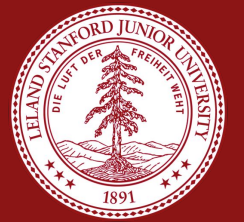
More Methods and Graphics

Lecture 8

CS106A, Summer 2019
Sarai Gould && Laura Cruz-Albrecht

With inspiration from slides created by Keith Schwarz, Mehran Sahami, Eric Roberts, Stuart Reges, Chris Piech and others.



Announcements

- Hope you had a great 4th!
- Assignment 2 due Wednesday July 10th, at 10AM

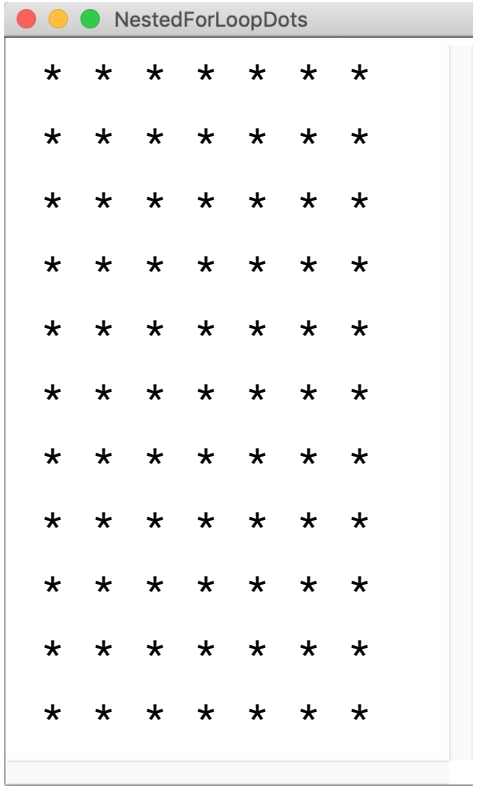
Plan for Today

- Review: Nested Loops, Infinite Loops, Intro to Graphics
- Returning to Returns
- More Graphics!
- Stoplights

Plan for Today

- Review: Nested Loops, Infinite Loops, Intro to Graphics
- Returning to Returns
- More Graphics!
- Stoplights

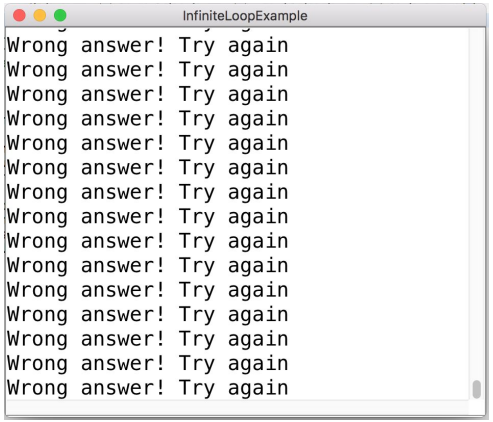
Review: Nested For Loops




```
for (int i = 0; i < NUM_ROWS; i++) {
    for (int j = 0; j < NUM_COLS; j++) {
        print("*");
    }
    println();
}
```

Review: Infinite Loops

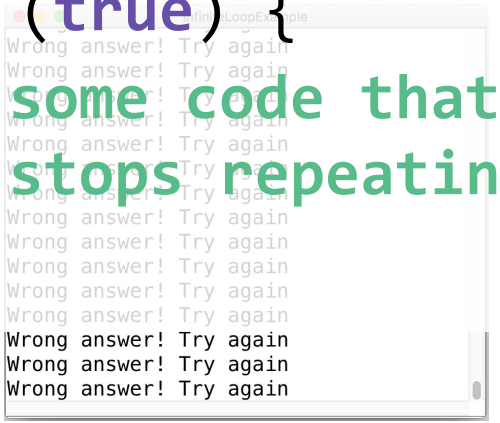
```
int answer = readInt("What is 1 + 1?");

while (answer != 2){
     println("Wrong answer! Try again");
}
```



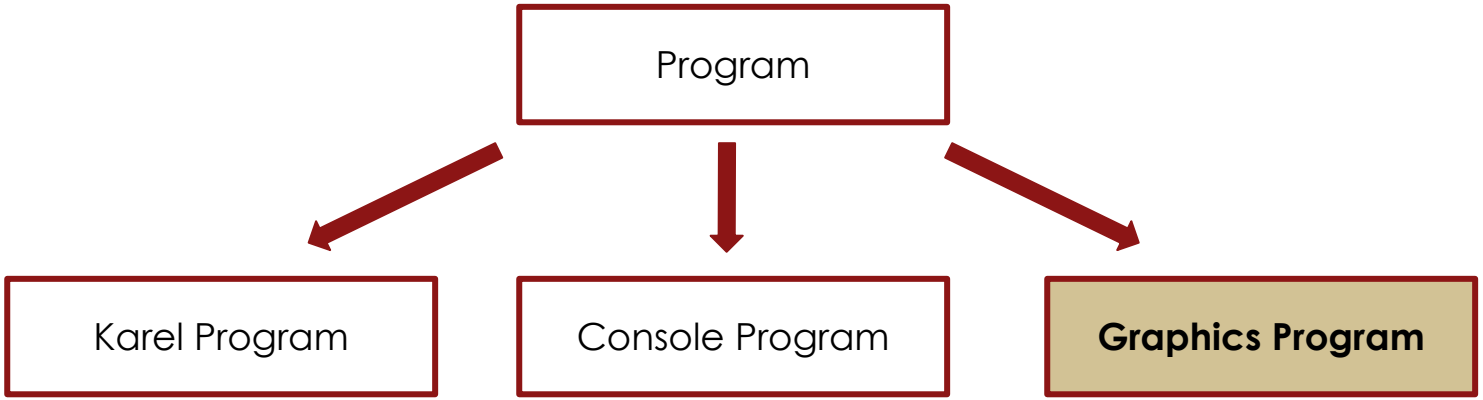
Review: Infinite Loops

```
int answer = readInt("What is 1 + 1?");

while (answer != 2){
     println("Wrong answer! Try again");
}
```

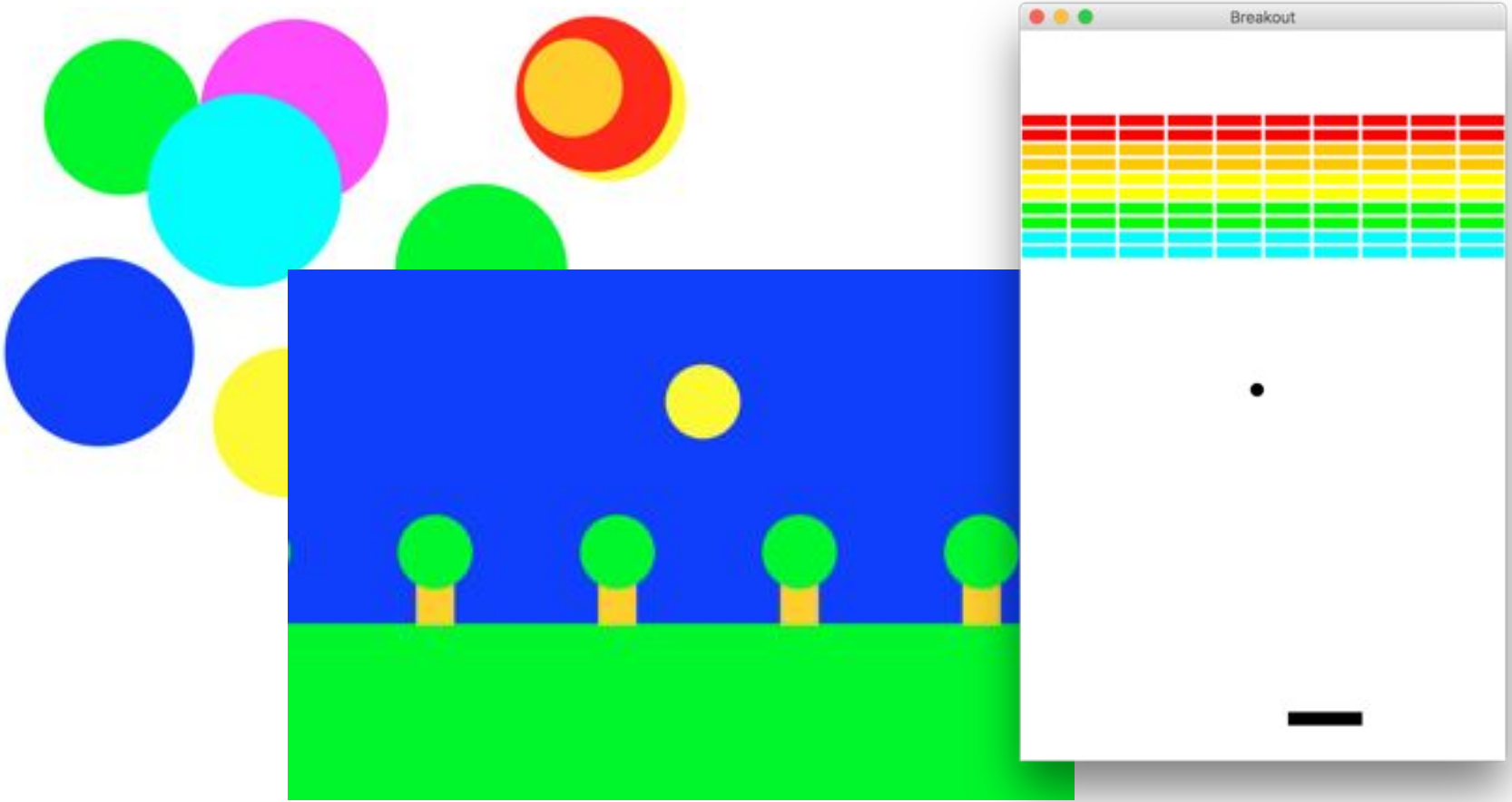
```
while (true) {
    // some code that never
    // stops repeating ...
}
```



Review: Intro to Graphics Programs

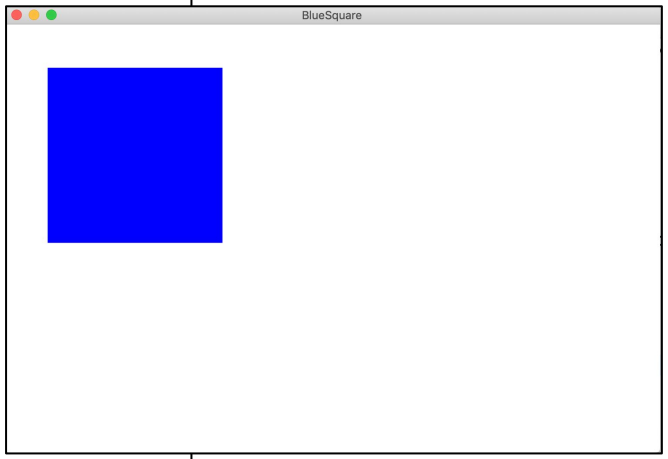


Review: Intro to Graphics Programs



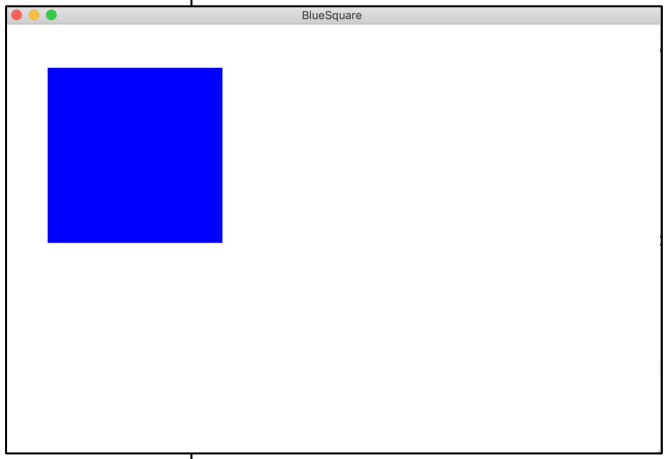
Review: Our First Graphics Program

```
import acm.program.*;
import acm.graphics.*; // Stanford graphical objects
import java.awt.*;     // Java graphical objects

public class BlueSquare extends GraphicsProgram {
     public void run(){
          drawBlueSquare();
     }

     private void drawBlueSquare(){
          GRect rect = new GRect(200, 200);
          rect.setFilled(true);
          rect.setColor(Color.BLUE);
          add(rect, 50, 50);
     }
}
```

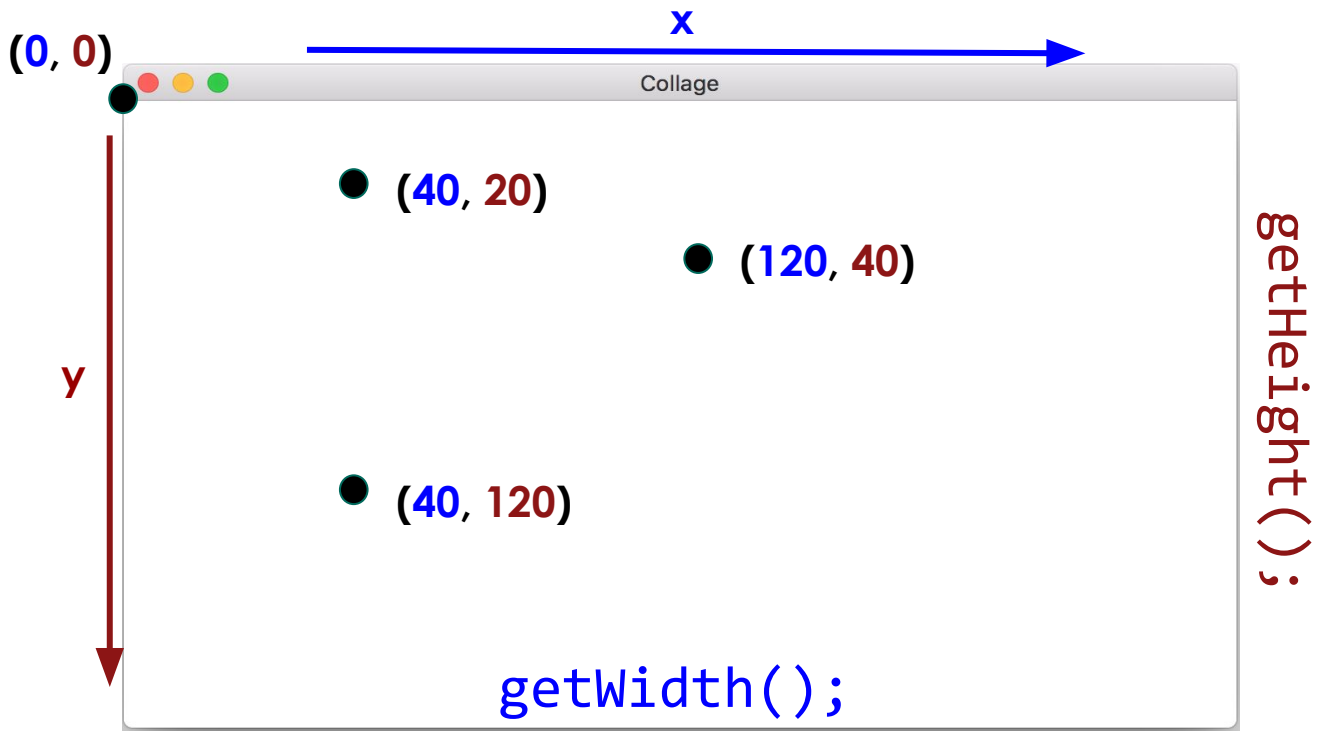


Review: Our First Graphics Program

```
import acm.program.*;
import acm.graphics.*; // Stanford graphical objects
import java.awt.*;     // Java graphical objects

public class BlueSquare extends GraphicsProgram {
     public void run(){
          drawBlueSquare();
     }

     private void drawBlueSquare(){
          GRect rect = new GRect(200, 200);
          rect.setFilled(true);
          rect.setColor(Color.BLUE);
          add(rect, 50, 50);
     }
}
```

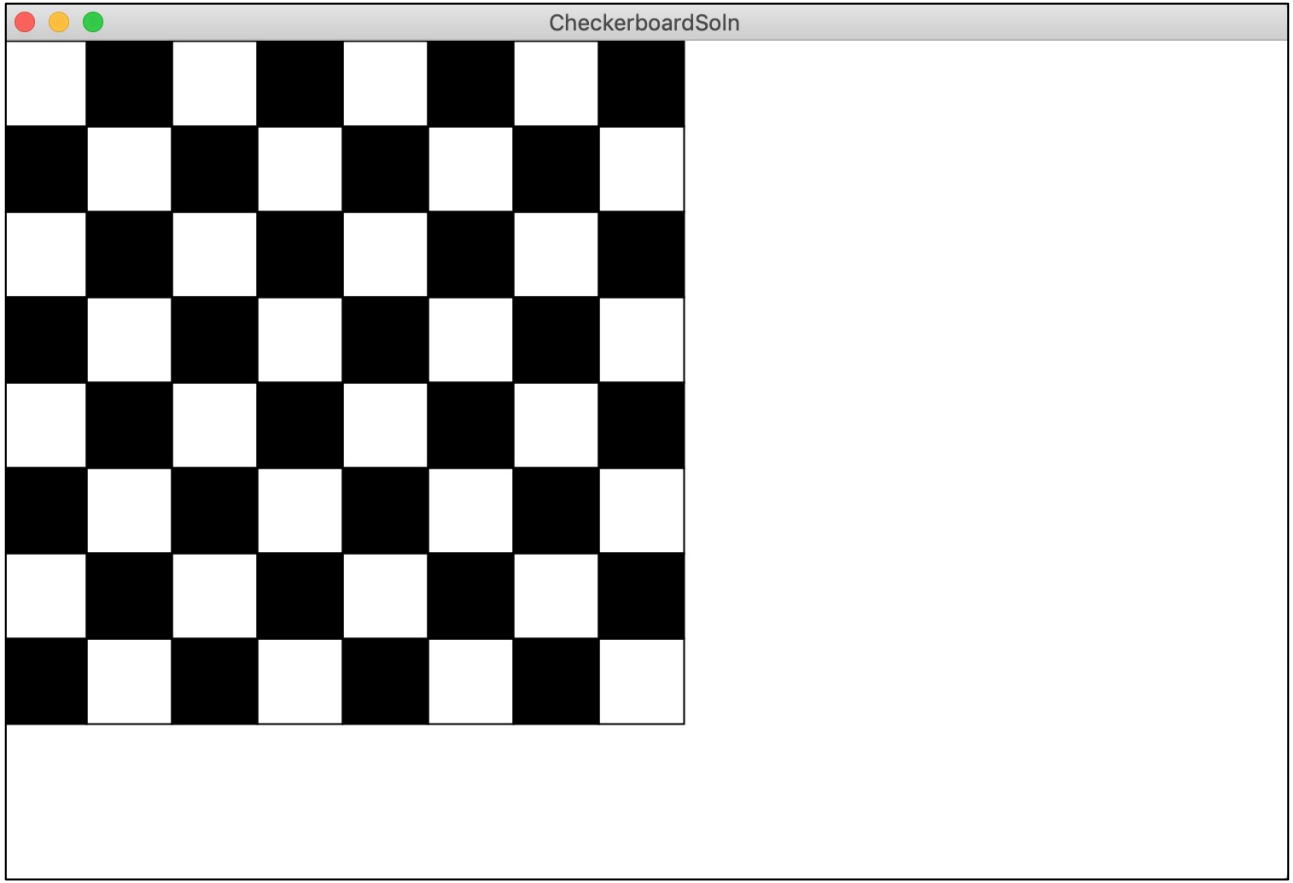


Review: The Graphics Canvas



* Note: The y coordinate gets *bigger* as we go *down*. This is opposite of how it acts in most math classes!

Review: Checkerboard



Plan for Today

- Review: Nested Loops, Infinite Loops, Intro to Graphics
- Returning to Returns
- More Graphics!
- Stoplights

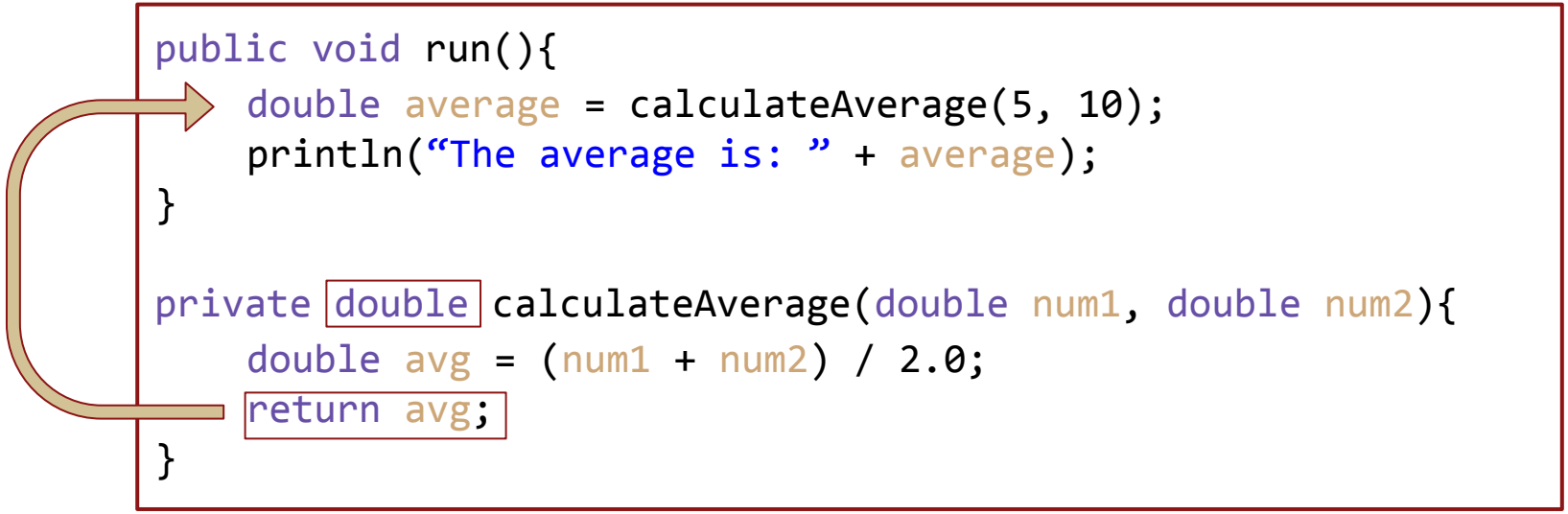
Returning to Returns

- Recall that methods can `return` data

```
public void run(){
    double average = calculateAverage(5, 10);
    println("The average is: " + average);
}

private double calculateAverage(double num1, double num2){
    double avg = (num1 + num2) / 2.0;
    return avg;
}
```

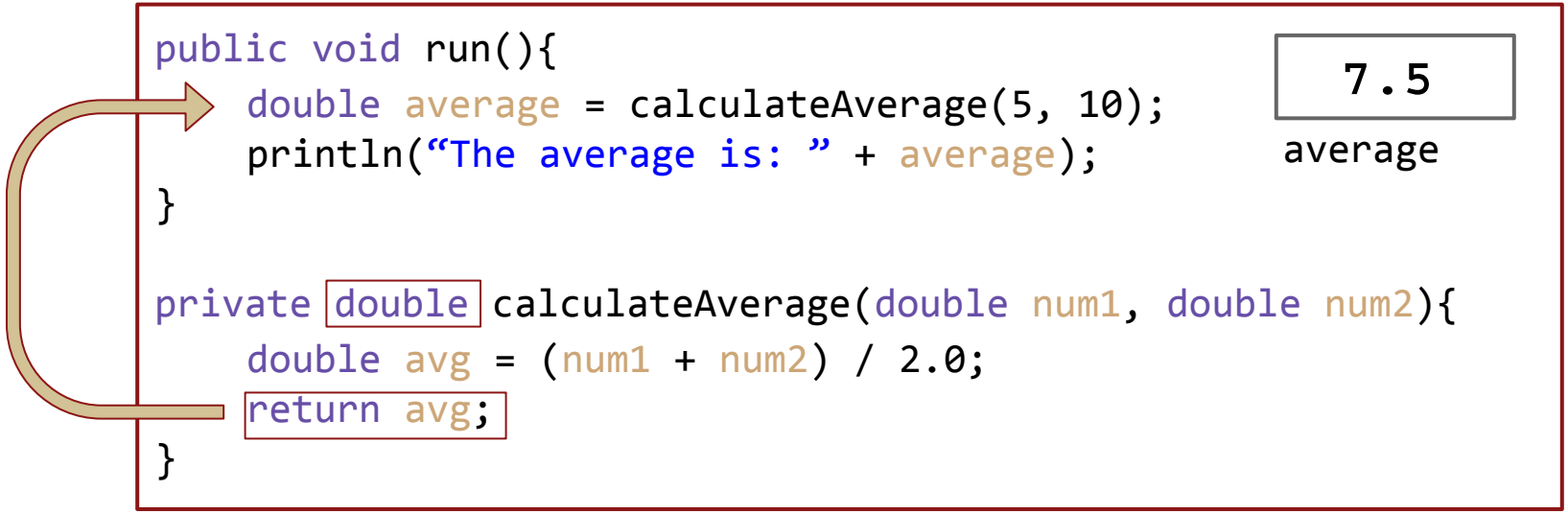
Returning to Returns

- Recall that methods can `return` data



Returning to Returns

- Recall that methods can `return` data



Multiple Return Statements

- You can have multiple return statements in a method

Multiple Return Statements

- You can have multiple return statements in a method

```
private int max(int num1, int num2) {
    if (num1 >= num2) {
        return num1;
    }
    return num2;
}
```

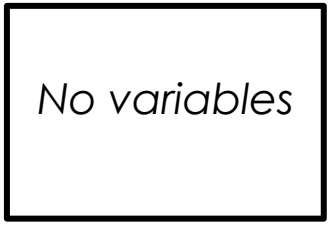
Multiple Return Statements

```
public void run() {
    int larger = max(5, 1);
}

private int max(int num1, int num2) {
    if (num1 >= num2) {
        return num1;
    }
    return num2;
}
```

Multiple Return Statements

```
public void run() {
    int larger = max(5, 1);
}

private int max(int num1, int num2) {
    if (num1 >= num2) {
        return num1;
    }
    return num2;
}
```

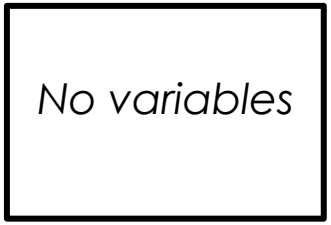
run memory



Multiple Return Statements

```
public void run() {
    int larger = max(5, 1);
}

private int max(int num1, int num2) {
    if (num1 >= num2) {
        return num1;
    }
    return num2;
}
```

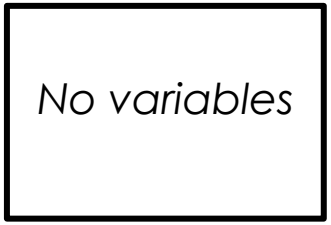
run memory



Multiple Return Statements

```
public void run() {
    int larger = max(5, 1);
}

private int max(int num1, int num2) {
    if (num1 >= num2) {
        return num1;
    }
    return num2;
}
```

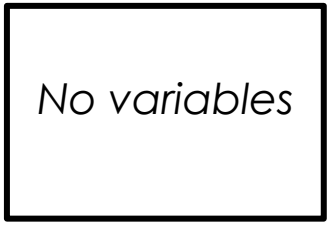
run memory



Multiple Return Statements

```
public void run() {
    int larger = max(5, 1);
}

private int max(int num1, int num2) {
    if (num1 >= num2) {
        return num1;
    }
    return num2;
}
```

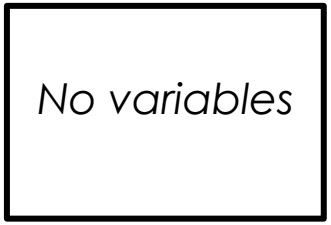
run memory



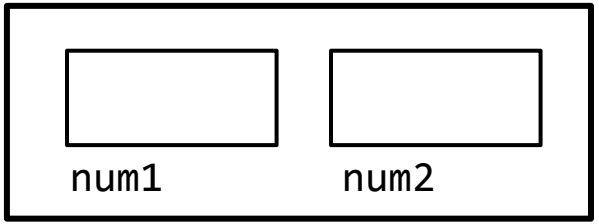
Multiple Return Statements

```
public void run() {
    int larger = max(5, 1);
}

private int max(int num1, int num2) {
    if (num1 >= num2) {
        return num1;
    }
    return num2;
}
```

run memory



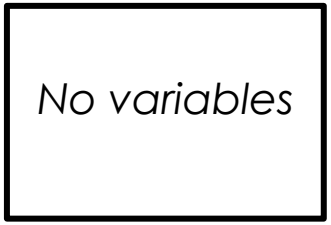
max memory



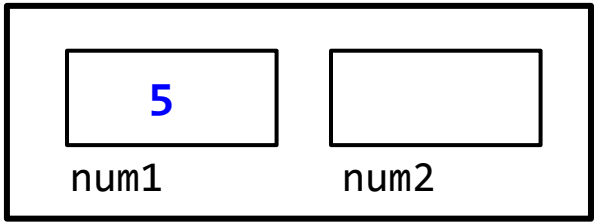
Multiple Return Statements

```
public void run() {
    int larger = max(5, 1);
}

private int max(int num1, int num2) {
    if (num1 >= num2) {
        return num1;
    }
    return num2;
}
```

run memory



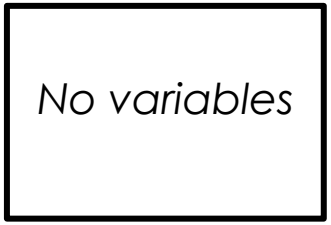
max memory



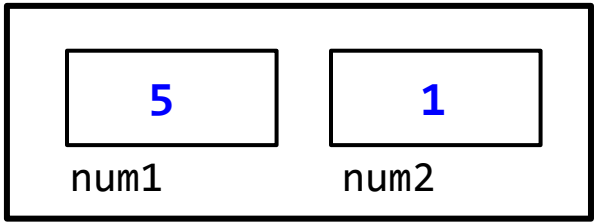
Multiple Return Statements

```
public void run() {
    int larger = max(5, 1);
}

private int max(int num1, int num2) {
    if (num1 >= num2) {
        return num1;
    }
    return num2;
}
```

run memory



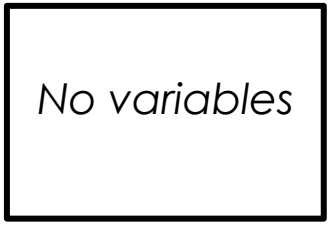
max memory



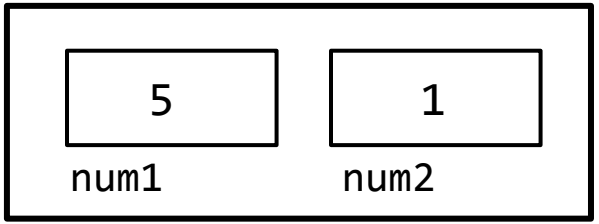
Multiple Return Statements

```
public void run() {
    int larger = max(5, 1);
}

private int max(int num1, int num2) {
    if (num1 >= num2) {
        return num1;
    }
    return num2;
}
```

run memory



max memory

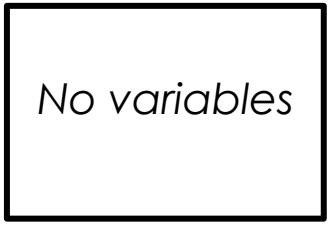


Multiple Return Statements

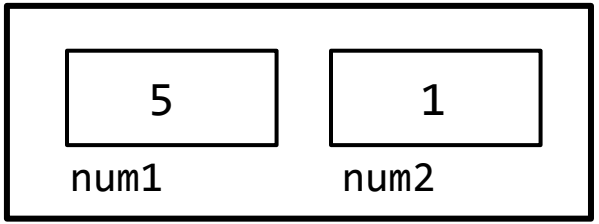
```
public void run() {
    int larger = max(5, 1);
}

private int max(int num1, int num2) {
    if (num1 >= num2) {
        return num1;
    }
    return num2;
}
```

5

run memory



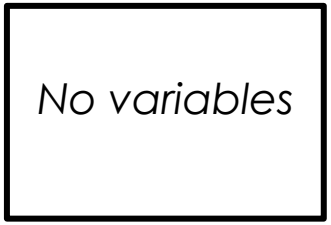
max memory



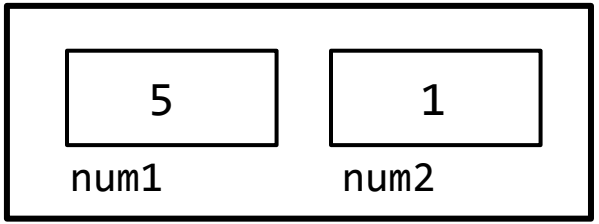
Multiple Return Statements

```
public void run() {
    int larger = max(5, 1); 5
}

private int max(int num1, int num2) {
    if (num1 >= num2) {
        return num1;
    }
    return num2;
}
```

run memory



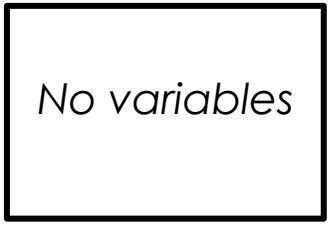
max memory



Multiple Return Statements

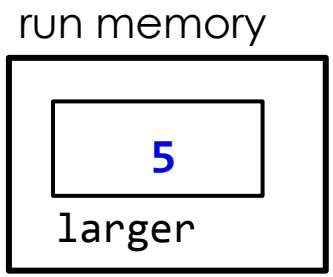
```
public void run() {
    int larger = max(5, 1); 5
}

private int max(int num1, int num2) {
    if (num1 >= num2) {
        return num1;
    }
    return num2;
}
```

run memory



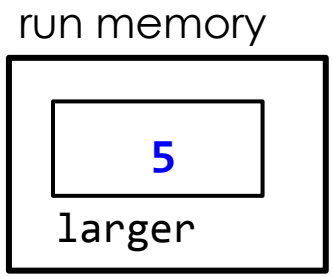
Multiple Return Statements

```
public void run() {
    int larger = max(5, 1); 5
}

private int max(int num1, int num2) {
    if (num1 >= num2) {
        return num1;
    }
    return num2;
}
```



Multiple Return Statements

```
public void run() {
    int larger = max(5, 1);
}

private int max(int num1, int num2) {
    if (num1 >= num2) {
        return num1;
    }
    return num2;
}
```

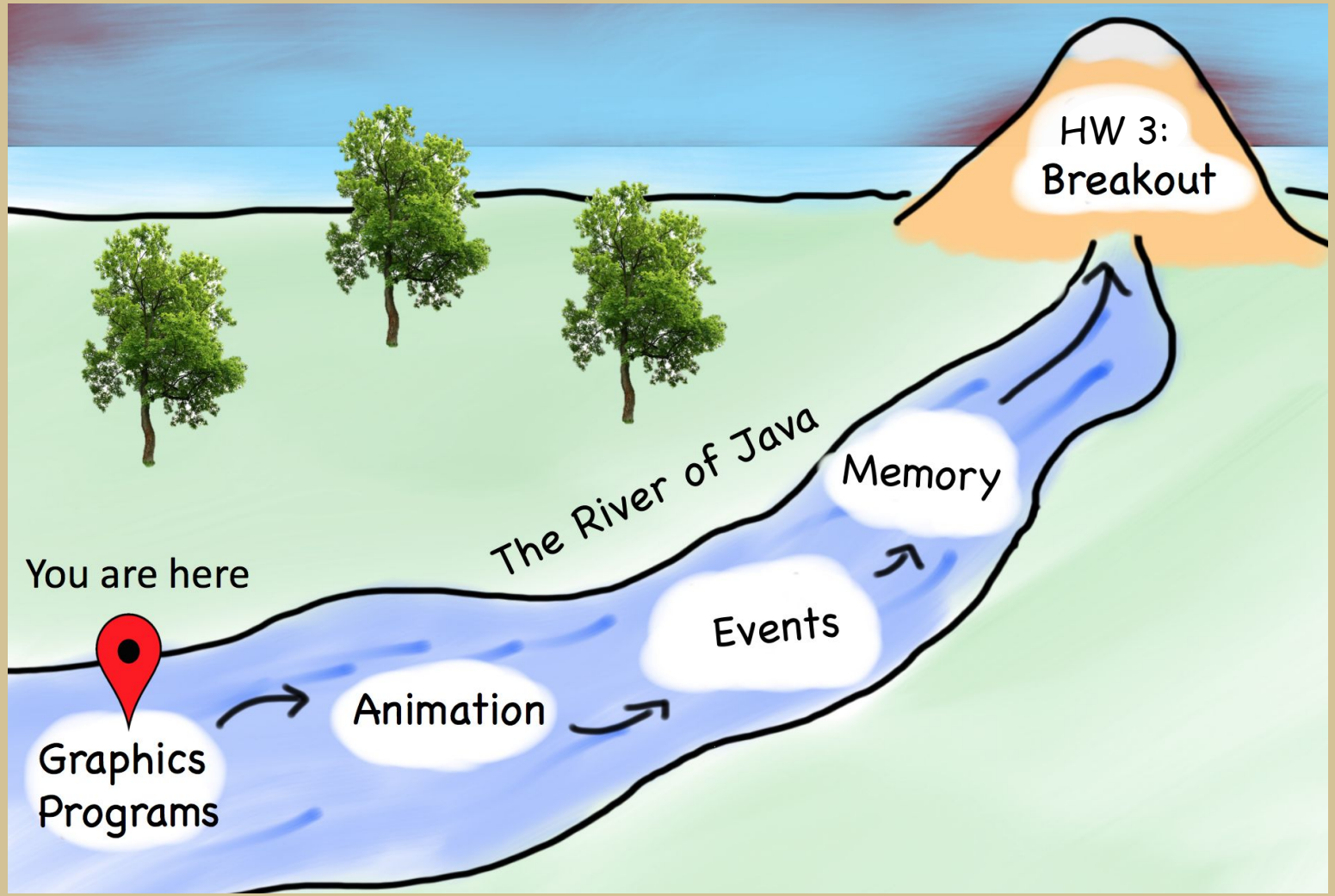


Returning to our scheduled programming

```
private topic getTopic() {
    if (isMonday()) {
        return moreGraphics;
    } else if (isTues()) {
        return animation;
    } else {
        return randomExcitingTopic;
    }
}
```

Plan for Today

- Review: Nested Loops, Infinite Loops, Intro to Graphics
- Returning to Returns
- More Graphics!
- Stoplights


HW 3:
Breakout
The River of Java
Memory
You are here
Events
Animation
Graphics
Programs

Working with Graphics

We make graphics programs by *creating graphics objects* and *manipulating their properties*.

Working with Graphics

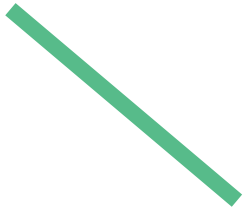
We make graphics programs by *creating* ***graphics objects*** and *manipulating their properties*.

Graphical Objects

GRect

GOval

GLine



GLabel

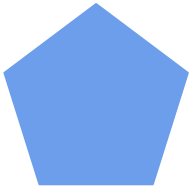
Hello

GImage

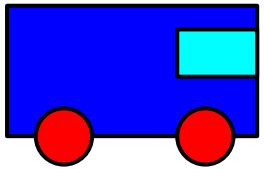
GRoundRect

GArc

GPolygon



GCompound



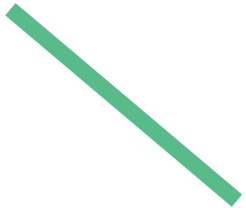
and more ...

Graphical Objects

GRect

GOval

GLine



GLabel

Hello

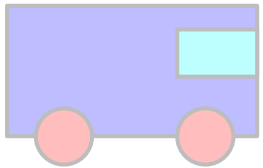
GImage

GRoundRect

GArc

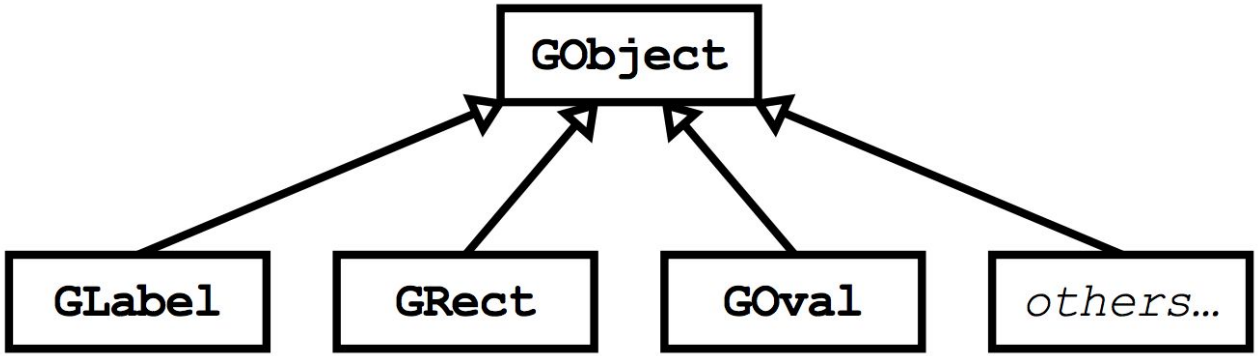
GPolygon

GCompound



and more ...

Graphical Objects



Working with Graphics

We make graphics programs by ***creating*** *graphics objects* and *manipulating their properties*.

Creating Graphics Objects

- To create a graphics object, we need to:
  - declare a variable to hold that object, and
  - actually create the object using the **new** keyword.

- Initialization syntax:

```
type name = new type(...);
```

- Example:

```
GRect rect = new GRect(100, 200);
```

Creating Graphics Objects

- To create a graphics object, we need to:
  - declare a variable to hold that object, and
  - actually create the object using the **new** keyword.

- Initialization syntax:

```
type name = new type(...);
```

- Example:

```
GRect rect = new GRect(100, 200);
```

This is called a "constructor"

Primitives vs. Objects

Primitive Variable Types

```
int
double
char
boolean
```

Object Variable Types

```
GRect
GOval
GLine
...
```

Primitives vs. Objects

Primitive Variable Types	Object Variable Types
`int`	`GRect`
`double`	`GOval`
`char`	`GLine`
`boolean`	`...`

Object variables:

Primitives vs. Objects

Primitive Variable Types	Object Variable Types
`int`	`GRect`
`double`	`GOval`
`char`	`GLine`
`boolean`	`...`

Object variables:

1. Have UpperCamelCase types

Primitives vs. Objects

Primitive Variable Types	Object Variable Types
`int`	`GRect`
`double`	`GOval`
`char`	`GLine`
`boolean`	`...`

Object variables:

1. Have UpperCamelCase types
2. Are constructed using `new`

Primitives vs. Objects

Primitive Variable Types	Object Variable Types
`int`	`GRect`
`double`	`GOval`
`char`	`GLine`
`boolean`	`...`

Object variables:

1. Have UpperCamelCase types
2. Are constructed using `new`
3. You can call methods on them

Working with Graphics

We make graphics programs by *creating graphics objects* and ***manipulating their properties***.

Methods on Graphics Objects

- You can manipulate graphics objects by calling methods on those objects
- To call a method on an object, use the syntax:

```
object.method(parameters);
```

- Example:

```
rect.setColor(Color.BLUE);
```

Methods on Graphics Objects

- You can manipulate graphics objects by calling methods on those objects
- To call a method on an object, use the syntax:

```
object.method(parameters);
```

receiver — message

- Example:

```
rect.setColor(Color.BLUE);
```

Methods on Graphics Objects

- You can manipulate graphics objects by calling methods on those objects
- To call a method on an object, use the syntax:

```
object.method(parameters);
```

- Example:

```
rect.setColor(Color.BLUE);
```

who?

Methods on Graphics Objects

- You can manipulate graphics objects by calling methods on those objects
- To call a method on an object, use the syntax:

```
object.method(parameters);
```

- Example:

```
rect.setColor(Color.BLUE);
```

who? what?

Methods on Graphics Objects

- You can manipulate graphics objects by calling methods on those objects
- To call a method on an object, use the syntax:

```
object.method(parameters);
```

- Example:

```
rect.setColor(Color.BLUE);
```

who? what? what specifically?

GObject Methods

The following operations apply to all `GObjects`:

object`.setColor(`*color*`)` Sets the color of the object to the specified color constant.
object`.setLocation(`*x*`,` *y*`)` Changes the location of the object to the point (*x*, *y*).
object`.move(`*dx*`,` *dy*`)` Moves the object on the screen by adding *dx* and *dy* to its current coordinates.
object`.getWidth()` Returns the width of the object
object`.getHeight()` Returns the height of the object
and more…

Graphic courtesy of Eric Roberts

Colors

- Specified as predefined `Color` constants:

  `Color.NAME`, where `NAME` is one of:

BLACK	BLUE	CYAN	DARK_GRAY	GRAY
GREEN	LIGHT_GRAY	MAGENTA	ORANGE	PINK
RED	WHITE	YELLOW		

```
rect.setColor(Color.MAGENTA);
```

Colors

- Specified as predefined `Color` constants:

  `Color.NAME`, where `NAME` is one of:

BLACK	BLUE	CYAN	DARK_GRAY	GRAY
GREEN	LIGHT_GRAY	MAGENTA	ORANGE	PINK
RED	WHITE	YELLOW		

```
rect.setColor(Color.MAGENTA);
```

- Or, create one using Red-Green-Blue (RGB) values of 0-255

```
new Color(red, green, blue);
```

ex:

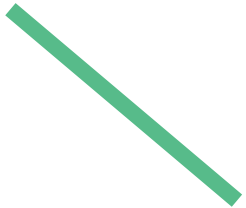
```
rect.setColor(new Color(218, 165, 32)); // goldenrold
```

Graphical Objects

GRect

GOval

GLine



GLabel

Hello

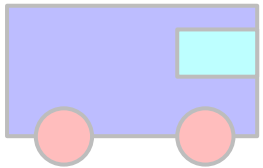
GImage

GRoundRect

GArc

GPolygon

GCompound

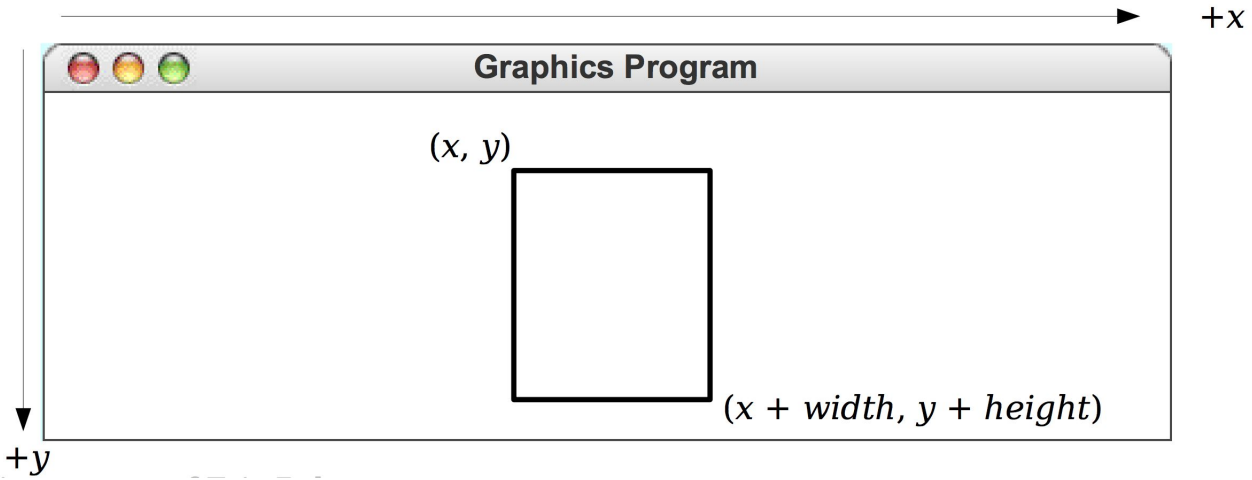


and more ...

GRect

```
new GRect(x, y, width, height);
```
Creates a rectangle with the given width and height, whose upper-left corner is at (x, y).

```
new GRect(width, height);
```
Same as above, but defaults (x, y) to (0, 0).



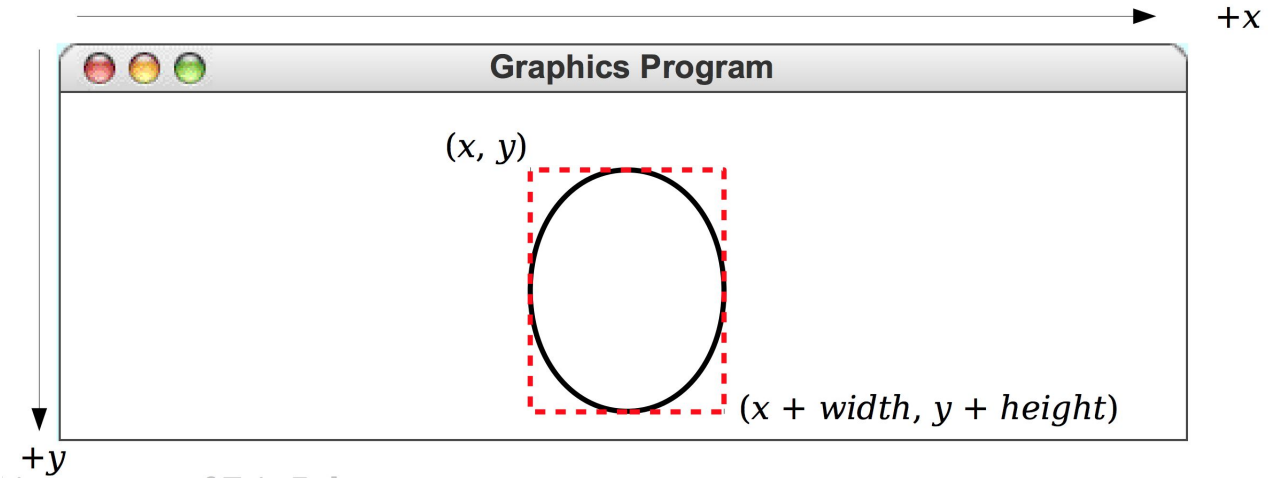
GOval

```
new GOval(x, y, width, height);
```

Creates an oval that fits within a rectangle with the given width and height, whose upper-left corner is at (x, y).

```
new GOval(width, height);
```

Same as above, but defaults (x, y) to (0, 0).



GRect & GOval

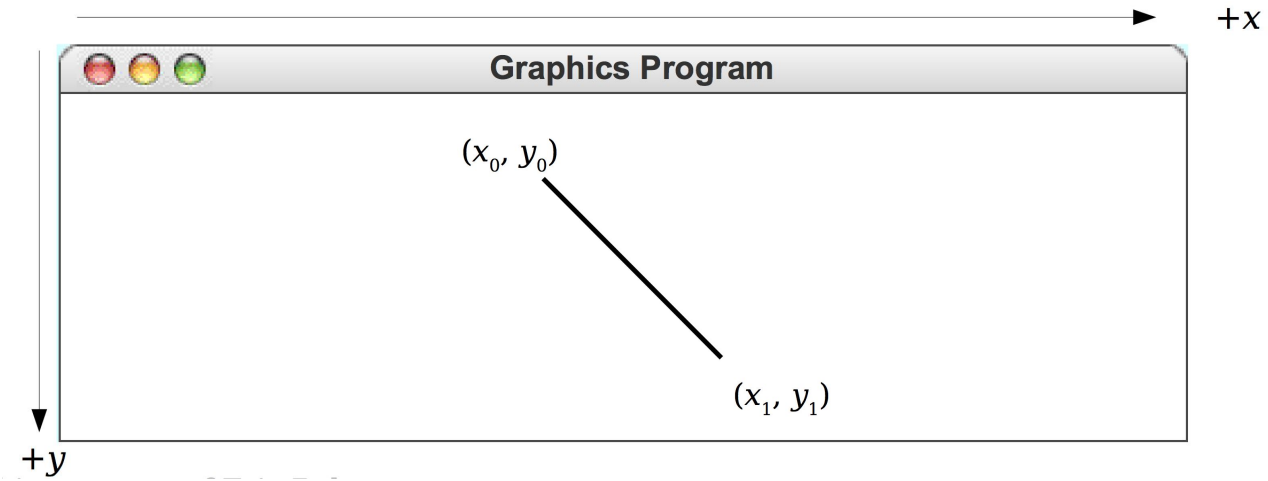
Methods shared by the `GRect` and `GOval` classes:

***object*`.setFilled(`*`fill`*`);`** If *fill* is `true`, fills in the interior of the object; if `false`, shows only the outline.
***object*`.setFillColor(`*`color`*`);`** Sets the color used to fill the interior, which can be different from the border.
***object*`.setSize(`*`width, height`*`);`** Sets the object's size to be the given width and height.

GLine

```
new GLine(x0, y0, x1, y1);
```

Creates a line extending from (x0, y0) to (x1, y1).



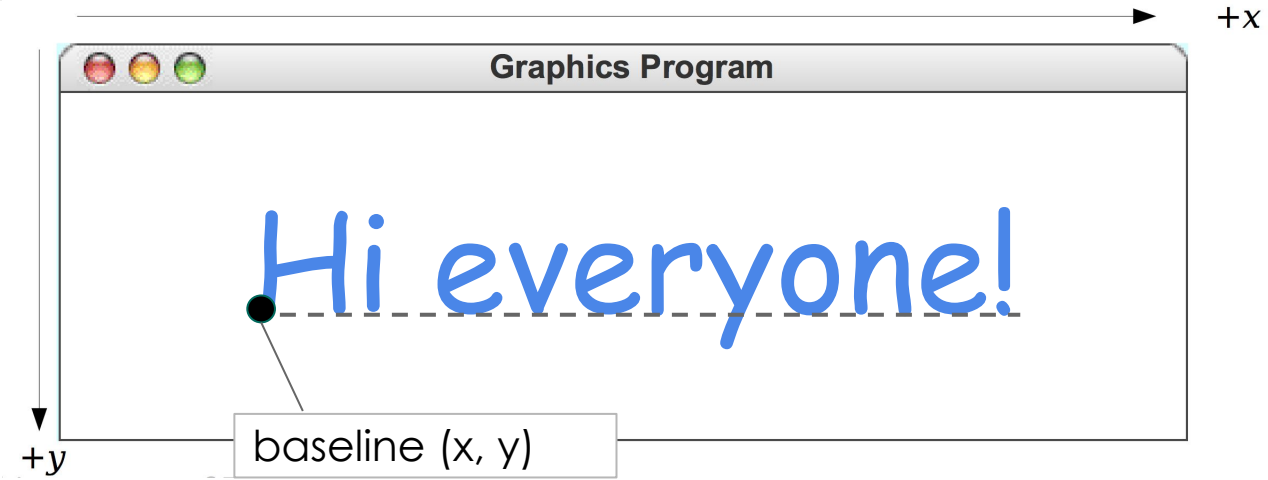
GLabel

`new GLabel("your text here", x, y);`

Creates a label with the given text, whose ***baseline*** starts at (x, y). Not positioned according to the top-left corner!

`new GLabel("your text here");`

Same as above, but defaults (x, y) to (0, 0).



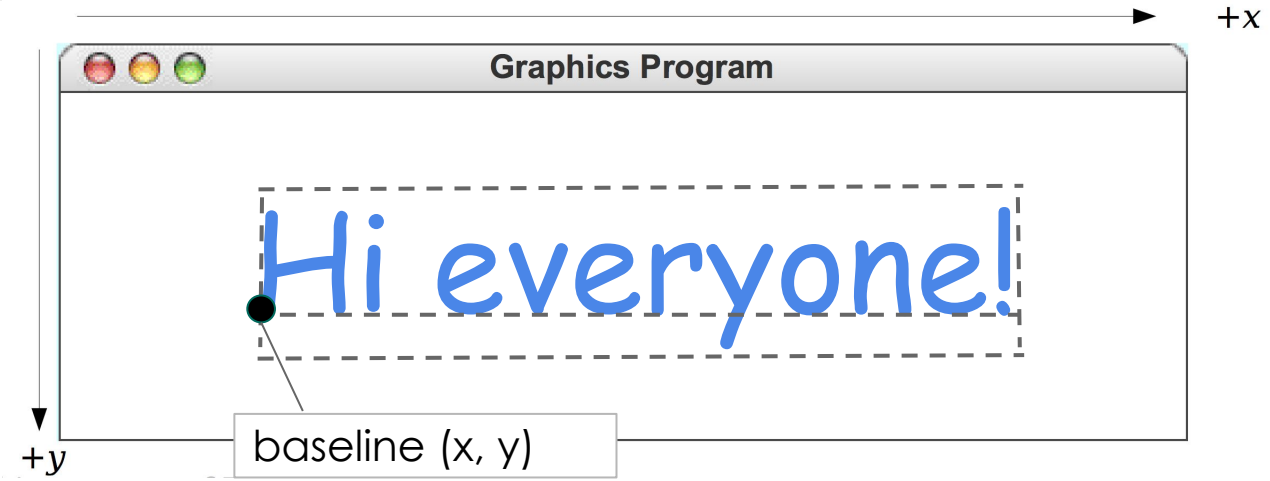
GLabel

```
new GLabel("your text here", x, y);
```

Creates a label with the given text, whose ***baseline*** starts at (x, y). Not positioned according to the top-left corner!

```
new GLabel("your text here");
```

Same as above, but defaults (x, y) to (0, 0).



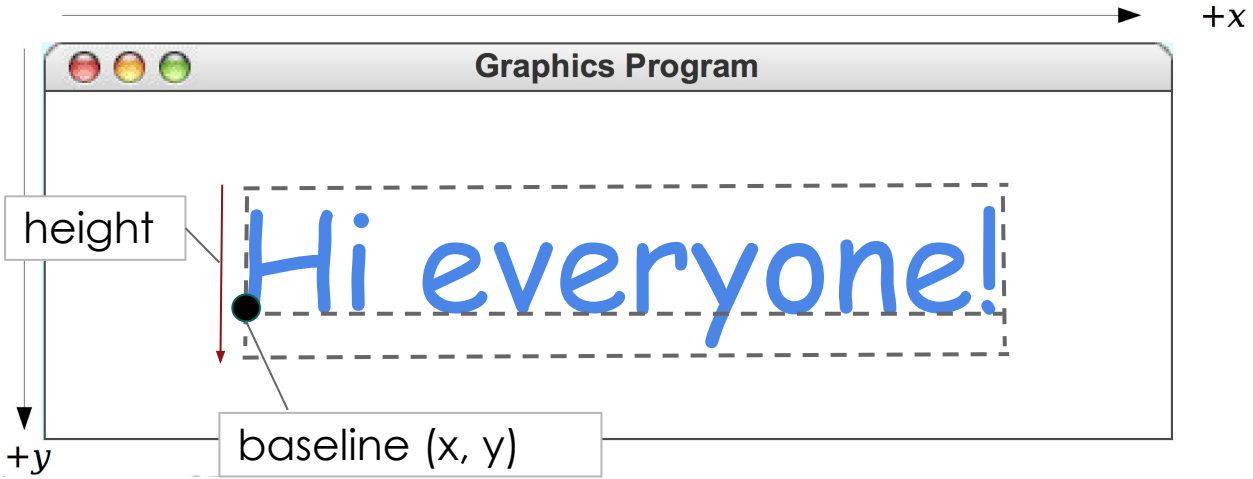
GLabel

`new GLabel("your text here", x, y);`

Creates a label with the given text, whose ***baseline*** starts at (x, y). Not positioned according to the top-left corner!

`new GLabel("your text here");`

Same as above, but defaults (x, y) to (0, 0).



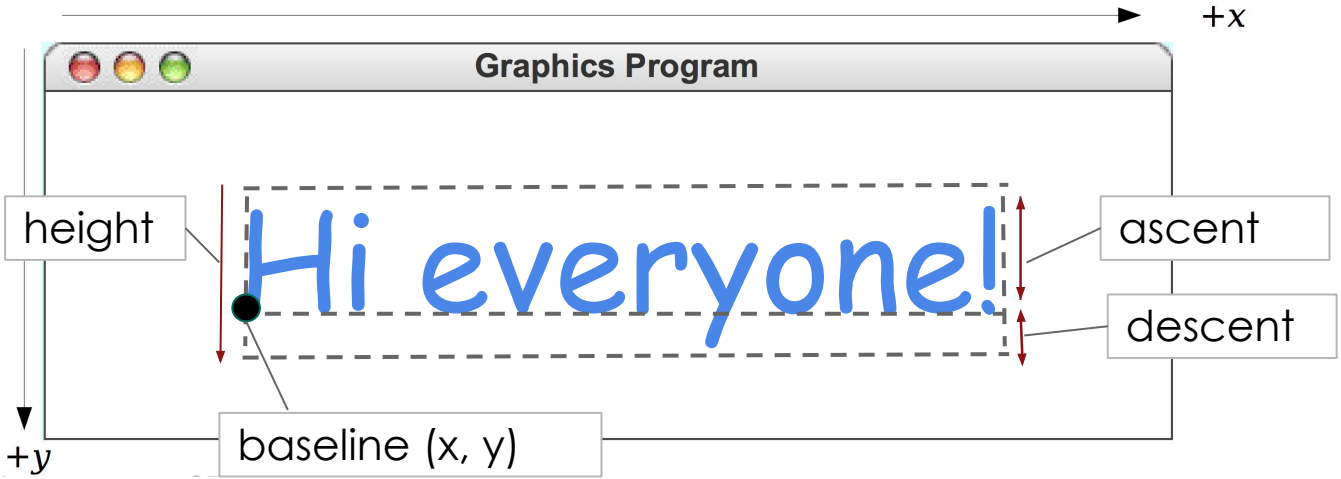
GLabel

`new GLabel("your text here", x, y);`

Creates a label with the given text, whose ***baseline*** starts at (x, y). Not positioned according to the top-left corner!

`new GLabel("your text here");`

Same as above, but defaults (x, y) to (0, 0).



GLabel Methods

Methods specific to the `GLabel` class:

***label*`.getDescent();`** Returns the height of the label below its baseline.
***label*`.getAscent();`** Returns the height of the label above its baseline.
***label*`.setFont(`*`font`*`);`** Sets the font used to display the label as specified by the font string.

The font is typically specified as a string in the form:

"family-style-size"

family: is the name of a font family

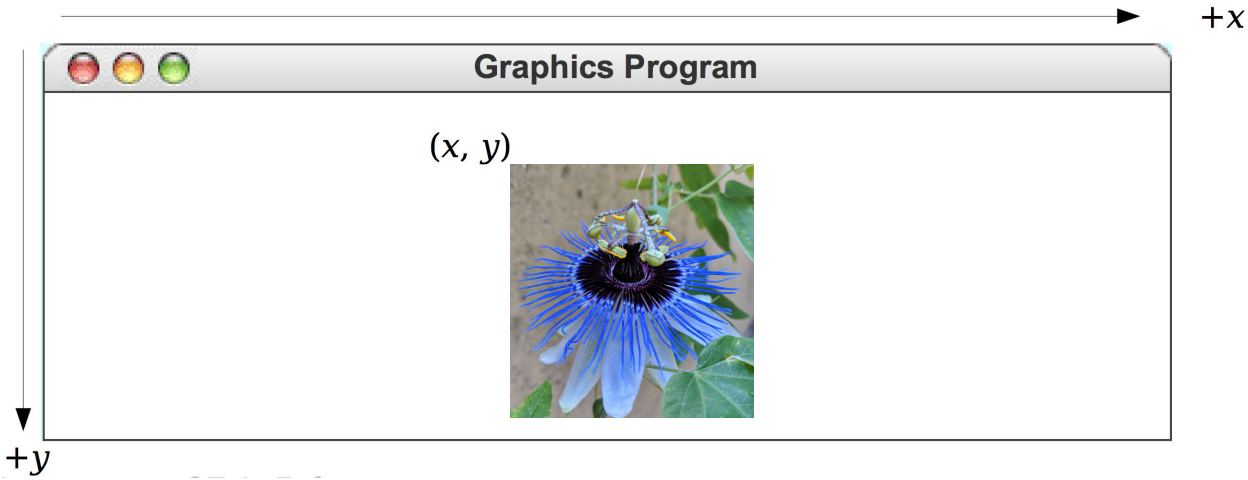
style: is either `PLAIN`, `BOLD`, `ITALIC`, or `BOLDITALIC`

size: is an integer indicating the point size

GImage

```
new GImage("filename", x, y);
```
Creates an image displaying the given file, whose upper-left corner is at (x, y).

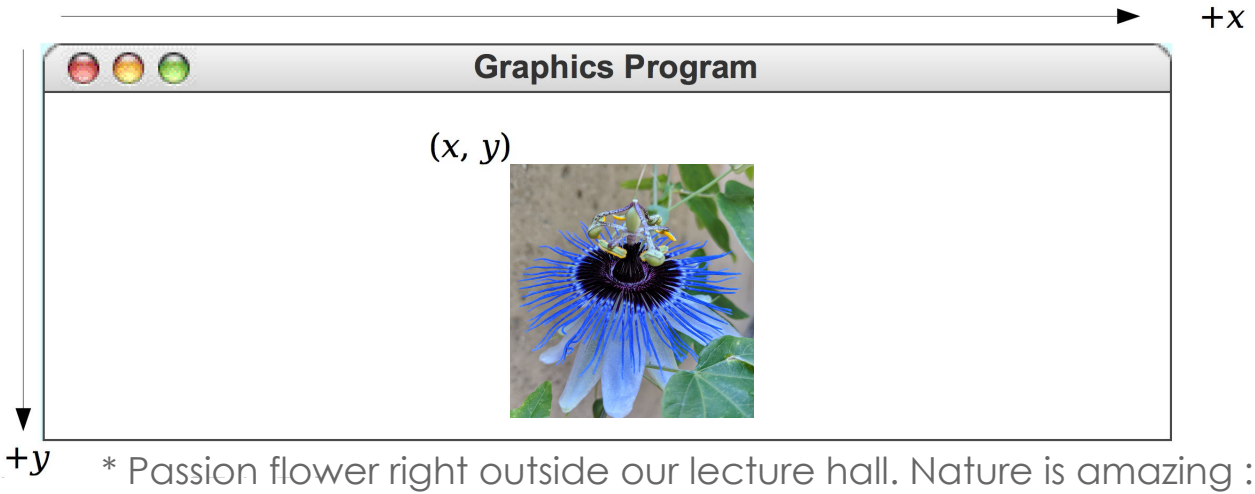
```
new GImage("filename");
```
Same as above, but defaults (x, y) to (0, 0).



GImage

```
new GImage("filename", x, y);
```
Creates an image displaying the given file, whose upper-left corner is at (x, y).

```
new GImage("filename");
```
Same as above, but defaults (x, y) to (0, 0).



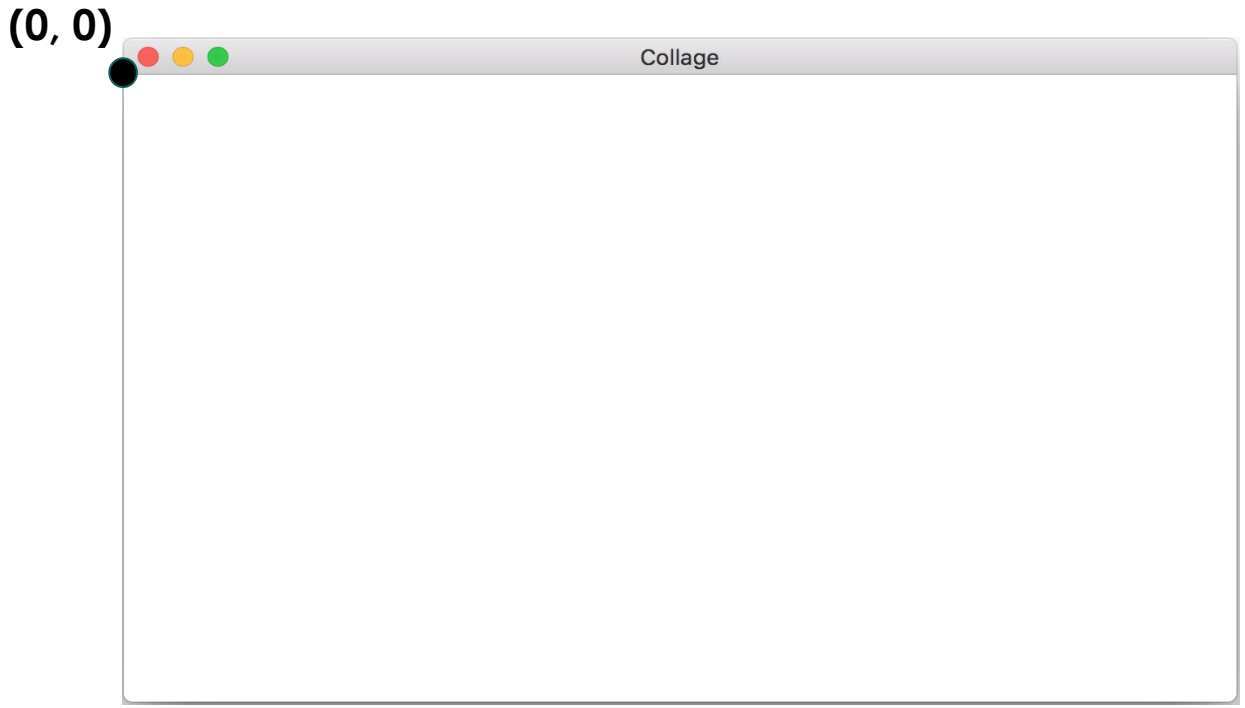
* Passion flower right outside our lecture hall. Nature is amazing :)

GImage Methods

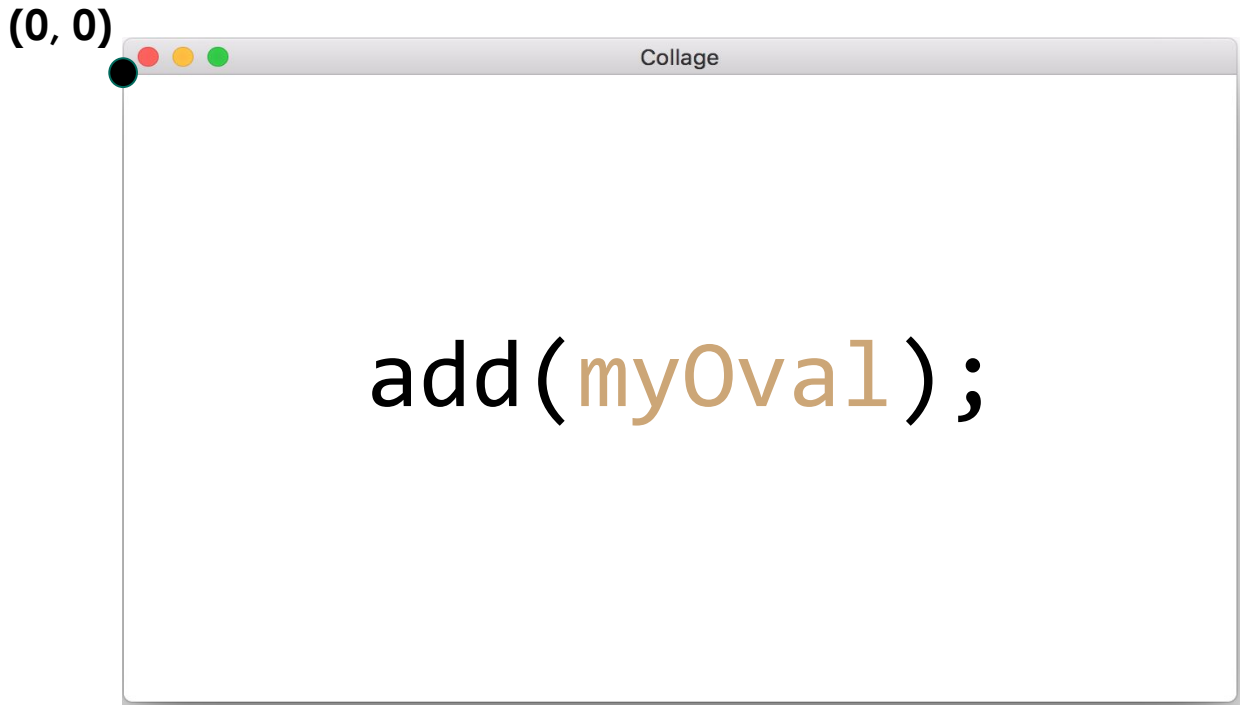
object.**setSize**(*width*, *height*);

Sets the object's size to be the given width and height.

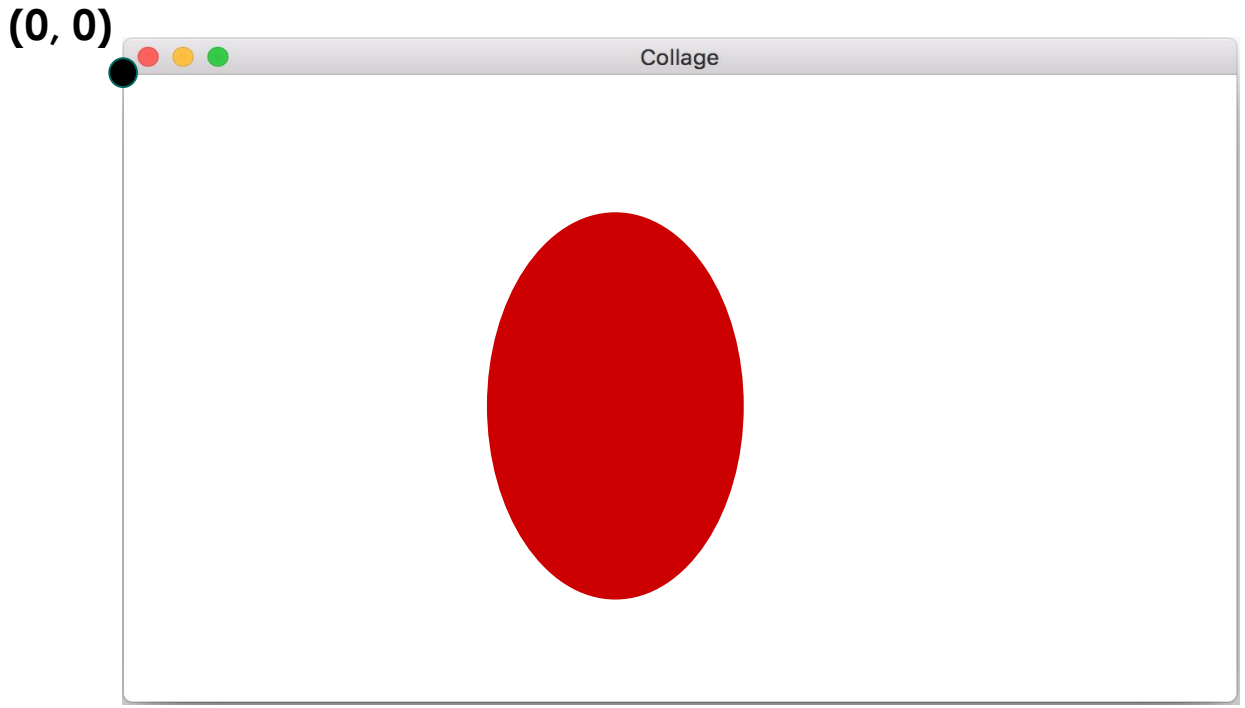
The Collage Model



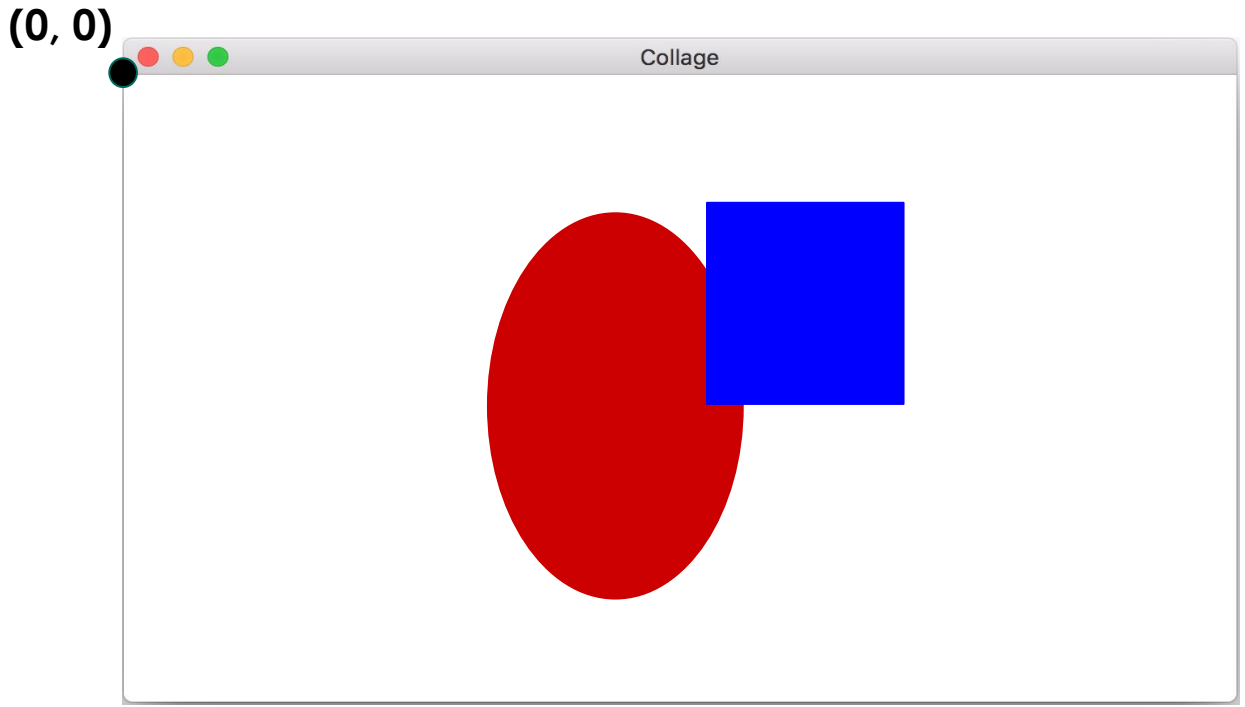
The Collage Model



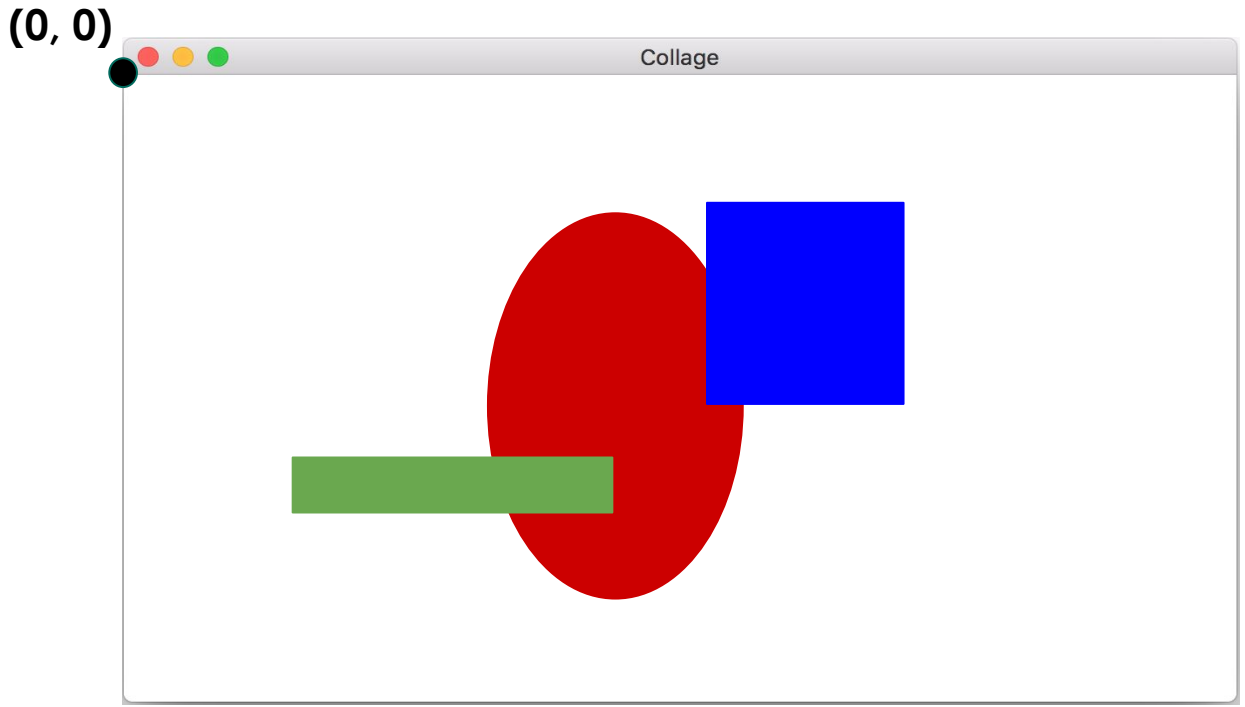
The Collage Model



The Collage Model



The Collage Model



GraphicsProgram Methods

`GraphicsProgram` contains these useful methods:

Method	Description
`add(gobj);` `add(gobj, x, y);`	adds a graphical object to the window
`getElementAt(x, y)`	return the object at the given (x,y) position(s)
`getElementCount()`	return number of graphical objects onscreen
`getWidth(), getHeight()`	return dimensions of window
`remove(gobj);`	removes a graphical object from the window
`removeAll();`	remove all graphical objects from window
`setCanvasSize(w, h);`	set size of drawing area
`setBackground(color);`	set window's background color

GraphicsProgram Methods

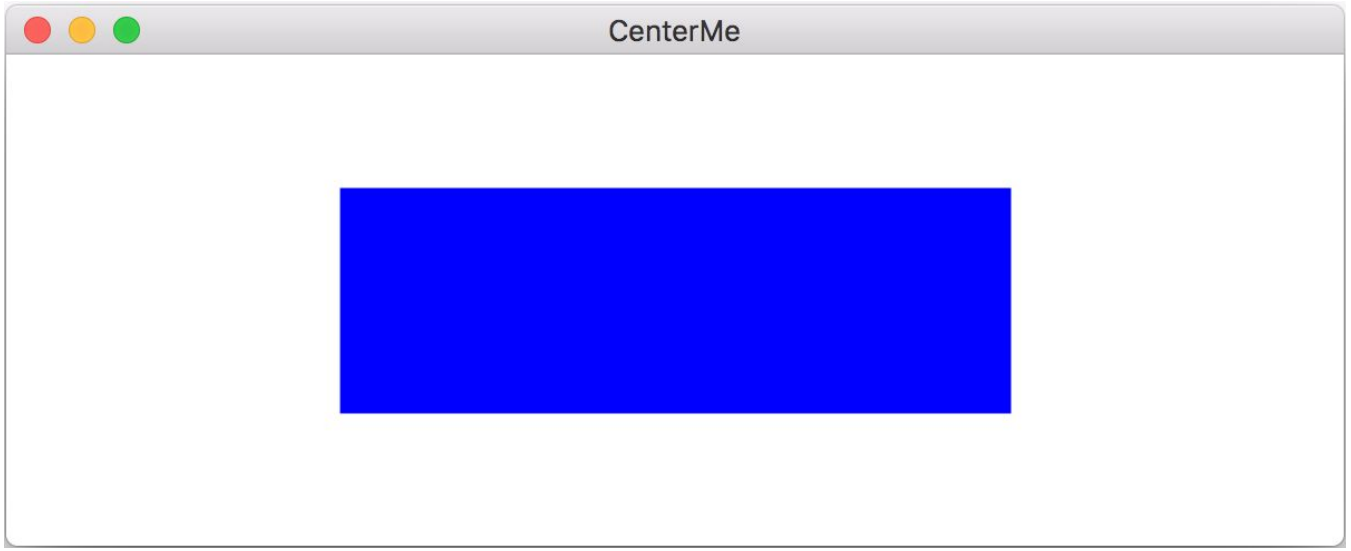
`GraphicsProgram` contains these useful methods:

Method	Description
`add(gobj);` `add(gobj, x, y);`	adds a graphical object to the window
`getElementAt(x, y)`	return the object at the given (x,y) position(s)
`getElementCount()`	return number of graphical objects onscreen
`getWidth(), getHeight()`	return dimensions of window
`remove(gobj);`	removes a graphical object from the window
`removeAll();`	remove all graphical objects from window
`setCanvasSize(w, h);`	set size of drawing area
`setBackground(color);`	set window's background color

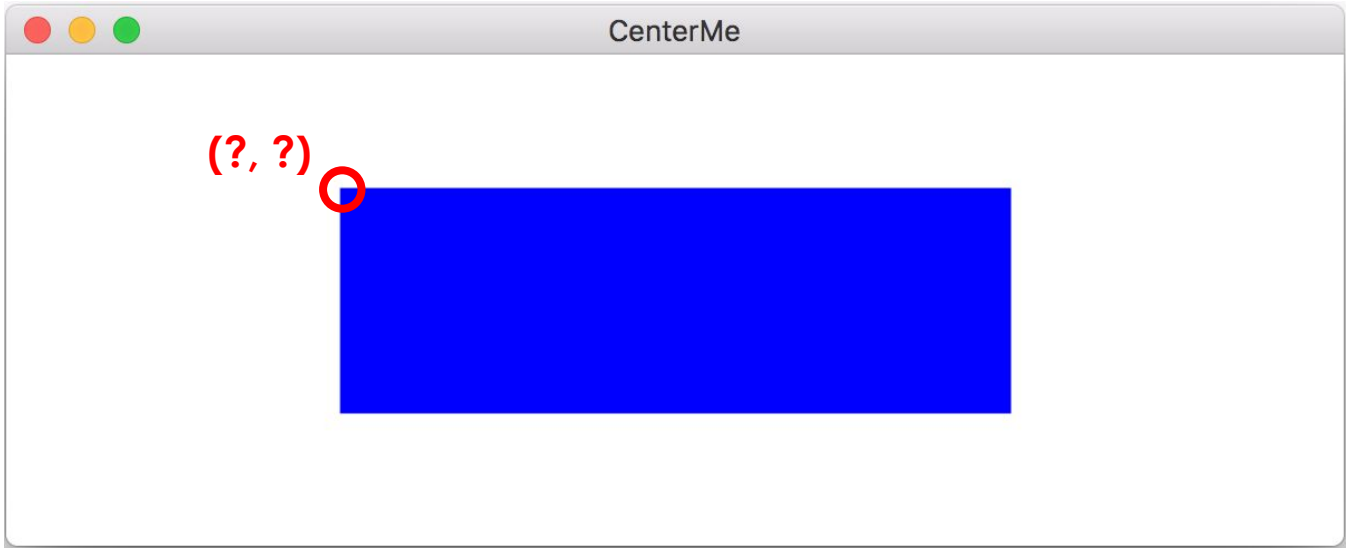
Practice: Centering

How can we nicely center an object on our canvas?

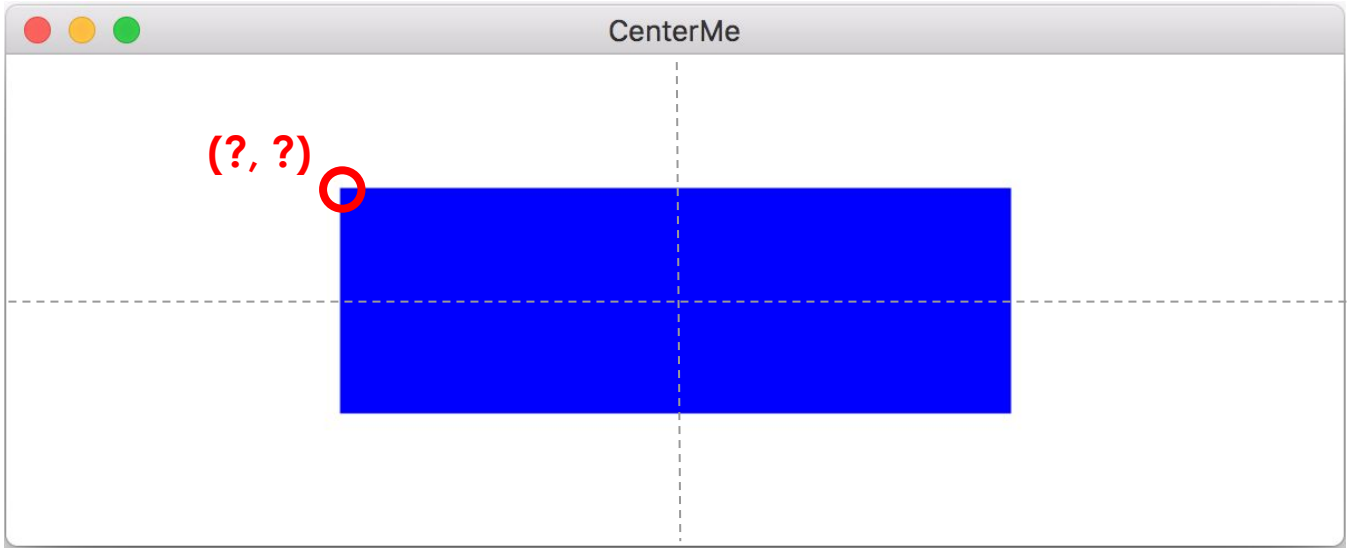
Practice: Centering



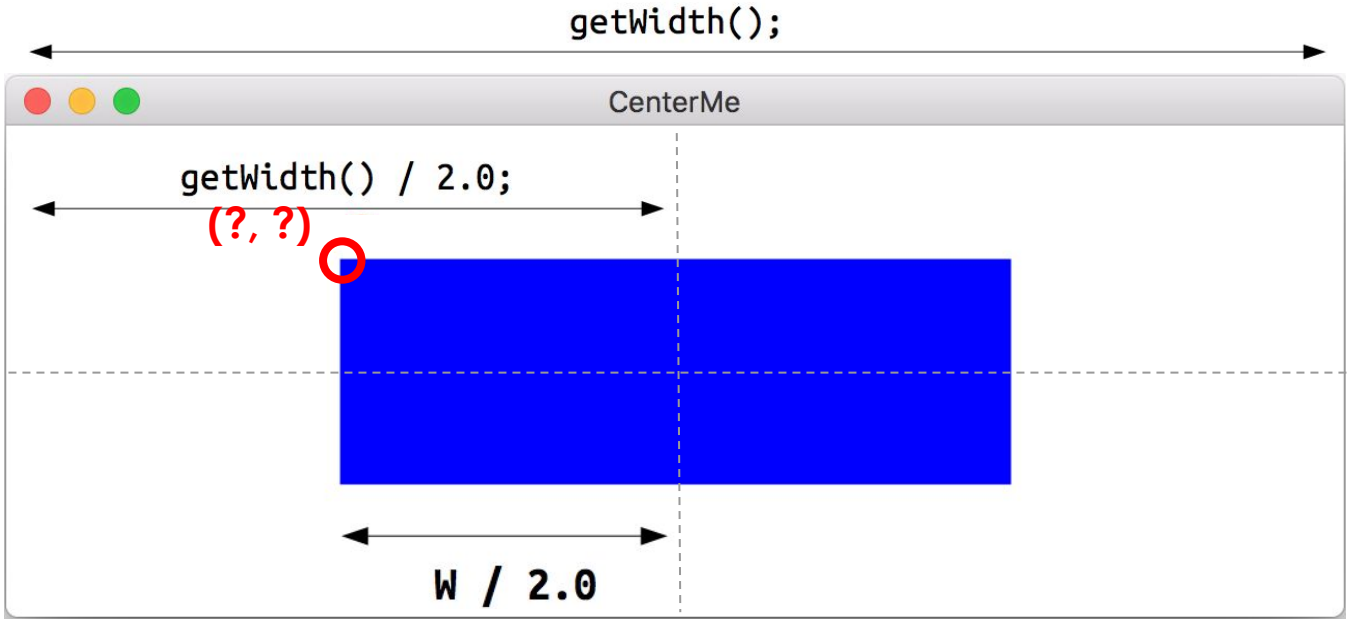
Practice: Centering



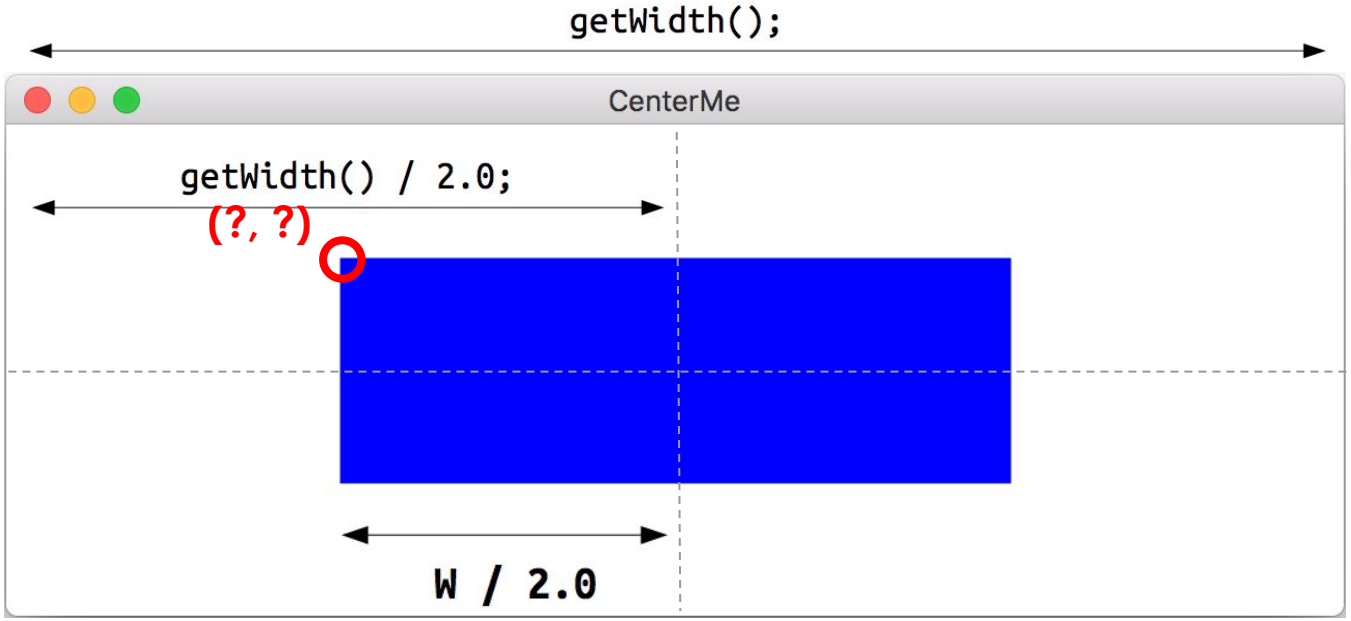
Practice: Centering



Practice: Centering

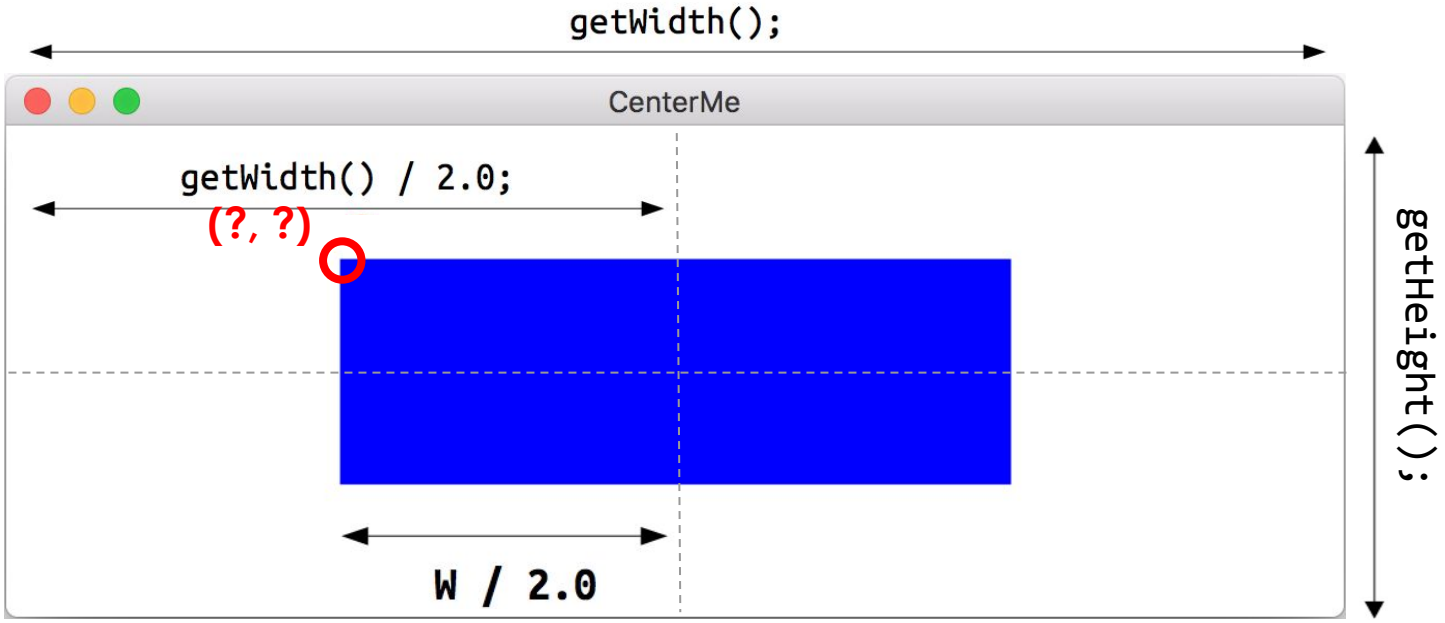


Practice: Centering



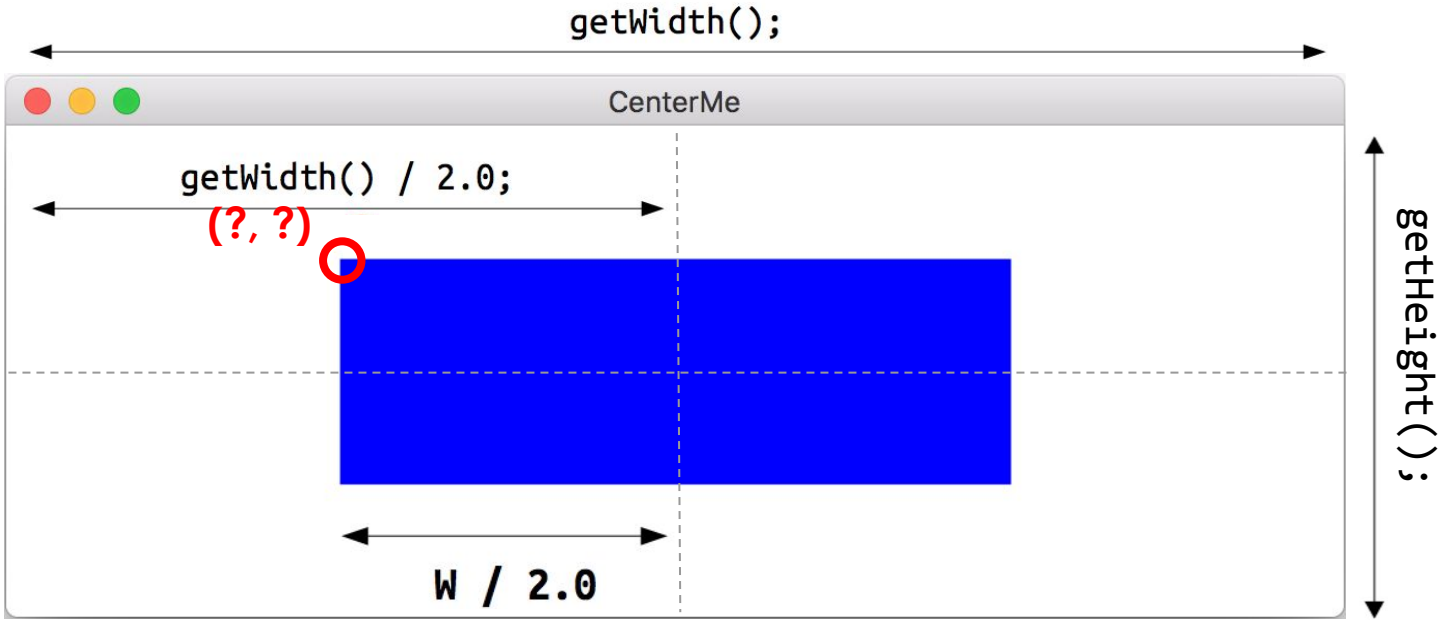

```
double x = getWidth() / 2.0 - W / 2.0;
```

Practice: Centering




```
double x = getWidth() / 2.0 - W / 2.0;
```

Practice: Centering

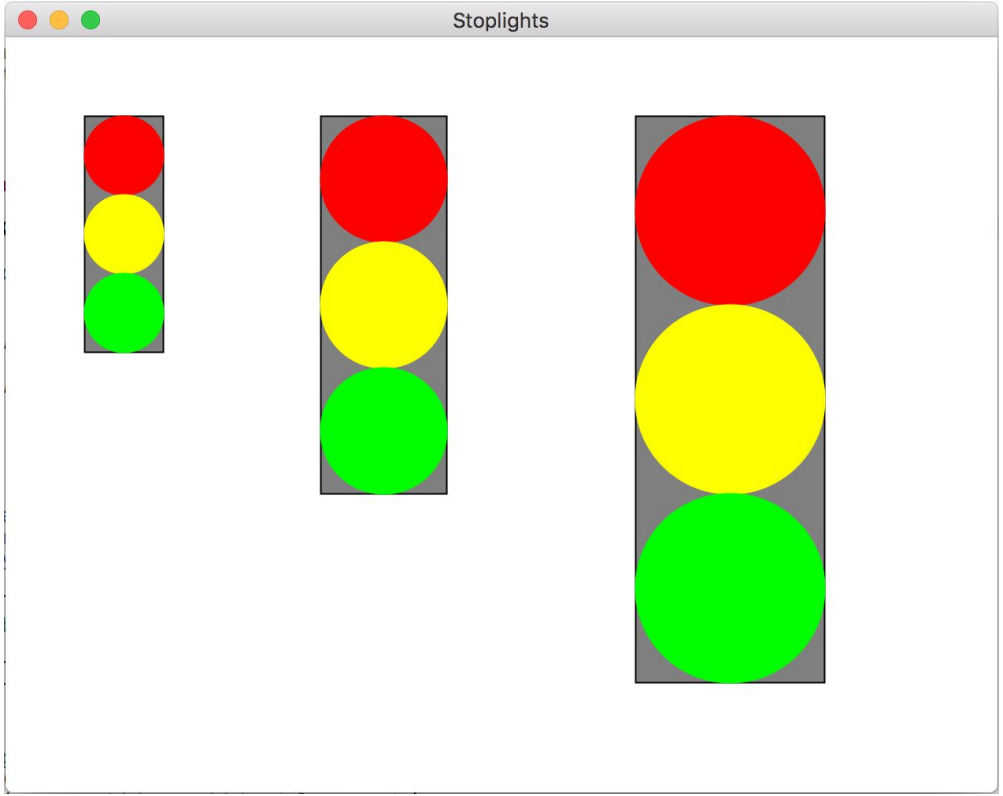



```
double x = getWidth() / 2.0 - W / 2.0;
double y = getHeight() / 2.0 - H / 2.0;
```

Plan for Today

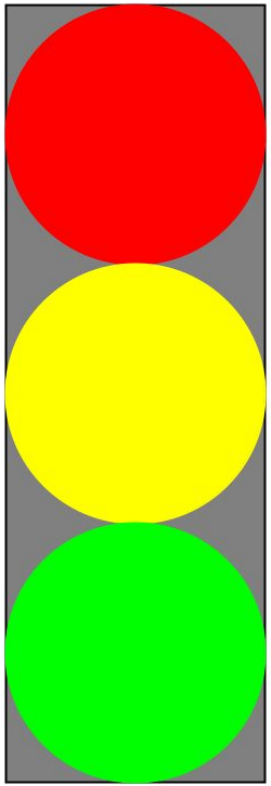
- Review: Nested Loops, Infinite Loops, Intro to Graphics
- Returning to Returns
- More Graphics!
- Stoplights

Practice: Stoplights



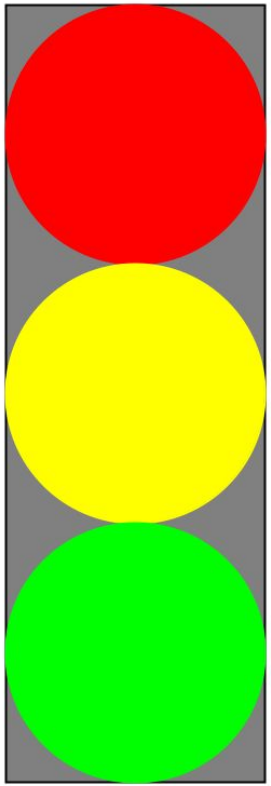
How would you make a **method** for drawing stoplights of different locations & sizes?

Practice: Stoplights



What information do we need in order to draw this?

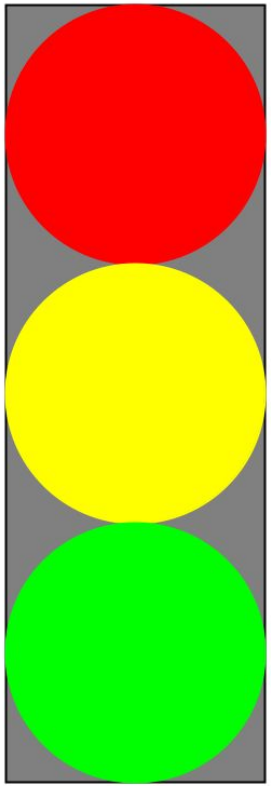
Practice: Stoplights



What's the Pseudocode?

Make stoplight (needs what info??):

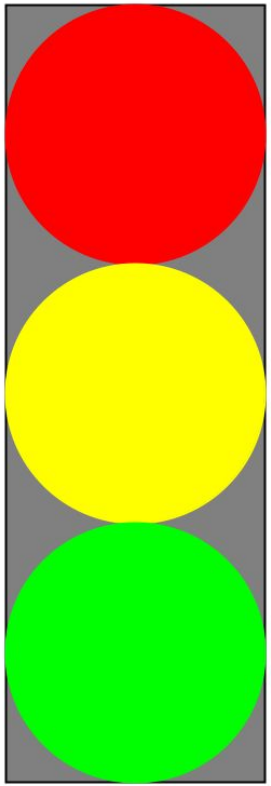
Practice: Stoplights



What's the Pseudocode?

Make stoplight (location, size):

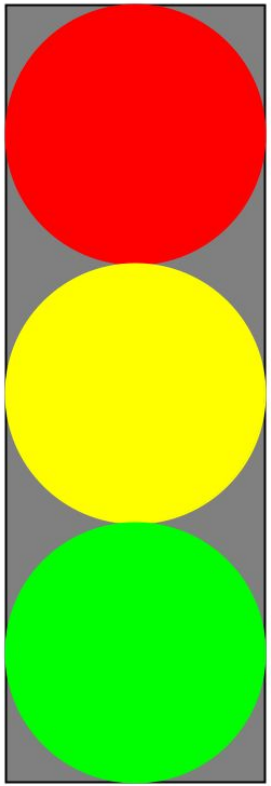
Practice: Stoplights



What's the Pseudocode?

Make stoplight (location, size):
    Make background
    Make red circle
    Make yellow circle
    Make green circle

Practice: Stoplights

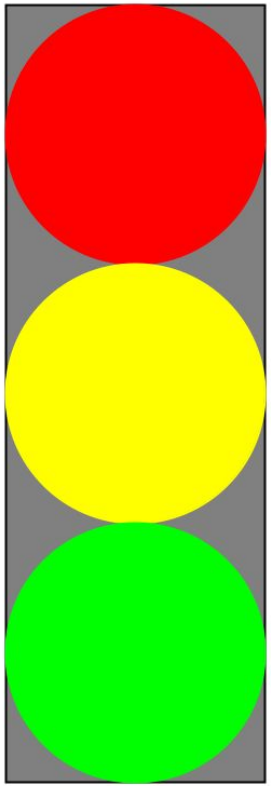


What's the Pseudocode?

Make stoplight (location, size):

- Make background
- **Make red circle**
- **Make yellow circle**
- **Make green circle**

Practice: Stoplights



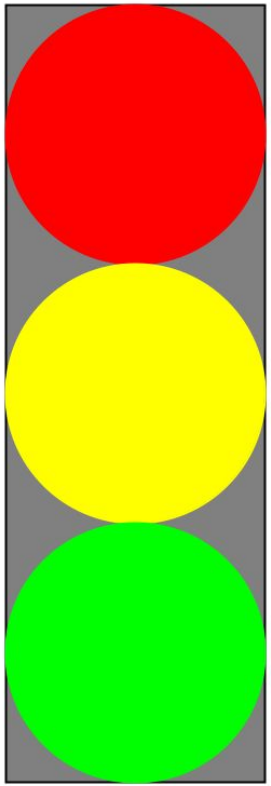
A helper method!

What's the Pseudocode?

Make stoplight (location, size):

- Make background
- **Make colored circle (red)**
- **Make colored circle (yellow)**
- **Make colored circle (green)**

Practice: Stoplights

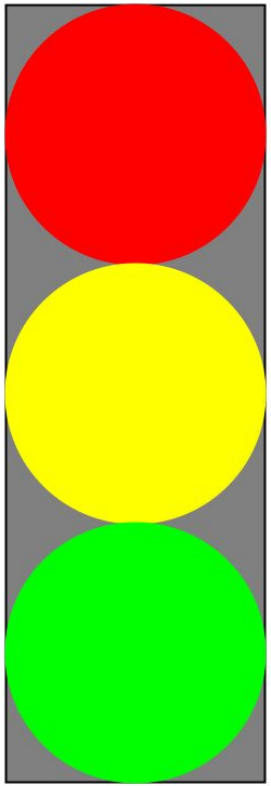


What's the Pseudocode?

Make stoplight (location, size):
    Make background
    Make colored circle (red)
    Make colored circle (yellow)
    Make colored circle (green)

Make colored circle (needs what info?):

Practice: Stoplights

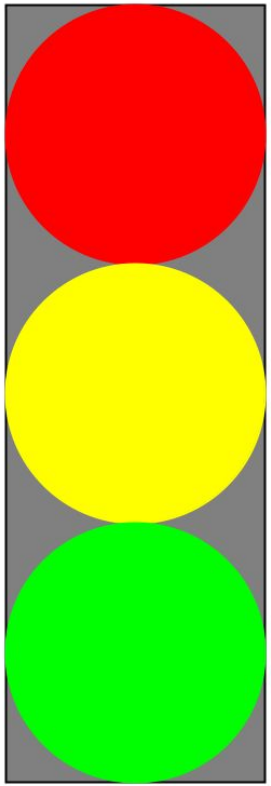


What's the Pseudocode?

Make stoplight (location, size):
    Make background
    Make colored circle (red)
    Make colored circle (yellow)
    Make colored circle (green)

Make colored circle (location, size, color):

Practice: Stoplights



What's the Pseudocode?

Make stoplight (location, size):
    Make background
    Make colored circle (red)
    Make colored circle (yellow)
    Make colored circle (green)

Make colored circle (location, size, color):
    Make circle of provided size and color
    and add it at the provided location

Let’s Code It!

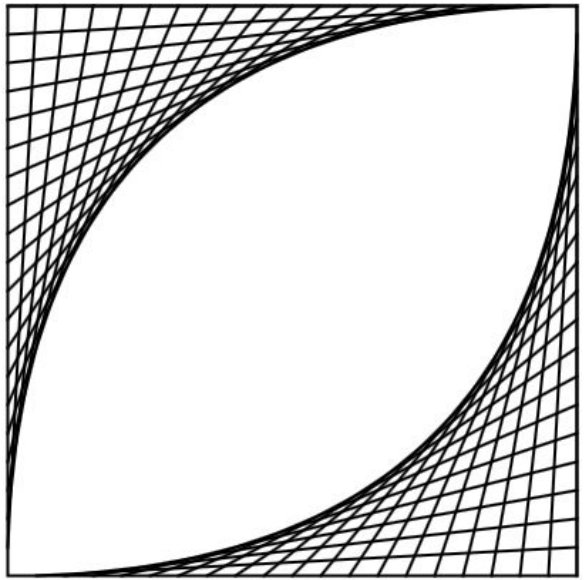
Extra Practice: Line Art

Write a graphical program LineArt that draws a series of lines (see lecture code for solution):

- Outer square is at (10, 30) and size 200x200
- Each line is 10px apart in each direction

coordinates of top-left lines:
- (210, 30) to (10, 30)
- (200, 30) to (10, 40)
- (190, 30) to (10, 50)
- ...
- (20, 30) to (10, 220)

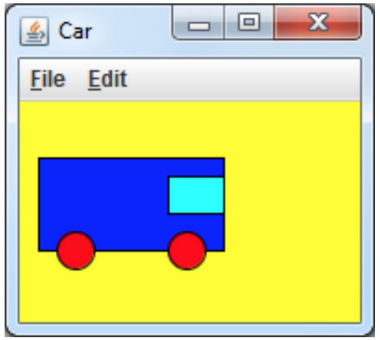
coordinates of bottom-right lines:
- (210, 30) to (210, 230)
- (210, 40) to (200, 230)
- ...
- (210, 220) to (20, 230)



Extra: GCompound

A `GCompound` contains other `GObject`s. It's useful when you want to do one operation on multiple `GObject`s at the same time (e.g., move all of them by the same amount).

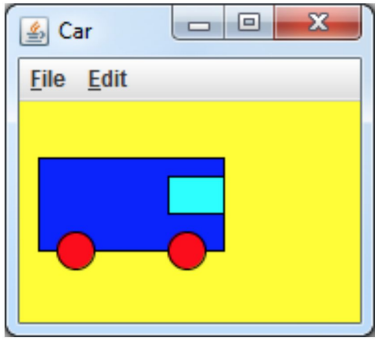
```
GCompound compound = new GCompound();
compound.add(shape);
compound.add(shape);
...
compound.add(shape);
add(compound);
```



Example: you can make a `GCompound` to represent a car.

Extra: GCompound

```
setBackground(Color.YELLOW);
GCompound car = new GCompound();

GRect body = new GRect(10, 30, 100, 50);
body.setFilled(true);
body.setFillColor(Color.BLUE);
car.add(body);

GOval wheel1 = new GOval(20, 70, 20, 20);
wheel1.setFilled(true);
wheel1.setFillColor(Color.RED);
car.add(wheel1);

GOval wheel2 = new GOval(80, 70, 20, 20);
wheel2.setFilled(true);
wheel2.setFillColor(Color.RED);
car.add(wheel2);

GRect windshield = new GRect(80, 40, 30, 20);
windshield.setFilled(true);
windshield.setFillColor(Color.CYAN);
car.add(windshield);

add(car);   // adds whole GCompound (like a "sub-canvas") to the canvas
```



Plan for Today

- Review: Nested Loops, Infinite Loops, Intro to Graphics
- Returning to Returns
- More Graphics!
- Stoplights

Extra: Line Art practice & GCompound

Next time: Animation!