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

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

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

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

```java
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

Program

- Karel Program
- Console Program
- Graphics Program
Review: Intro to Graphics Programs
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);
    }
}
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

```java
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

```java
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

```java
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;
}
```

average = 7.5
Multiple Return Statements

- You can have multiple return statements in a method
Multiple Return Statements

- You can have multiple return statements in a method

```java
private int max(int num1, int num2) {
    if (num1 >= num2) {
        return num1;
    }
    return num2;
}
```
public void run() {
    int larger = max(5, 1);
}

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

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

private int max(int num1, int num2) {
    if (num1 >= num2) {
        return num1;
    }
    return num2;
}
```
public void run() {
    int larger = max(5, 1);
}

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

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

private int max(int num1, int num2) {
    if (num1 >= num2) {
        return num1;
    }
    return num2;
}
```
public void run() {
    int larger = max(5, 1);
}

private int max(int num1, int num2) {
    if (num1 >= num2) {
        return num1;
    }
    return num2;
}
public void run() {
    int larger = max(5, 1);
}

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

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

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

Run memory

| No variables |

Max memory

<table>
<thead>
<tr>
<th>5</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>num1</td>
<td>num2</td>
</tr>
</tbody>
</table>
public void run() {
    int larger = max(5, 1);  // 5
}

private int max(int num1, int num2) {
    if (num1 >= num2) {
        return num1;
    }
    return num2;
}
public void run() {
    int larger = max(5, 1); 5
}

private int max(int num1, int num2) {
    if (num1 >= num2) {
        return num1;
    }
    return num2;
}
public void run() {
    int larger = max(5, 1);
}

private int max(int num1, int num2) {
    if (num1 >= num2) {
        return num1;
    }
    return num2;
}
public void run() {
    int larger = max(5, 1);
}

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

You are here
- Graphics Programs
- Animation
We make graphics programs by creating graphics objects and manipulating their properties.
We make graphics programs by creating *graphics objects* and manipulating their properties.
Graphical Objects

- GRect
- GOval
- GLine
- GLLabel
- GImage
- GRoundRect
- GArc
- GPolygon
- GCompound
- and more...
Graphical Objects

- GRect
- GOval
- GLine
- GLabel
- GImage
- GRoundRect
- GArc
- GPolygon
- GCompound
- and more...

Hello
Graphical Objects

- GLabel
- GRect
- GOval
- others...
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:
  
  \[
  \text{type name} = \text{new type(...);} \\
  \]

- Example:

  \[
  \text{GRect rect} = \text{new GRect(100, 200);} \\
  \]
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:
  
  \[
  \text{type name} = \text{new type(...);} \\
  \]

● Example:
  
  \[
  \text{GRect rect} = \text{new GRect(100, 200);} \\
  \]

This is called a “constructor”
### Primitives vs. Objects

<table>
<thead>
<tr>
<th>Primitive Variable Types</th>
<th>Object Variable Types</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>int</code></td>
<td><code>GRect</code></td>
</tr>
<tr>
<td><code>double</code></td>
<td><code>GOval</code></td>
</tr>
<tr>
<td><code>char</code></td>
<td><code>GLine</code></td>
</tr>
<tr>
<td><code>boolean</code></td>
<td><code>...</code></td>
</tr>
</tbody>
</table>
## Primitives vs. Objects

<table>
<thead>
<tr>
<th>Primitive Variable Types</th>
<th>Object Variable Types</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>int</code></td>
<td><code>GRect</code></td>
</tr>
<tr>
<td><code>double</code></td>
<td><code>GOval</code></td>
</tr>
<tr>
<td><code>char</code></td>
<td><code>GLine</code></td>
</tr>
<tr>
<td><code>boolean</code></td>
<td>...</td>
</tr>
</tbody>
</table>

**Object variables:**
# Primitives vs. Objects

## Primitive Variable Types
- int
- double
- char
- boolean

## Object Variable Types
- GRect
- GOval
- GLine

### Object variables:
1. Have UpperCamelCase types
## Primitives vs. Objects

<table>
<thead>
<tr>
<th>Primitive Variable Types</th>
<th>Object Variable Types</th>
</tr>
</thead>
<tbody>
<tr>
<td>int</td>
<td>GRect</td>
</tr>
<tr>
<td>double</td>
<td>GOval</td>
</tr>
<tr>
<td>char</td>
<td>GLine</td>
</tr>
<tr>
<td>boolean</td>
<td>...</td>
</tr>
</tbody>
</table>

**Object variables:**

1. Have UpperCamelCase types
2. Are constructed using `new`
## Primitives vs. Objects

<table>
<thead>
<tr>
<th>Primitive Variable Types</th>
<th>Object Variable Types</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>int</code></td>
<td><code>GRect</code></td>
</tr>
<tr>
<td><code>double</code></td>
<td><code>GOval</code></td>
</tr>
<tr>
<td><code>char</code></td>
<td><code>GLine</code></td>
</tr>
<tr>
<td><code>boolean</code></td>
<td><code>...</code></td>
</tr>
</tbody>
</table>

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

  \[ \text{object.method(parameters);} \]

- Example:

  \[ \text{rect.setColor(Color.BLUE);} \]
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:

  \[ \text{object.method(parameters)}; \]

- Example:

  \[ \text{rect.setColor(Color.BLUE);} \]
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:

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

- Example:

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

who?
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:
  
  \[
  \text{object.method(parameters);} 
  \]

- Example:
  
  \[
  \text{rect.setColor(Color.BLUE);} 
  \]
The following operations apply to all GObject:

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>object.setColor(color)</code></td>
<td>Sets the color of the object to the specified color constant.</td>
</tr>
<tr>
<td><code>object.setLocation(x, y)</code></td>
<td>Changes the location of the object to the point (x, y).</td>
</tr>
<tr>
<td><code>object.move(dx, dy)</code></td>
<td>Moves the object on the screen by adding dx and dy to its current coordinates.</td>
</tr>
<tr>
<td><code>object.getWidth()</code></td>
<td>Returns the width of the object</td>
</tr>
<tr>
<td><code>object.getHeight()</code></td>
<td>Returns the height of the object</td>
</tr>
</tbody>
</table>

and more...
Specified as predefined Color constants:

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

<table>
<thead>
<tr>
<th>BLACK</th>
<th>BLUE</th>
<th>CYAN</th>
<th>DARK_GRAY</th>
<th>GRAY</th>
</tr>
</thead>
<tbody>
<tr>
<td>GREEN</td>
<td>LIGHT_GRAY</td>
<td>MAGENTA</td>
<td>ORANGE</td>
<td>PINK</td>
</tr>
<tr>
<td>RED</td>
<td>WHITE</td>
<td>YELLOW</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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

- Specified as predefined Color constants: `Color.NAME`, where `NAME` is one of:

<table>
<thead>
<tr>
<th>BLACK</th>
<th>BLUE</th>
<th>CYAN</th>
<th>DARK_GRAY</th>
<th>GRAY</th>
</tr>
</thead>
<tbody>
<tr>
<td>GREEN</td>
<td>LIGHT_GRAY</td>
<td>MAGENTA</td>
<td>ORANGE</td>
<td>PINK</td>
</tr>
<tr>
<td>RED</td>
<td>WHITE</td>
<td>YELLOW</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

```
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  GImage

GRoundRect  GArc  GPolygon  GCompound  and more ...

Hello
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).
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).
Methods shared by the GRect and GOval classes:

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>object.setFilled(fill);</code></td>
<td>If <code>fill</code> is <code>true</code>, fills in the interior of the object; if <code>false</code>, shows only the outline.</td>
</tr>
<tr>
<td><code>object.setFillColor(color);</code></td>
<td>Sets the color used to fill the interior, which can be different from the border.</td>
</tr>
<tr>
<td><code>object.setSize(width, height);</code></td>
<td>Sets the object’s size to be the given width and height.</td>
</tr>
</tbody>
</table>
new GLine(x0, y0, x1, y1);
Creates a line extending from \((x_0, y_0)\) to \((x_1, y_1)\).
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).
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).
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).

Hi everyone!

![Diagram of GLabel with baseline, height, and x and y axes]
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).
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
new GImage("filename", x, y);
Create 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).

![Graphics Program](image)
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 :)

Graphics Program

(x, y)
object.setSize(width, height);
Sets the object's size to be the given width and height.
The Collage Model
add(myOval);
The Collage Model
The Collage Model
The Collage Model
GraphicsProgram contains these useful methods:

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>add(gobj);</td>
<td>adds a graphical object to the window</td>
</tr>
<tr>
<td>add(gobj, x, y);</td>
<td></td>
</tr>
<tr>
<td>getElementAt(x, y)</td>
<td>return the object at the given (x,y) position(s)</td>
</tr>
<tr>
<td>getElementCount()</td>
<td>return number of graphical objects onscreen</td>
</tr>
<tr>
<td>getWidth(), getHeight()</td>
<td>return dimensions of window</td>
</tr>
<tr>
<td>remove(gobj);</td>
<td>removes a graphical object from the window</td>
</tr>
<tr>
<td>removeAll();</td>
<td>remove all graphical objects from window</td>
</tr>
<tr>
<td>setCanvasSize(w, h);</td>
<td>set size of drawing area</td>
</tr>
<tr>
<td>setBackground(color);</td>
<td>set window's background color</td>
</tr>
</tbody>
</table>
GraphicsProgram contains these useful methods:

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>add($obj$);</td>
<td>adds a graphical object to the window</td>
</tr>
<tr>
<td>add($obj$, $x$, $y$);</td>
<td></td>
</tr>
<tr>
<td>getElementAt($x$, $y$)</td>
<td>return the object at the given (x,y) position(s)</td>
</tr>
<tr>
<td>getElementCount()</td>
<td>return number of graphical objects onscreen</td>
</tr>
<tr>
<td>getWidth(), getHeight()</td>
<td>return dimensions of window</td>
</tr>
<tr>
<td>remove($obj$);</td>
<td>removes a graphical object from the window</td>
</tr>
<tr>
<td>removeAll()</td>
<td>remove all graphical objects from window</td>
</tr>
<tr>
<td>setCanvasSize($w$, $h$);</td>
<td>set size of drawing area</td>
</tr>
<tr>
<td>setBackground($color$);</td>
<td>set window's background color</td>
</tr>
</tbody>
</table>
How can we nicely center an object on our canvas?
Practice: Centering
Practice: Centering

(?, ?)
Practice: Centering

```javascript
getWidth();

getWidth() / 2.0;
```

![Diagram showing the centering mechanism](image)

```javascript
W / 2.0
```
double $x = \frac{\text{getWidth()} }{2.0} - \frac{W}{2.0};$
double x = getWidth() / 2.0 - W / 2.0;
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
How would you make a *method* for drawing stoplights of different locations & sizes?
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):
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

Hmm, these are almost the same… what can we do?
Practice: Stoplights

What’s the Pseudocode?

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

A helper method!
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?):
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)

cordinates of bottom-right lines:
- (210, 30) to (210, 230)
- (210, 40) to (200, 230)
- ...
- (210, 220) to (20, 230)
A GCompound contains other GObjects. It’s useful when you want to do one operation on multiple GObjects at the same time (e.g., move all of them by the same amount).

```java
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!