Module 01

Processing Recap

CS 106 Winter 2019

Processing is…

…a language

…a library

…an environment

Variables

A *variable* is a named value. It has a *type* (which can't change) and a current value (which can change).

Variables

A *declaration* introduces a new variable, and optionally gives it an initial value.

```
int a;
float b = 6.28;
boolean c = b > 19;
```

Three declarations

Variables

Say a variable's name to read from it. Use assignment (=) to write to it.

Processing includes many built-in names.

- True *constants* can't be changed.
- Some variables are meant to be read-only.
- Some are updated automatically, and are meant to be read repeatedly.

CQ

What does this program print?

```
int a = 17;
void test( int a ){
  println( a + 5 );
}
void setup(){
  test( 10 );
}
```

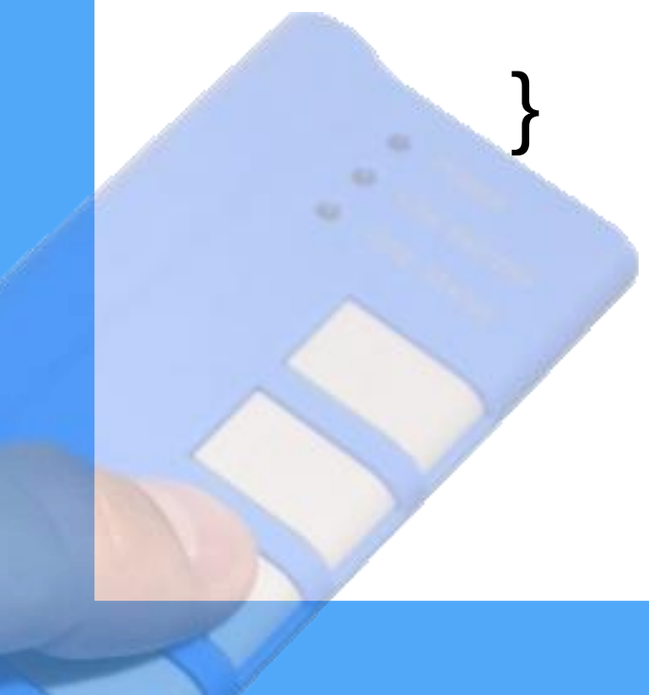
(A) 5

(B) 15

(C) 22

(D) a + 5

(E) Nothing



Scope

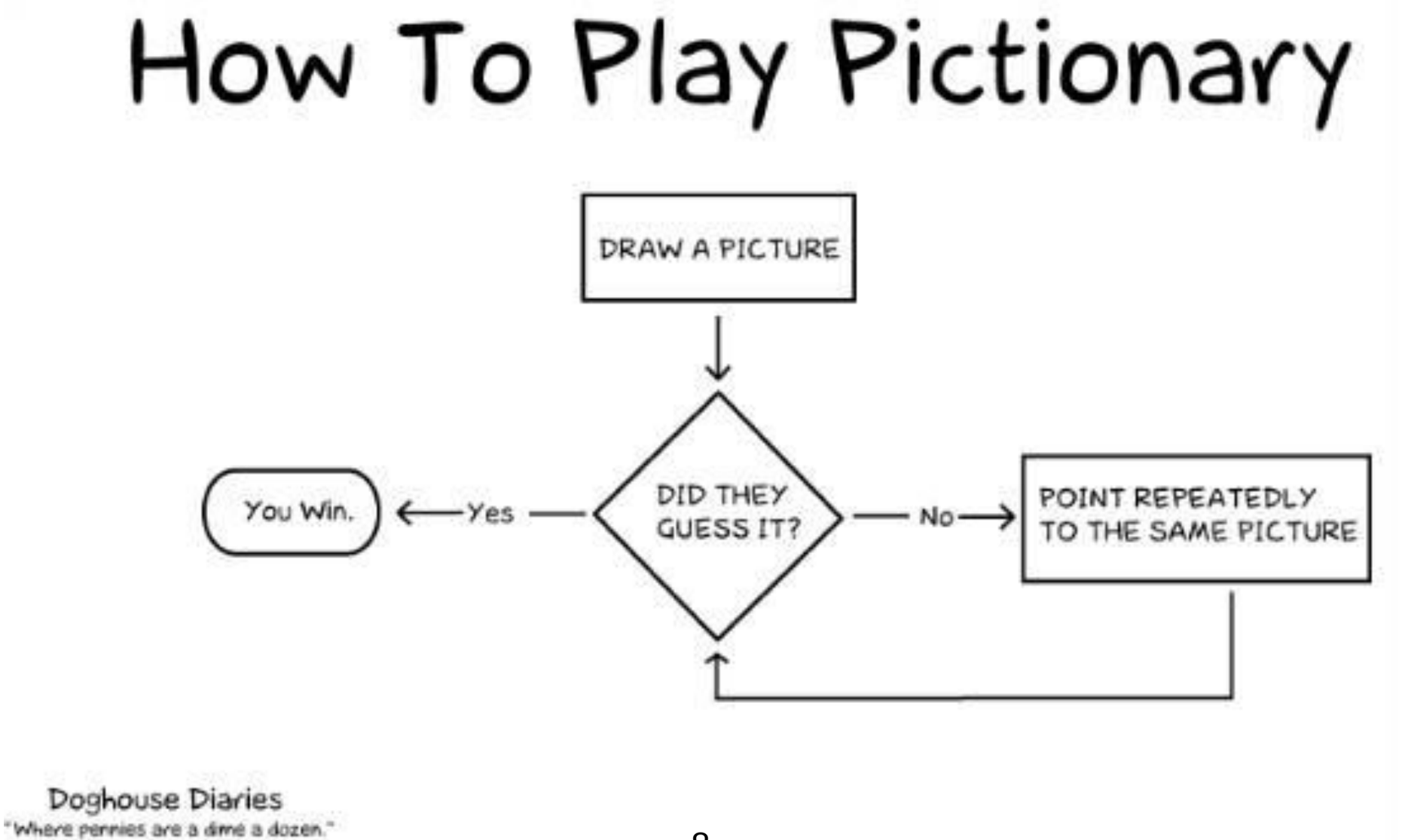
Every declaration in Processing has a *scope*: the part of the program source code in which that declaration is valid.

Usually either “global” or bounded by the nearest enclosing {}.

Scope is a complicated topic. If in doubt, just avoid re-using the same names!

Control flow

By default, Processing will execute statements in the order they're given. Control flow can modify that order.



Conditionals

```
if( keyPressed && key == ' ' ) {
  ellipse( mouseX, mouseY, 20, 20 );
}
```

An if statement

Conditionals

```
if( keyPressed && key == ' ' ) {
  ellipse( mouseX, mouseY, 20, 20 );
} else {
  rect( mouseX, mouseY, 20, 20 );
}
```

Conditionals

```
if ( keyPressed ) {
  if ( key == 'e' ) {
    ellipse( mouseX, mouseY, 20, 20 );
  } else if ( key == 'l' ) {
    line( 10, 10, 100, 100 );
  } else {
    rect( mouseX, mouseY, 20, 20 );
  }
}
```

While loops

```
int y = 0;

while( y < height ) {
  line( 0, y, width, y );
  y = y + 10;
}
```

For loops

```
for( int y = 0; y < height; y += 10 ) {
  line( 0, y, width, y );
}
```

Functions

A *function* gives a name to a computation.

Benefits:

- Ease of (error-free) repetition.
- Encapsulation: hide the messy details.
- Abstraction: think about problem solving at a higher level.
- Establish a point of connection between parts of a program.

Hooks

Processing knows about a few predetermined function names. If you define functions (*hooks*) with those names, Processing will call them at the right times.

Examples: setup(), draw(), mousePressed(), keyPressed()

Some libraries add more hooks.

Arrays

An *array* is a sequence of values, all of the same type, bundled into a single master value.

```
float[] ts1 = {
  -4.8,   -4.79,  -4.764, -4.762,
  -4.764, -4.824, /* 86 more numbers... */
  -1.083, -1.2,   -1.3,   -1.41
};
float[] ts2 = {
  -21.08933, -21.814,  -22.542, -22.01667,
  -20.912,   -21.564 /* 86 more numbers... */
  -27.48999, -27.43200, -27.88466, -28.09467
};
```

```
float[] ts1 = {
  -4.8,   -4.79,  -4.764, -4.762,
  -4.764, -4.824, /* 86 more numbers... */
  -1.083, -1.2,   -1.3,   -1.41
};
```

```
for( int idx = 0; idx < ts1.length; ++idx ) {
  if( ts1[idx] > 0.0 ) {
    println( "Where's my sunscreen?" );
  }
}
```

Classes and objects

- This is a review from CS105
- This material is in L01 due Wednesday this week (Jan 9)

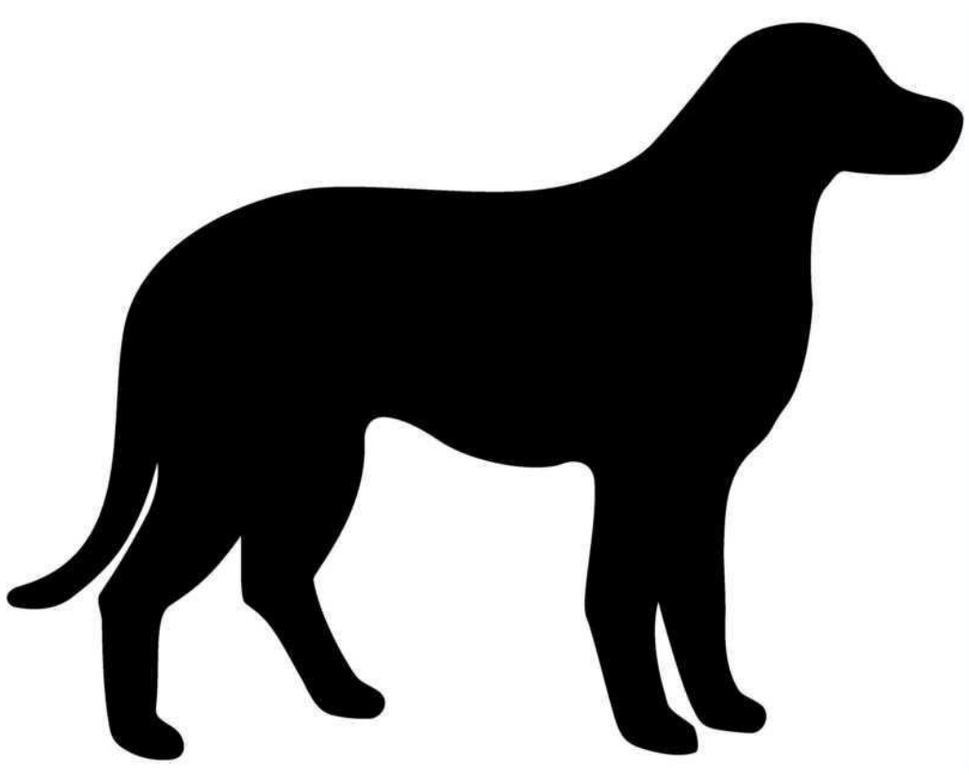
Objects

- **LP Chapter 8**
- object-oriented thinking
- declaring and creating
- assigning object "fields"
- calling object "methods“

- Slides created by Professor Dan Vogal.

The Properties and Actions of Something

Dog



Dog Properties

- Breed
- Weight
- Gender
- Colour

these are like **variables**

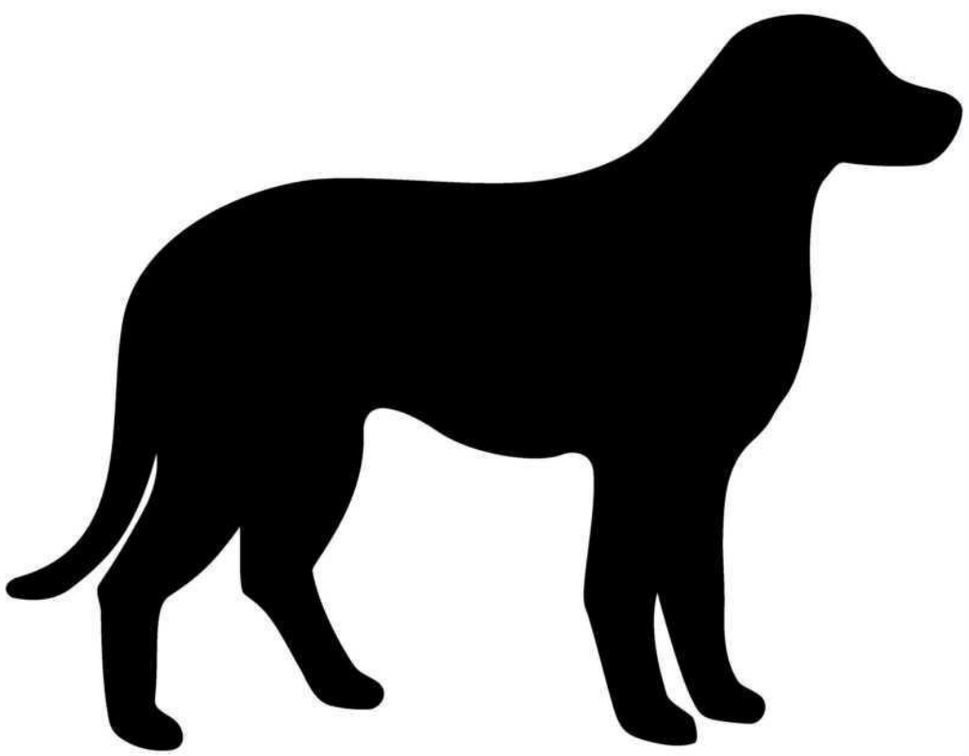
Dog Actions

- Sleep
- Eat
- Run
- Jump
- Fetch

these are like **functions**

Class vs. Object

A type of animal called "dog"



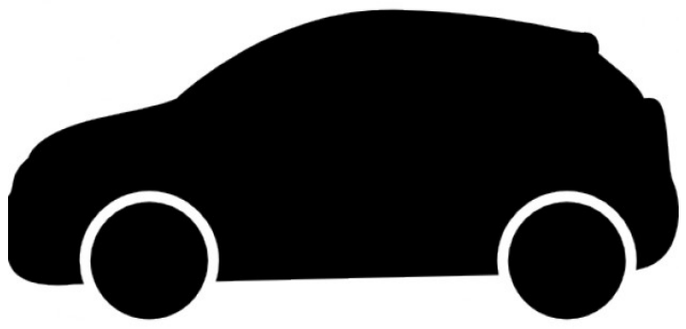
A 2 year-old German Shepherd named Rex

Class vs. Object

Car Class

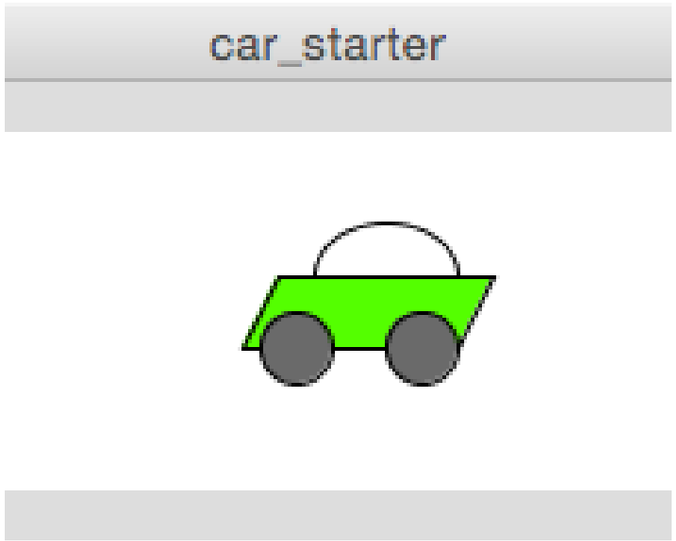
Properties
(variables)

Actions
(functions)



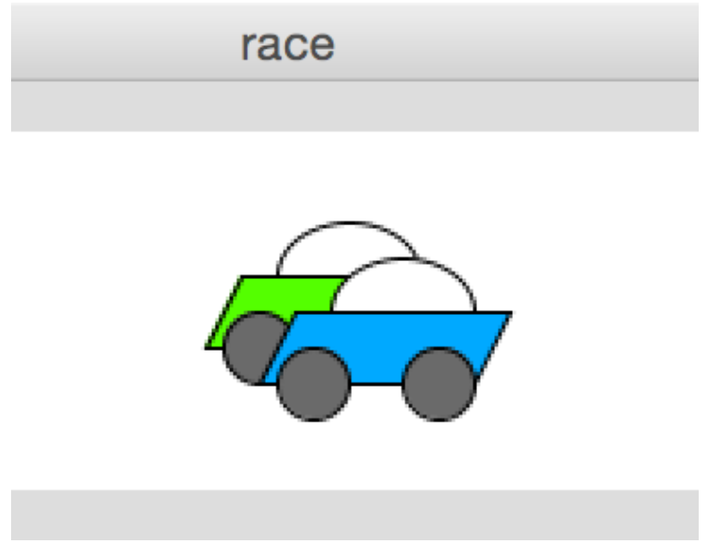
car (non object)

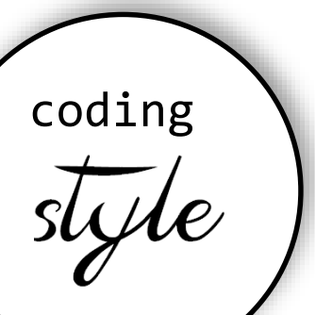
```
// car properties (variables)
float carX;
float carY;
float carHue;
float carSpeed;


// car actions (functions)
void carUpdate() { …

void carDraw(float x, float y, float hue) { …
```



race (defining car class)

```
class Car {

 // car properties (fields)
 float x;
 float y;
 float hue;
 float speed;

 // car actions (methods)

 void move() {
    …
 }

 void draw() {
    …
 }

}
```





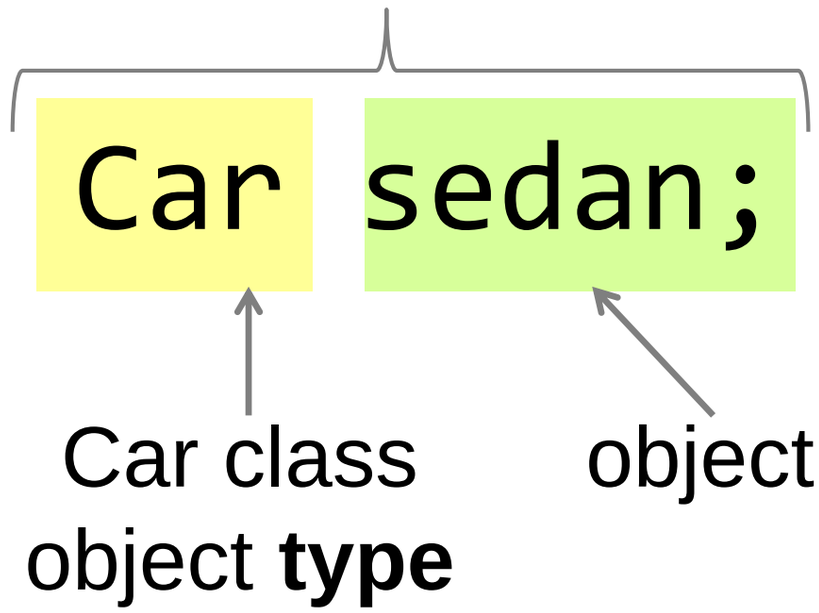
race (using car class)

```
Car car = new Car();

void setup() {
  size(600, 100);
  colorMode(HSB, 360, 100, 100, 100);

  car.x = width;
  car.y = 60;
  car.speed = 2;
  car.hue = 100;
}

void draw() {
  background(360);
  car.move();
  car.draw();
}
```

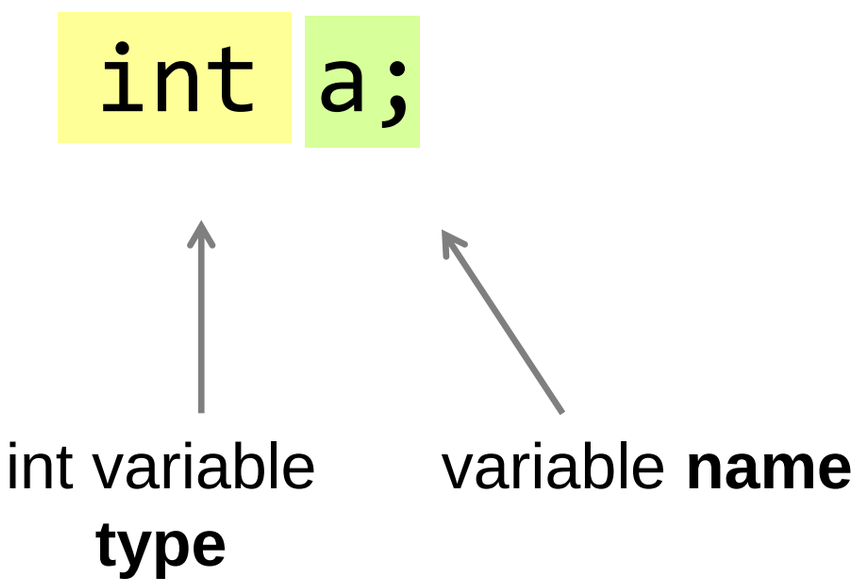
Declaring an Object

Object

declare a Car object called sedan



Variable
(for comparison)



Declare and Create an Object

- `new` is an operator that means “create”
- we need to “create” the object before using it
  - “normal” variables have a single value, no need to create
- creating the object also *initializes* the object

```
Car sedan = new Car();
```

declare an object of class Car

create an object of class Car

How to declare and **create** a Circle object called 'c'?

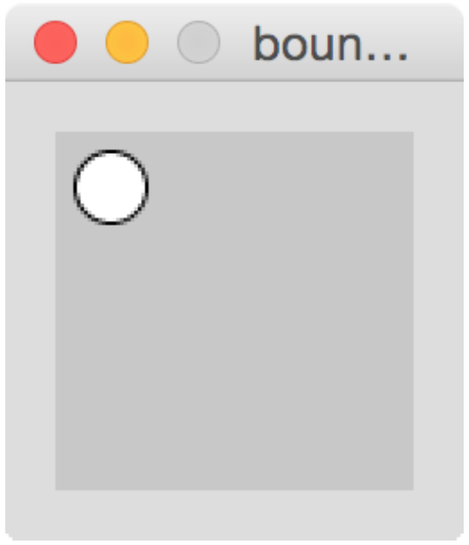
How to assign a value to the Circle 'c' radius?

race (using constructor)

```
Car(float xIn, float yIn, float speedIn, float hueIn) {
      x = xIn;
      y = yIn;
      hue = hueIn;
      speed = speedIn;
  }
```

In (“in”) is added to each parameter to avoid shadowing the object fields

bounce (non-object)




```
// car properties (variables)
float carX;
float carY;
float carHue;
float carSpeed;


// car actions (functions)
void carUpdate() { …

void carDraw(float x, float y, float hue) { …
```

bounce1 (ball class)

```
class Ball {

  // dot position, speed, and size
  float size = 20;
  float x = 50;
  float y = 50;
  // start moving to the left
  float xSpeed = -1.1;
  // start moving more slowly upward
  float ySpeed = 0.6;


  void moveAndDisplay() {
    …
  }
```

bounce1 (using ball class)

```
Ball ball = new Ball();

void draw() {
  background(200);

  ball.moveAndDisplay();
}
```

Anatomy of a Class

class name

fields

constructor

method

```
class MyClass {

  float x;
  int i;

  MyClass() {
    x = width;
    i = 99;
  }

  void doSomething() {
    text(i, x, 50);
  }
}
```

bounce2 (constructor)

```
class Ball {

  // dot position, speed, and size
  float size;
  float x;
  float y;
  float xSpeed;
  float ySpeed;
```

```
Ball() {
  x = 50;
  y = 50;
  xSpeed = random(-2, 2);
  ySpeed = random(-2, 2);
  size = random(10, 30);
}
```

…

bounce2 (three ball objects)

```
Ball ball1 = new Ball();
Ball ball2 = new Ball();
Ball ball3 = new Ball();

void draw() {
  background(200);

  ball1.moveAndDisplay();
  ball2.moveAndDisplay();
  ball3.moveAndDisplay();
}
```

bounce3 (many ball objects with array)

```
Ball[] balls = new Ball[10];

void setup() {
  for (int i = 0; i < balls.length; i++) {
    balls[i] = new Ball();
  }
}
```

```
void draw() {
  background(200);

  for (int i = 0; i < balls.length; i++) {
    balls[i].moveAndDisplay();
  }
}
```


iclicker