CS 106A, Lecture 12
More Graphics

reading:
Art & Science of Java, 9.4
Plan For Today

• Announcements
• Recap: Graphics
• GCompounds
• Getters
• Practice: Checkerboard
• Practice: Stoplights
Plan For Today

• Announcements
• Recap: Graphics
• GCompounds
• Getters
• Practice: Checkerboard
• Practice: Stoplights
Announcements: Docs

- Click the "**Stanford Library Docs**" link in the 106A website sidebar.
  - This site lists every kind of object in the Stanford libraries.
  - Click an object type on the left and see its behavior on the right.
  - These kinds of pages exist for Stanford libraries and standard Java.
Announcements: Midterm

• Midterm is next Monday 7/23 from 7PM-9PM in Hewlett 200
• You will need your own laptop
  – Email Annie right away if you need us to get you a loaner laptop
• Before the exam, you will need to download two things from the website’s midterm page (neither download is ready yet, but soon):
  – A program called BlueBook, which you will use to take the exam
  – An encrypted file that BlueBook will read to show you the exam
    • You will get the decryption password during the exam
• See the midterm page for practice exams and study strategies
  – All lectures through this Thursday are fair game for the exam
  – The first practice is last summer’s midterm—it is hard!
  – The second uses BlueBook but covers material we haven’t seen
Plan For Today

• Announcements
• Recap: Graphics
• GCompounds
• Getters
• Practice: Checkerboard
• Practice: Stoplights
The Graphics Canvas
Collage Model
Graphical Objects

**GRect**
(x, y)

**GOval**
(x, y)

**GLine**
(x₁, y₁)

**GLabel**
Hello there!

**GImage**

**GRoundRect**

**GLine**
(x₂, y₂)

**GPolygon**
Graphical Objects

Initialization syntax:

\[
\text{type name} = \text{new type}(\ldots);
\]

Example:

\[
\text{GRect rect} = \text{new GRect}(50, 50, 350, 270);
\]
## 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>Scanner</td>
</tr>
<tr>
<td></td>
<td>...</td>
</tr>
</tbody>
</table>

**Object variables:**
1. Have UpperCamelCase types
2. You can call methods on them
   - Uses “dot syntax”
3. Are constructed using `new`
Methods on Graphics Objects

We manipulate graphics objects by calling methods on them:

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

Who?  What?  What specifically?

Example:

```
rect.setColor(Color.RED);
```
**GOBJECT METHODS**

The following operations apply to all **GOBJECTS**:

<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…

*Graphic courtesy of Eric Roberts*
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 to \((x, y) = (0, 0)\)
new GOval(x, y, width, height);

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

new GOval(width, height);

- Same as above, but defaults to (x, y) = (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.setFill(false)</code></td>
<td>Shows only the outline.</td>
</tr>
<tr>
<td><code>object.setFill(true)</code></td>
<td>Fills in the interior of the object.</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 (x0, y0) to (x1, y1)
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 to (x, y) = (0, 0)
Methods specific to the **GLabel** class

- **`label.getAscent()`**
  - Returns the height of the label above its baseline.

- **`label.getDescent()`**
  - Returns the height of the label below 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

```plaintext
"family-style-size"
```

- `family` is the name of a font family (e.g. “SansSerif”)
- `style` is either **PLAIN**, **BOLD**, **ITALIC**, or **BOLDITALIC**
- `size` is an integer indicating the point size
new GImage("your filename here", x, y);
  – Creates a an image displaying the given file, whose upper-left corner is at (x, y)

new GImage("your filename here");
  – Same as above, but defaults to (x, y) = (0, 0)
**GImage Methods**

```java
object.setSize(width, height)
```

Sets the object's size to be the given width and height.
• GraphicsProgram contains these useful methods:

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

Graphics Program

(?, ?)
Recap Practice: Centering

rectangle's x value = \( \frac{\text{getWidth}()}{2.0} - \frac{W}{2.0} \)
Recap Practice: Centering

rectangle's y value = getHeight() / 2.0 - H / 2.0
Plan For Today

• Announcements
• Recap: Graphics
• GCompounds
• Getters
• Practice: Checkerboard
• Practice: Stoplights
A **GCompound** contains other GObjects. It’s useful when you want to do one operation on multiple GObjects at the same time.

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

– You can make a GCompound to represent a car.
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);          // at 0,0! Where we want this “sub-canvas” to go
Plan For Today

- Announcements
- Recap: Graphics
- GCompounds
- Getters
- Practice: Checkerboard
- Practice: Stoplights
• Methods of graphical objects that return values:

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>obj.getColor()</code></td>
<td>the color used to color the shape outline</td>
</tr>
<tr>
<td><code>obj.getFillColor()</code></td>
<td>the color used to color the shape interior</td>
</tr>
<tr>
<td><code>obj.getX()</code></td>
<td>the left x-coordinate of the shape</td>
</tr>
<tr>
<td><code>obj.getY()</code></td>
<td>the top y-coordinate of the shape</td>
</tr>
<tr>
<td><code>obj.getWidth()</code></td>
<td>number of pixels wide the shape is</td>
</tr>
<tr>
<td><code>obj.getHeight()</code></td>
<td>number of pixels tall the shape is</td>
</tr>
</tbody>
</table>

- Example: Swapping the x/y coordinates of a shape:

  ```java
  GRect rect = new GRect(...);
  ...
  int rx = rect.getX();
  int ry = rect.getY();
  rect.setLocation(ry, rx);
  ```
Plan For Today

• Announcements
• Recap: Graphics
• GCompounds
• Getters

• Practice: Checkerboard
• Practice: Stoplights
Write a graphical program named **Checkerboard** that draws a checkerboard pattern using GRects.
Milestone 1
Milestone 2
Milestone 3
Milestone 3

- Notice the pattern if we add the row and column indexes...
Plan For Today

• Announcements
• Recap: Graphics
• GCompounds
• Getters
• Practice: Checkerboard
• Practice: Stoplights
Practice: Stoplights

How would you make a method for drawing stoplights of different locations and sizes?
Practice: Stoplights

What information do we need in order to draw this?
Recap

• Announcements
• Recap: Graphics
• GCompounds
• Getters
• Practice: Stoplights
• Practice: Checkerboard

Next time: Animation
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 dimension

**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)