dy

2. What **methods** can you call on an instance?
   - `heartbeat();`
   - `getShape();`

3. What happens when you make a **`new`** one?
   - Sets initial values for all the ”instance” vars

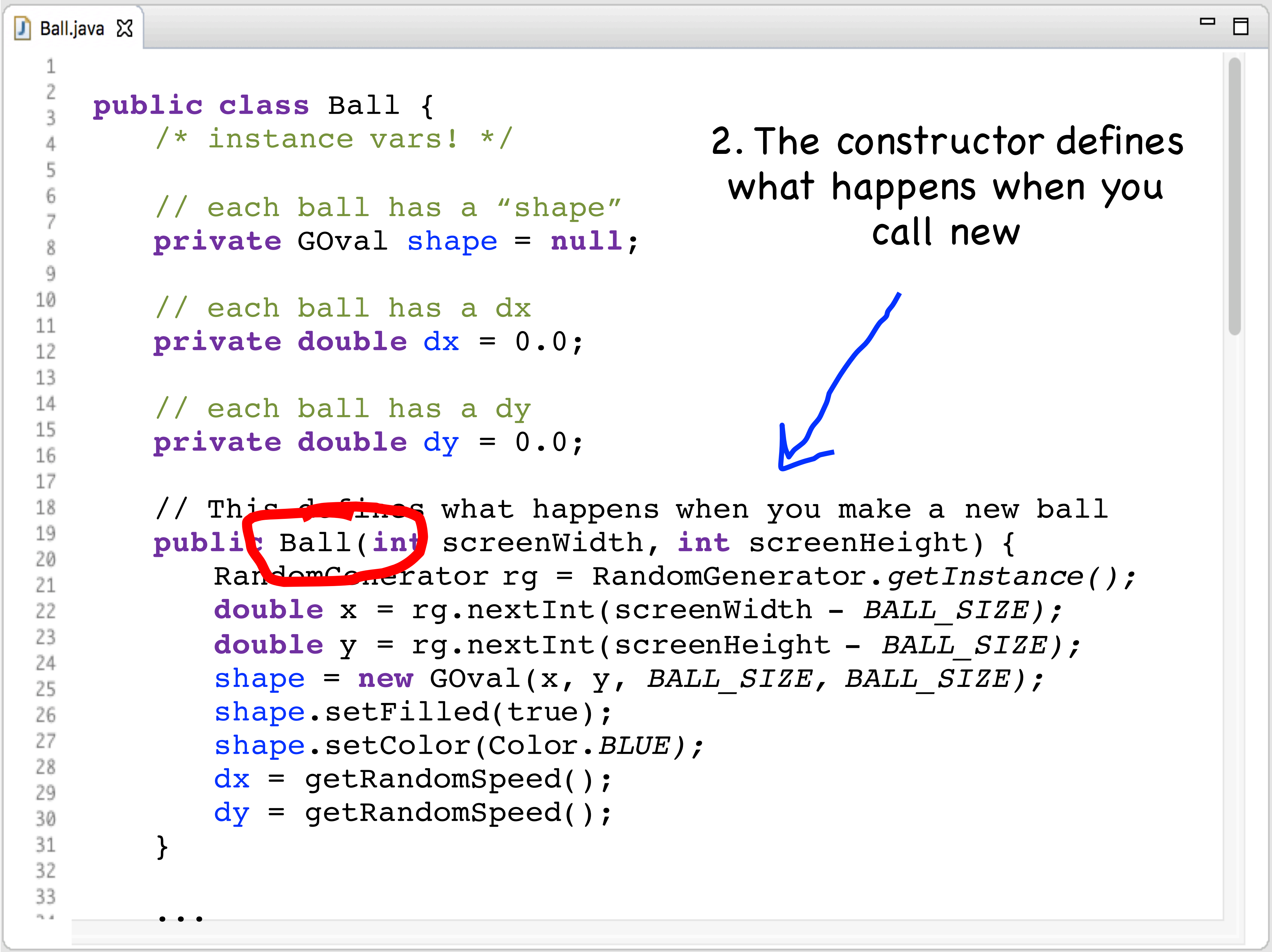
*details on how to define these three things coming soon



```java

public class Ball {
    /* instance vars! */

    // each ball has a “shape”
    private GOval shape = null;

    // each ball has a dx
    private double dx = 0.0;

    // each ball has a dy
    private double dy = 0.0;

    ...
```

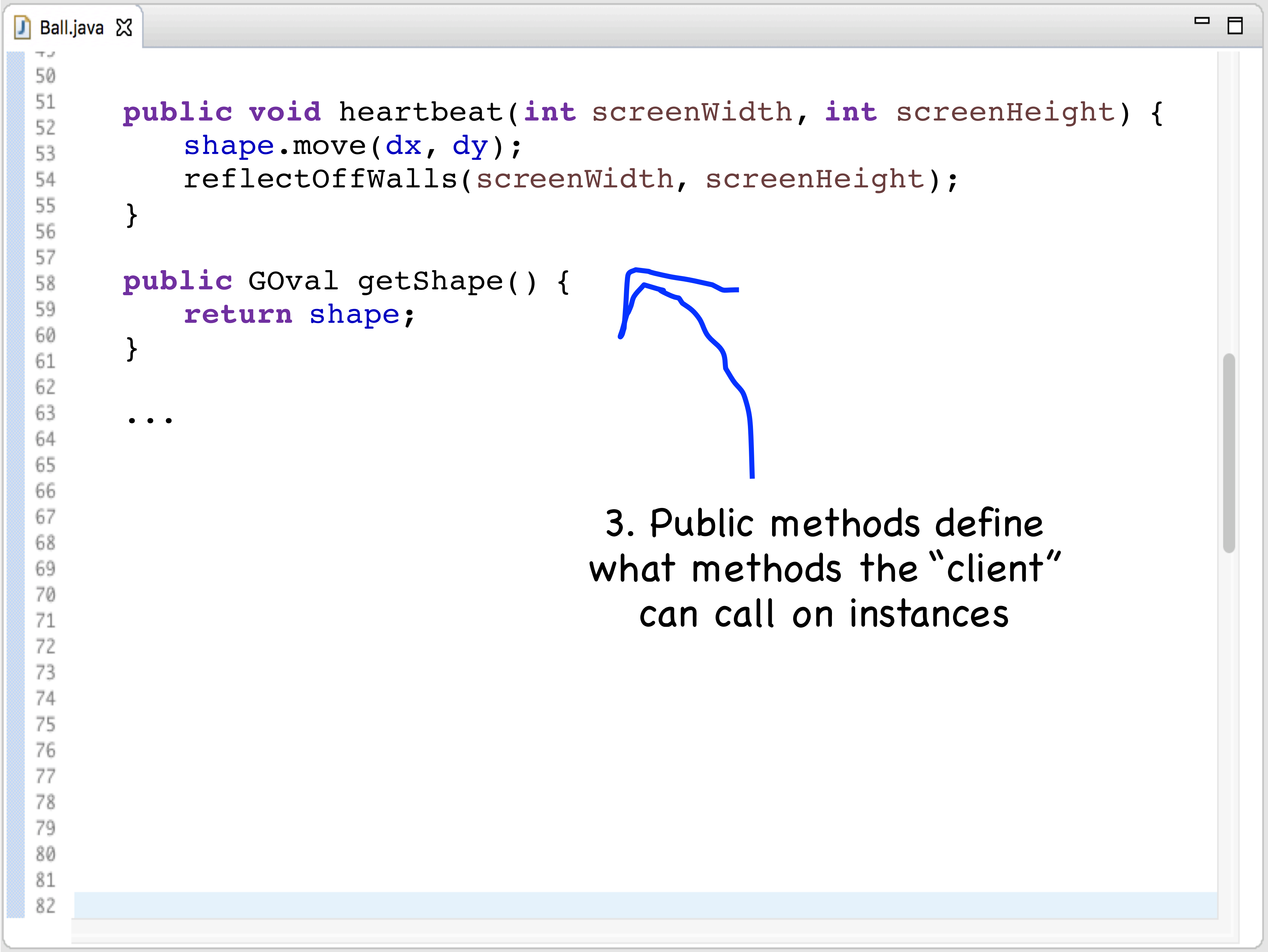
1. Instance vars define what makes up a variable of type Ball

Instance variables say what each ball “has”



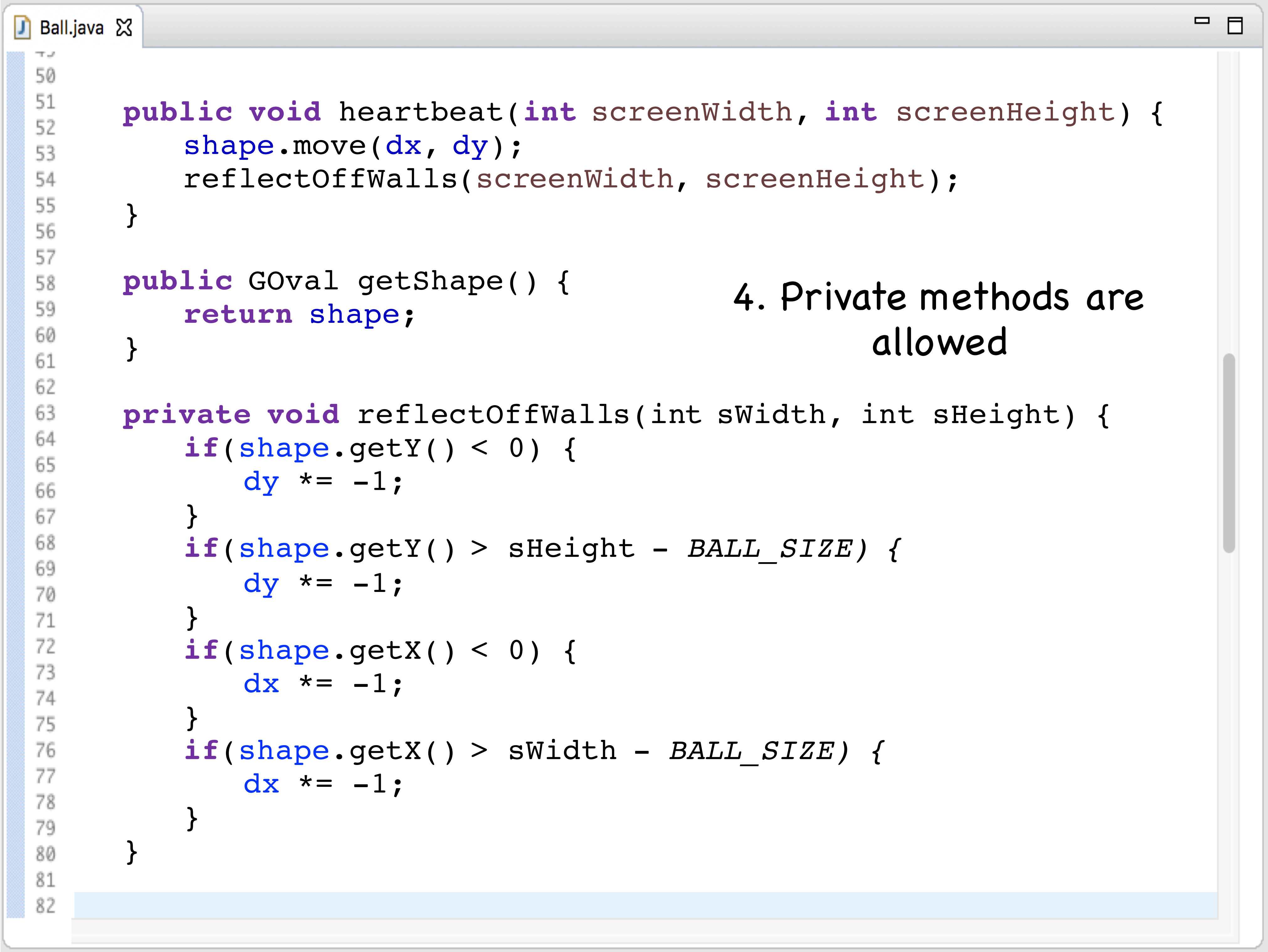
```java

public class Ball {
    /* instance vars! */

    // each ball has a “shape”
    private GOval shape = null;

    // each ball has a dx
    private double dx = 0.0;

    // each ball has a dy
    private double dy = 0.0;

    // This defines what happens when you make a new ball
    public Ball(int screenWidth, int screenHeight) {
        RandomGenerator rg = RandomGenerator.getInstance();
        double x = rg.nextInt(screenWidth - BALL_SIZE);
        double y = rg.nextInt(screenHeight - BALL_SIZE);
        shape = new GOval(x, y, BALL_SIZE, BALL_SIZE);
        shape.setFilled(true);
        shape.setColor(Color.BLUE);
        dx = getRandomSpeed();
        dy = getRandomSpeed();
    }

    ...
```

2. The constructor defines what happens when you call new



```
    public void heartbeat(int screenWidth, int screenHeight) {
        shape.move(dx, dy);
        reflectOffWalls(screenWidth, screenHeight);
    }

    public GOval getShape() {
        return shape;
    }

    ...
```

3. Public methods define what methods the "client" can call on instances



```
    public void heartbeat(int screenWidth, int screenHeight) {
        shape.move(dx, dy);
        reflectOffWalls(screenWidth, screenHeight);
    }

    public GOval getShape() {
        return shape;
    }

    private void reflectOffWalls(int sWidth, int sHeight) {
        if(shape.getY() < 0) {
            dy *= -1;
        }
        if(shape.getY() > sHeight - BALL_SIZE) {
            dy *= -1;
        }
        if(shape.getX() < 0) {
            dx *= -1;
        }
        if(shape.getX() > sWidth - BALL_SIZE) {
            dx *= -1;
        }
    }

```

4. Private methods are allowed

What does a class do?

A class defines a new variable type

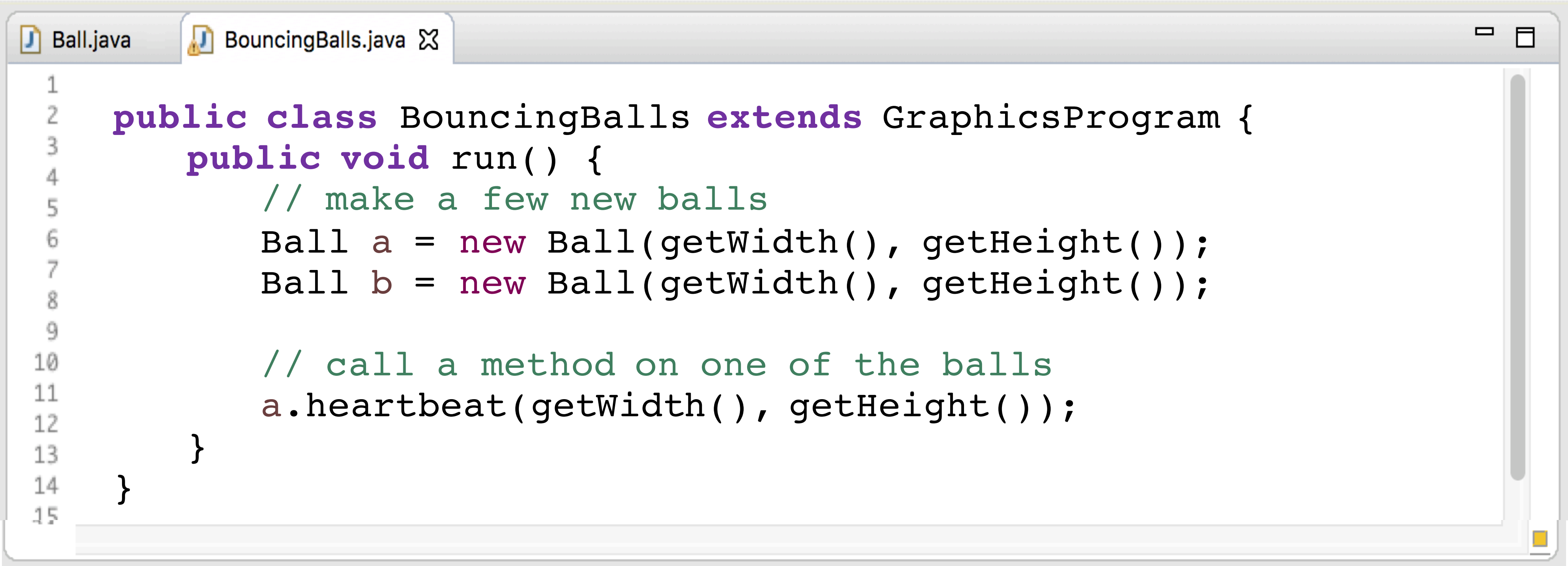
You must define three things

1. What **variables** does each instance store?

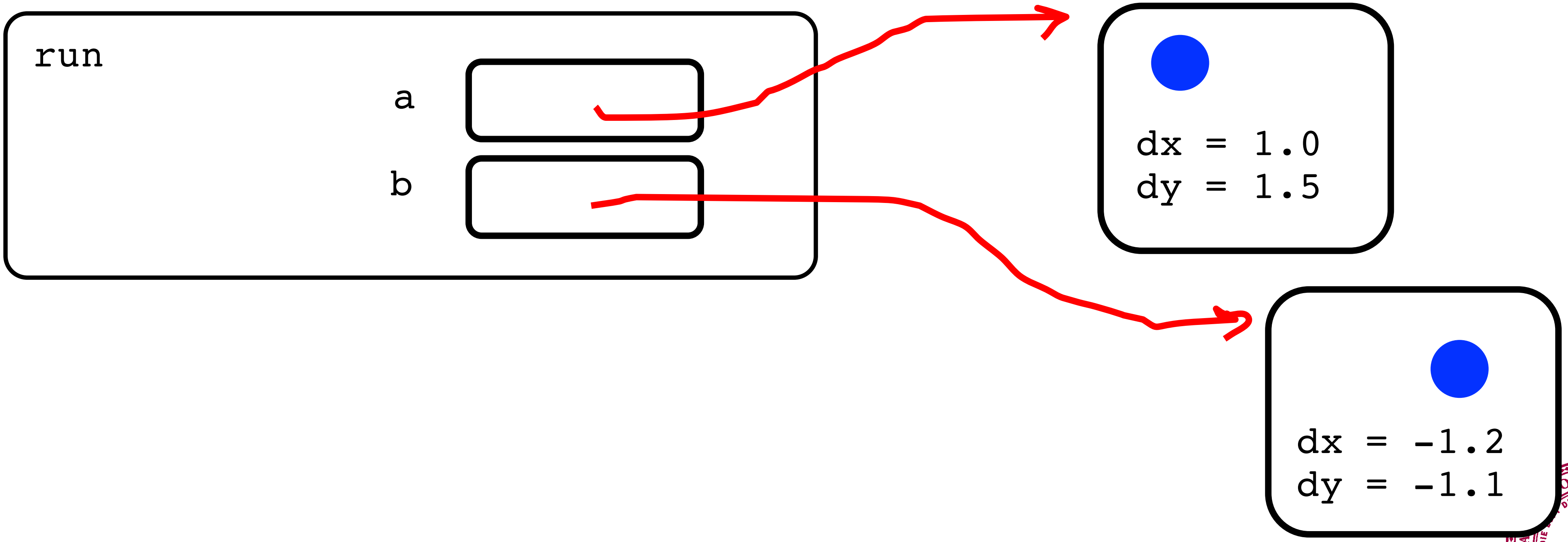
2. What **methods** can you call on an instance?

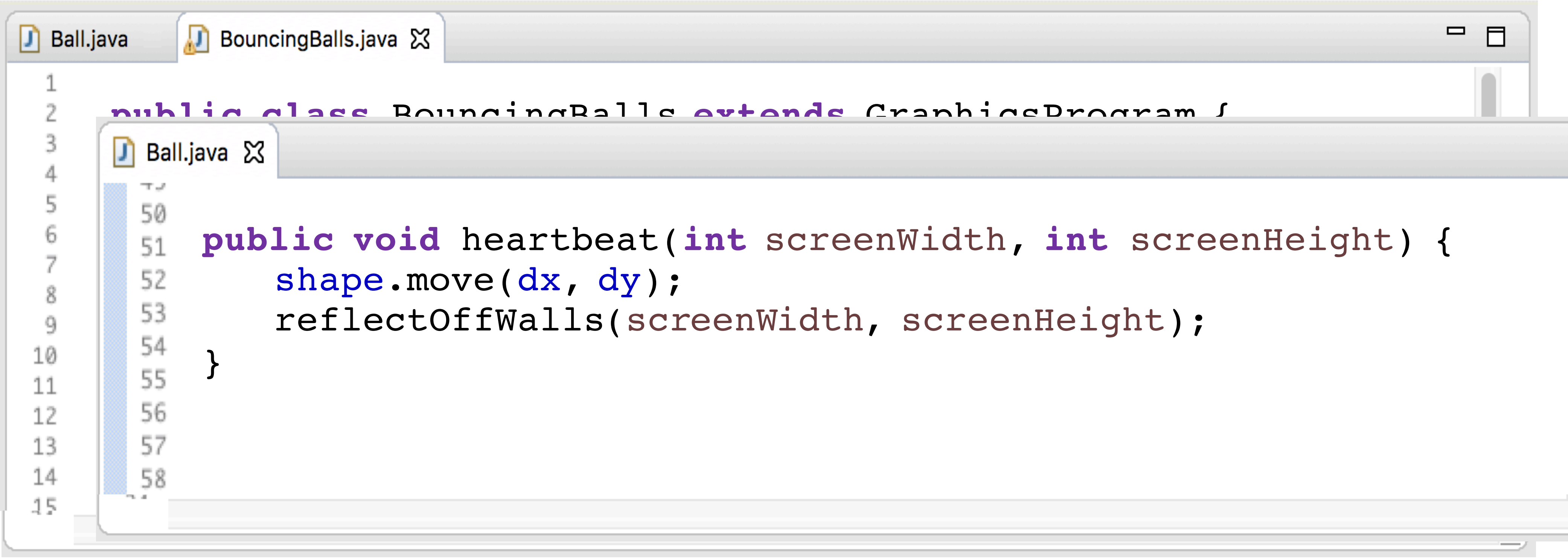
3. What happens when you make a **`new`** one?

Wait… if each ball has it's own dx and dy.
How does Java know which one to use?



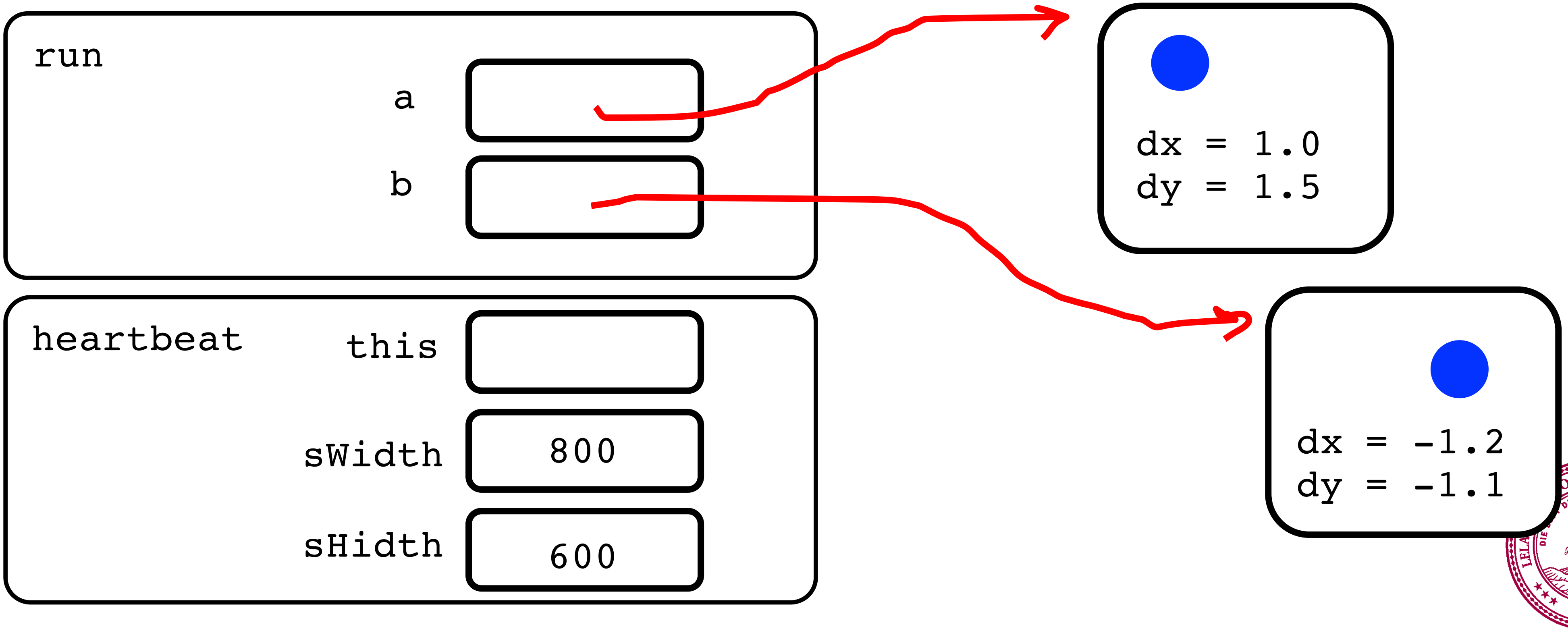
```java

public class BouncingBalls extends GraphicsProgram {
    public void run() {
        // make a few new balls
        Ball a = new Ball(getWidth(), getHeight());
        Ball b = new Ball(getWidth(), getHeight());

        // call a method on one of the balls
        a.heartbeat(getWidth(), getHeight());
    }
}
```





```
public class BouncingBalls extends GraphicsProgram {
```

```
public void heartbeat(int screenWidth, int screenHeight) {
    shape.move(dx, dy);
    reflectOffWalls(screenWidth, screenHeight);
}
```



run

a

b

dx = 1.0
dy = 1.5

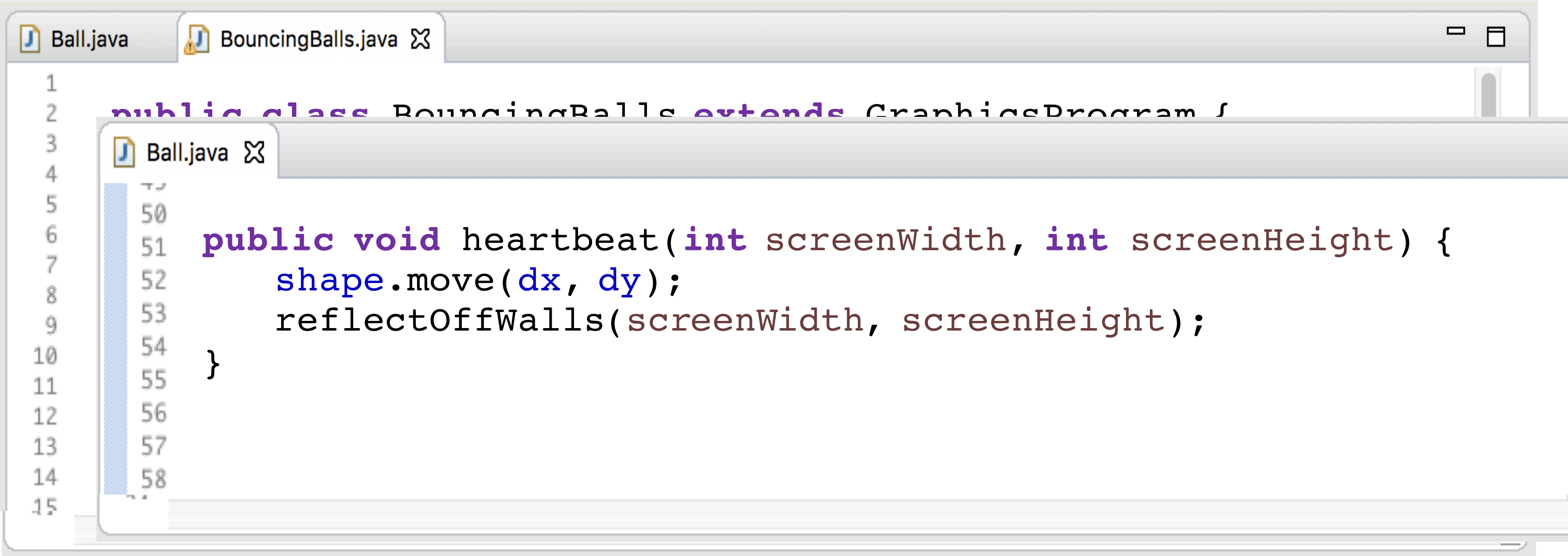
heartbeat

this

sWidth 800

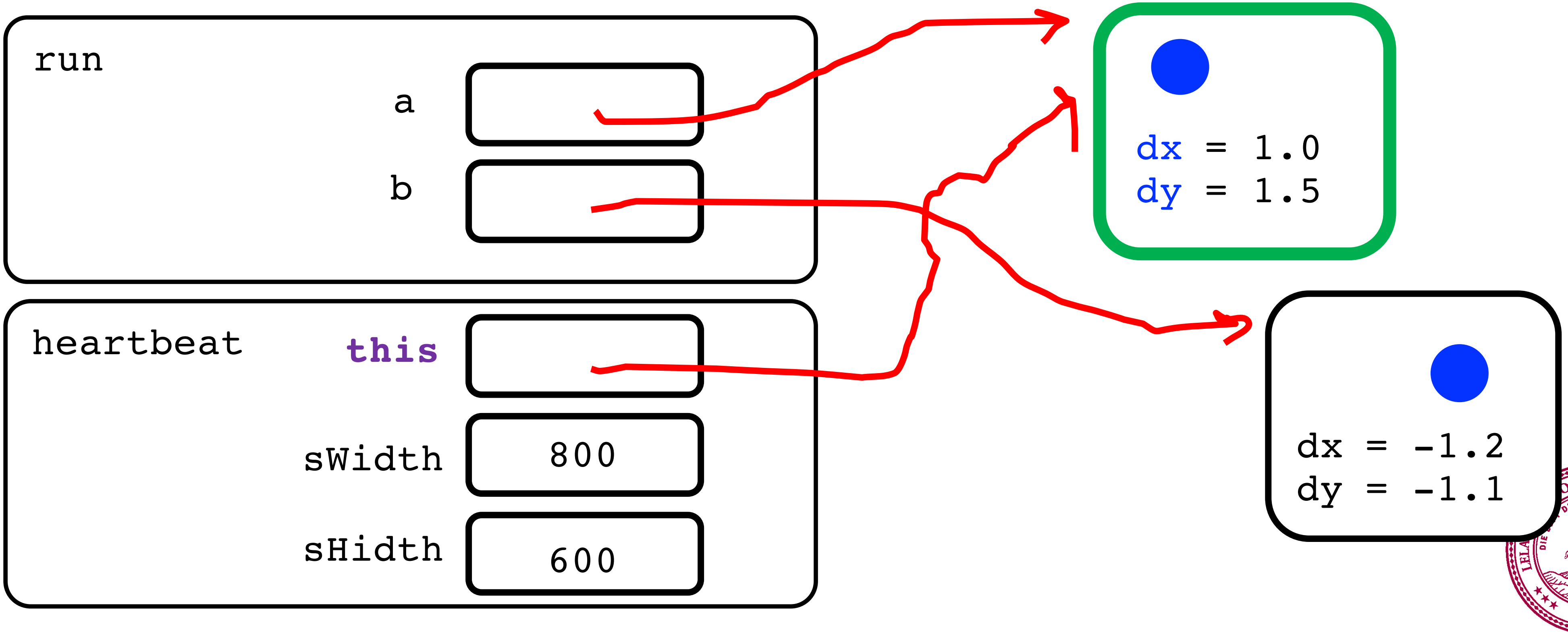
sHidth 600

dx = -1.2
dy = -1.1



```
public class BouncingBalls extends GraphicsProgram {
```

```
public void heartbeat(int screenWidth, int screenHeight) {
    shape.move(dx, dy);
    reflectOffWalls(screenWidth, screenHeight);
}
```



run

a

b

heartbeat

this

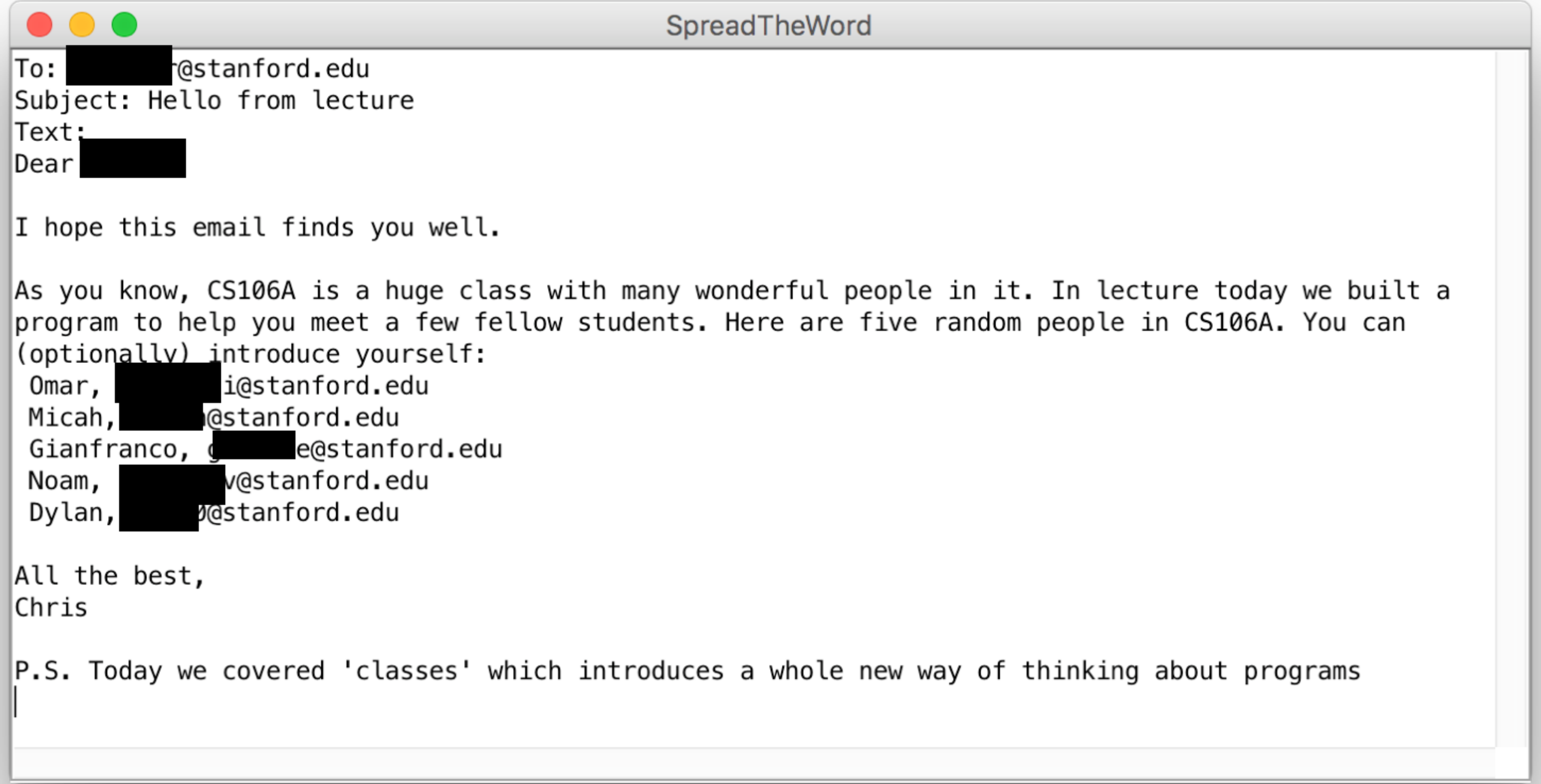
sWidth 800

sHidth 600

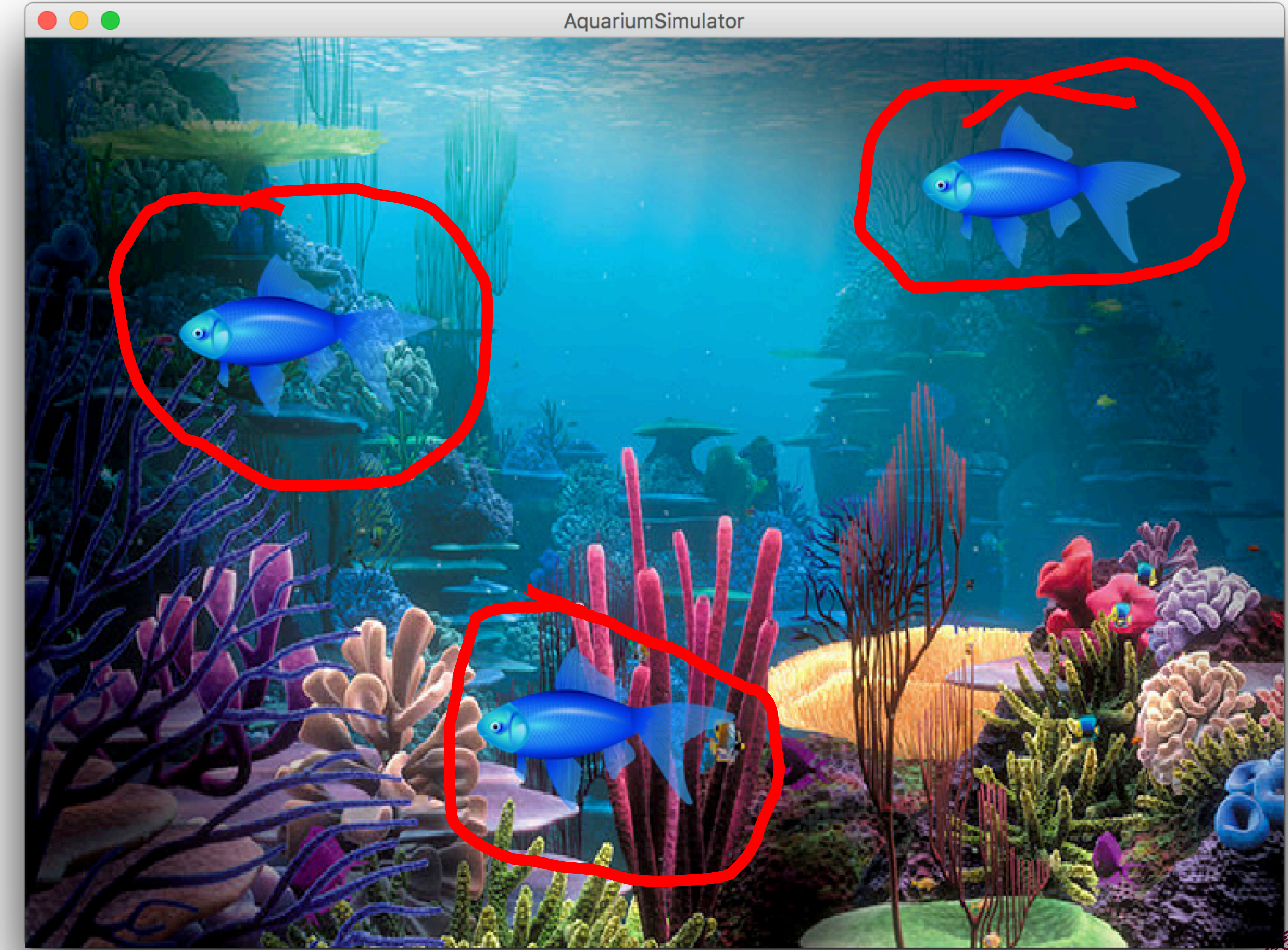
dx = 1.0
dy = 1.5

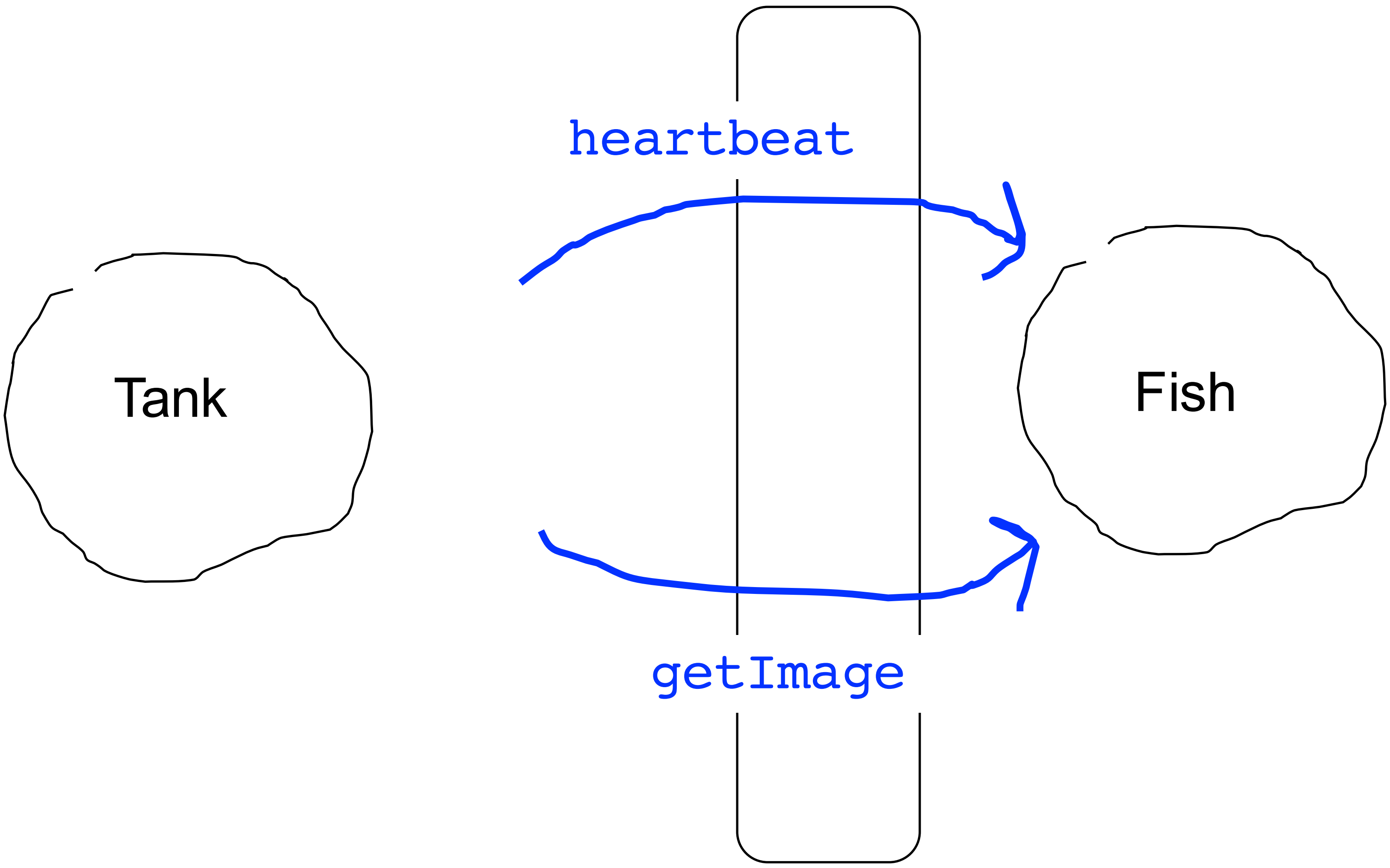
dx = -1.2
dy = -1.1

Tl;dr: Java knows which Ball you called heartbeat on



```
To: [redacted]@stanford.edu
Subject: Hello from lecture
Text:
Dear [redacted]

I hope this email finds you well.

As you know, CS106A is a huge class with many wonderful people in it. In lecture today we built a
program to help you meet a few fellow students. Here are five random people in CS106A. You can
(optionally) introduce yourself:
 Omar, [redacted]i@stanford.edu
 Micah, [redacted]@stanford.edu
 Gianfranco, [redacted]e@stanford.edu
 Noam, [redacted]v@stanford.edu
 Dylan, [redacted]@stanford.edu

All the best,
Chris

P.S. Today we covered 'classes' which introduces a whole new way of thinking about programs
```


AquariumSimulator



Tank

heartbeat

getImage

Fish

Wall of abstraction

Adding Privacy

```
private boolean isLeftImgShown;
```

- **encapsulation**: Hiding implementation details of an object from its clients.
  - Encapsulation provides *abstraction*.
    - separates external view (behavior) from internal view (state)
  - Encapsulation protects the integrity of an object's data.

- A class's instance variables should be declared *private*.
  - No code outside the class can access or change it.

What does a class do?

A class defines a new variable type

You must define three things

1. What **variables** does each instance store?

2. What **methods** can you call on an instance?

3. What happens when you make a **`new`** one?

More Practice

See Days Until

				1	2	3
4	5	6	7	8	9	10
11	12	13	14	15	16	17
18	19	20	21	22	23	24
25	26	27	28	29	30	31