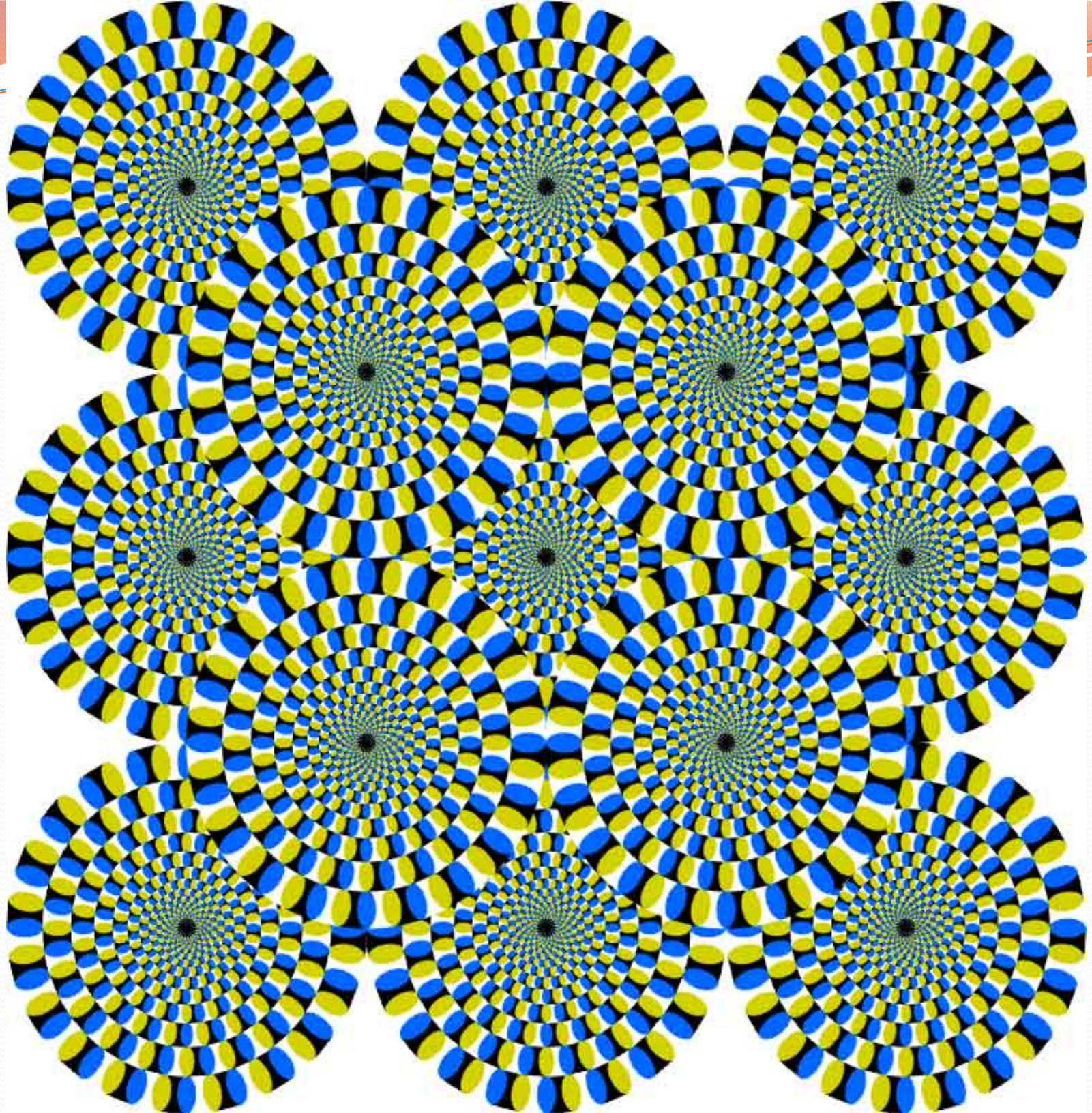
Building Java Programs

Chapter 3

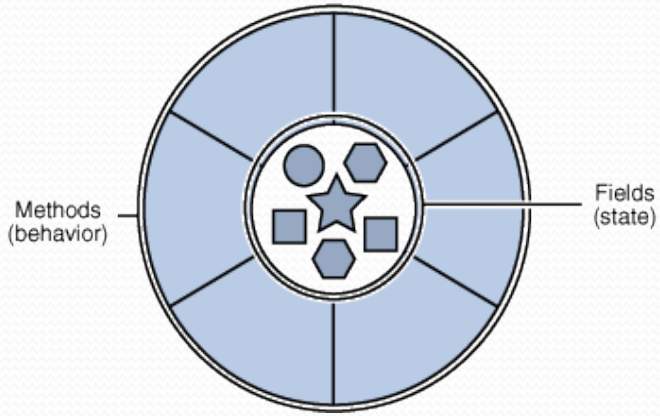
Lecture 6: Graphics

Reading: Supplement 3G



Objects (briefly)

- **object:** An entity that contains data and behavior.
  - *data*: variables inside the object
  - *behavior*: methods inside the object
    - You interact with the methods; the data is hidden in the object.
    - A **class** is a *type* of objects.



- Constructing (creating) an object:

```
Type objectName = new Type(parameters);
```

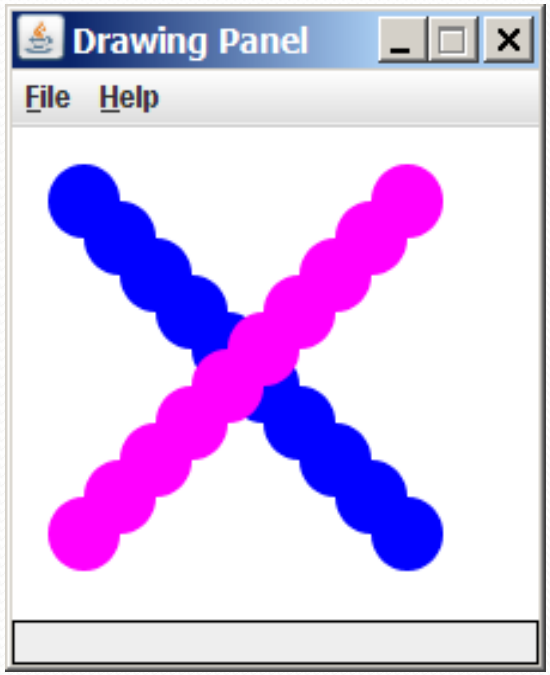
- Calling an object's method:

```
objectName.methodName(parameters);
```

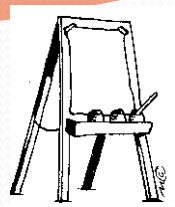
Graphical objects

We will draw graphics in Java using 3 kinds of objects:

- `DrawingPanel`: A window on the screen.
  - Not part of Java; provided by the authors. See class web site.
- `Graphics`: A "pen" to draw shapes and lines on a window.
- `Color`: Colors in which to draw shapes.



DrawingPanel



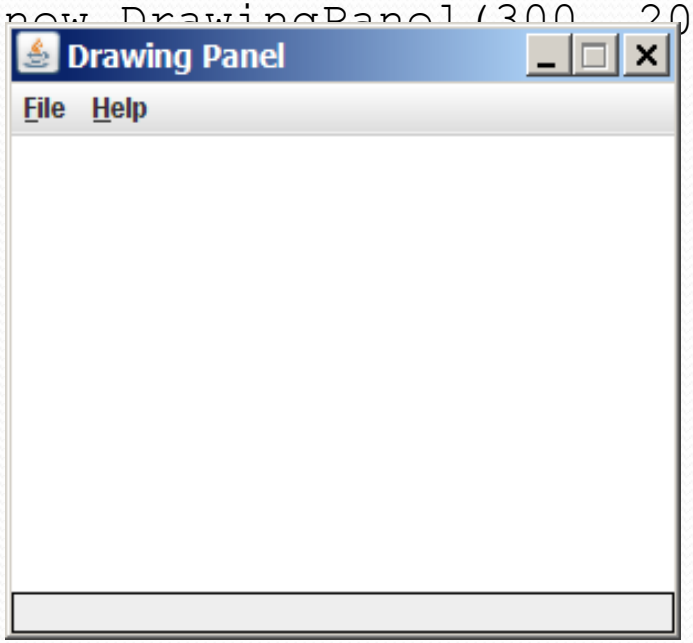
"Canvas" objects that represents windows/drawing surfaces

- To create a window:

```
DrawingPanel name = new DrawingPanel(width, height);
```

Example:

```
DrawingPanel panel = new DrawingPanel(300, 200);
```



Java class libraries, import

- **Java class libraries**: Classes included with the JDK (Java Development Kit).
  - organized into groups named *packages*
  - To use a package, put an *import declaration* in your program:

```
// put this at the very top of your program
import packageName.*;
```

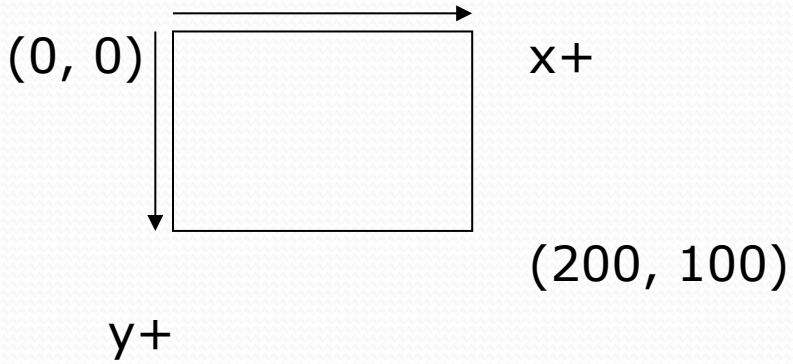
- `Graphics` belongs to a package named `java.awt`

```
import java.awt.*;
```

  - To use `Graphics`, you must place the above line at the very top of your program, before the `public class` header.

Coordinate system

- Each (x, y) position is a *pixel* ("picture element").
- Position (0, 0) is at the window's top-left corner.
  - x increases rightward and the y increases downward.
- The rectangle from (0, 0) to (200, 100) looks like this:



Graphics

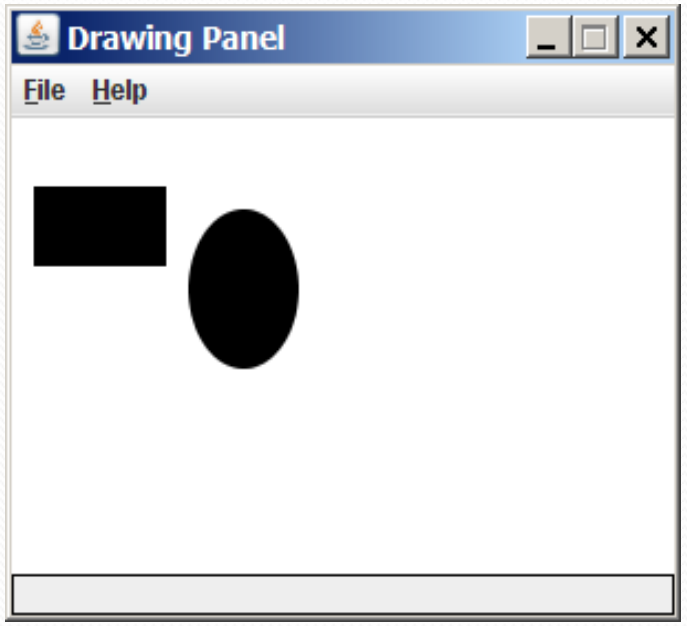
"Pen" or "paint brush" objects to draw lines and shapes

- Access it by calling `getGraphics` on your `DrawingPanel`.

```
Graphics g = panel.getGraphics();
```

- Draw shapes by calling methods on the `Graphics` object.

```
g.fillRect(10, 30, 60, 35);
g.fillOval(80, 40, 50, 70);
```



Graphics methods

Method name	Description
`g.drawLine(`*x1*, *y1*, *x2*, *y2*`);`	line between points (*x1*, *y1*), (*x2*, *y2*)
`g.drawOval(`*x*, *y*, *width*, *height*`);`	outline largest oval that fits in a box of size *width* * *height* with top-left at (*x*, *y*)
`g.drawRect(`*x*, *y*, *width*, *height*`);`	outline of rectangle of size *width* * *height* with top-left at (*x*, *y*)
`g.drawString(`*text*, *x*, *y*`);`	text with bottom-left at (x, y)
`g.fillOval(`*x*, *y*, *width*, *height*`);`	fill largest oval that fits in a box of size *width* * *height* with top-left at (*x*,*y*)
`g.fillRect(`*x*, *y*, *width*, *height*`);`	fill rectangle of size *width* * *height* with top-left at (*x*, *y*)
`g.setColor(`*Color*`);`	set `Graphics` to paint any following shapes in the given color

Color

- Specified as predefined `Color` class constants:

  `Color.`**`CONSTANT_NAME`**

  where **CONSTANT_NAME** is one of:

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

  - Example:

    ```
    Color.MAGENTA
    ```

Making your own colors

- Create colors using Red-Green-Blue (RGB) values of 0-255

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

  - Example:

```
Color brown = new Color(192, 128, 64);
```

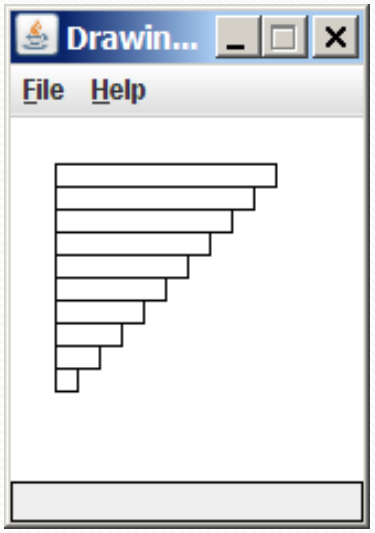
  - List of RGB colors: http://web.njit.edu/~kevin/rgb.txt.html

Drawing w/ loops questions

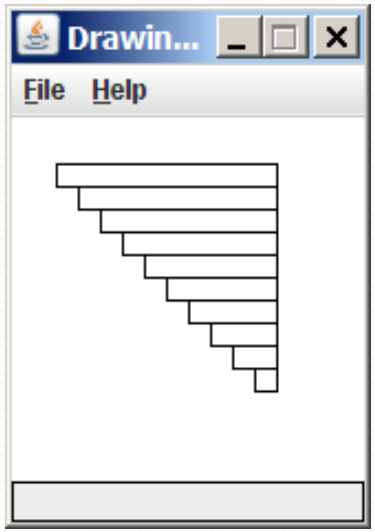
- Code from previous slide:

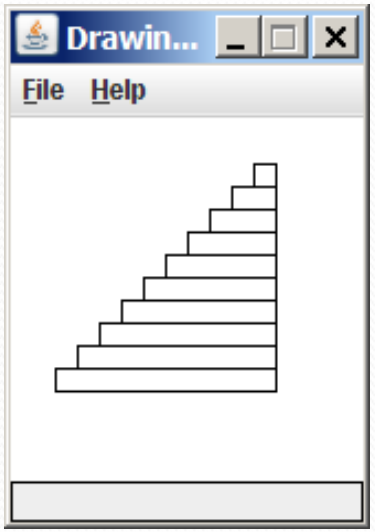
```
DrawingPanel panel = new DrawingPanel(160, 160);
Graphics g = panel.getGraphics();

for (int i = 0; i < 10; i++) {
    g.drawRect(20, 20 + 10 * i,
               100 - 10 * i, 10);
}
```



- Write variations of the above program that draw the figures at right as output.





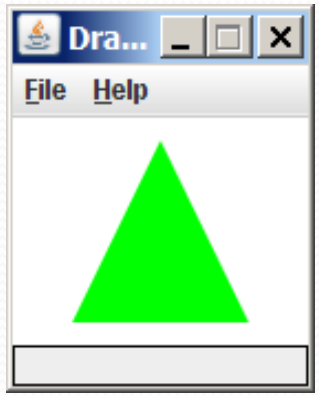
Polygon

Objects that represent arbitrary shapes

- Add points to a `Polygon` using its `addPoint(`***<x>***`, `***<y>***`)` method.
- Example:

```
DrawingPanel p = new DrawingPanel(100, 100);
Graphics g = p.getGraphics();
g.setColor(Color.GREEN);

Polygon poly = new Polygon();
poly.addPoint(10, 90);
poly.addPoint(50, 10);
poly.addPoint(90, 90);
g.fillPolygon(poly);
```



`DrawingPanel` methods

- **panel**`.clear();`
  Erases any shapes that are drawn on the drawing panel.

- **panel**`.setWidth(`**width**`);`
  panel`.setHeight(`**height**`);`
  panel`.setSize(`**width**`, `**height**`);`
  Changes the drawing panel's size to the given value(s).

- **panel**`.save(`**filename**`);`
  Saves the image on the panel to the given file (String).

- **panel**`.sleep(`**milliseconds**`);`
  Pauses the drawing for the given number of milliseconds.

Animation with `sleep`

- `DrawingPanel`'s `sleep` method pauses your program for a given number of milliseconds.

- You can use `sleep` to create simple animations.

```
DrawingPanel panel = new DrawingPanel(250, 200);
Graphics g = panel.getGraphics();

g.setColor(Color.BLUE);
for (int i = 1; i <= 10; i++) {
    g.fillOval(15 * i, 15 * i, 30, 30);
    panel.sleep(500);
}
```

  - Try adding `sleep` commands to loops in past exercises in this chapter and watch the panel draw itself piece by piece.

Animation exercise

- Modify the previous program to draw a "moving" animated car.