Object-Oriented Programming

Casual Dinner for Women in CS

- Next **Thursday, January 24** in Gates 219 at 6:00PM.
- Good food, great company, and everyone is invited!
- RSVP through email link (sent out earlier today).

Email Highlights

I think you look like David Guetta.
Like seriously.

Object-Oriented Programming

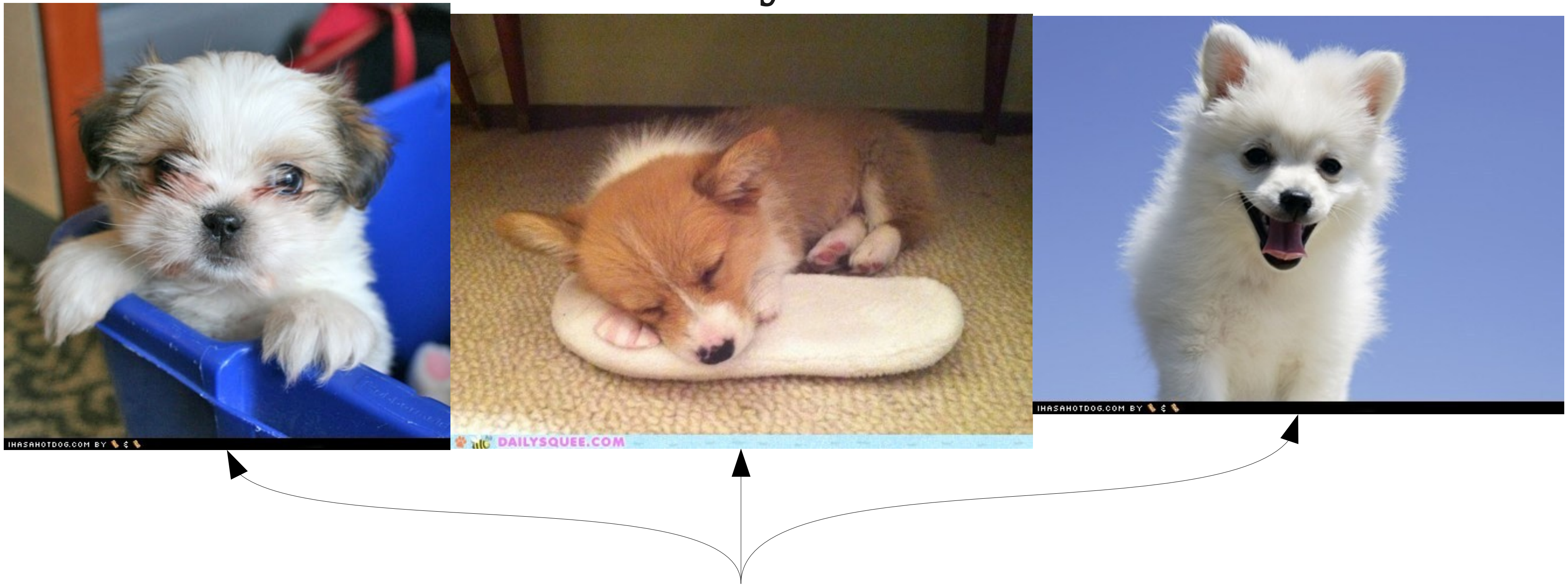
An **object** is an entity
that has state and behavior.

Has a fur color.
Has an energy level.
Has a level of cuteness.
Can be your friend.
Can sit.
Can stay.
Can bark.

A **class** is a set of features and behavior common to a group of objects.

Class Dog

Has a fur color.
Has an energy level.
Has a level of cuteness.
Can sit.
Can stay.
Can be your friend.



An **object** is an entity
that has state and behavior.

A **class** is a set of features and behavior
common to a group of objects.

An **instance of a class** is an
object that belongs to that class.

Class Dog

Has a fur color.
Has an energy level.
Has a level of cuteness.
Can sit.
Can stay.
Can be your friend.

Class Cat

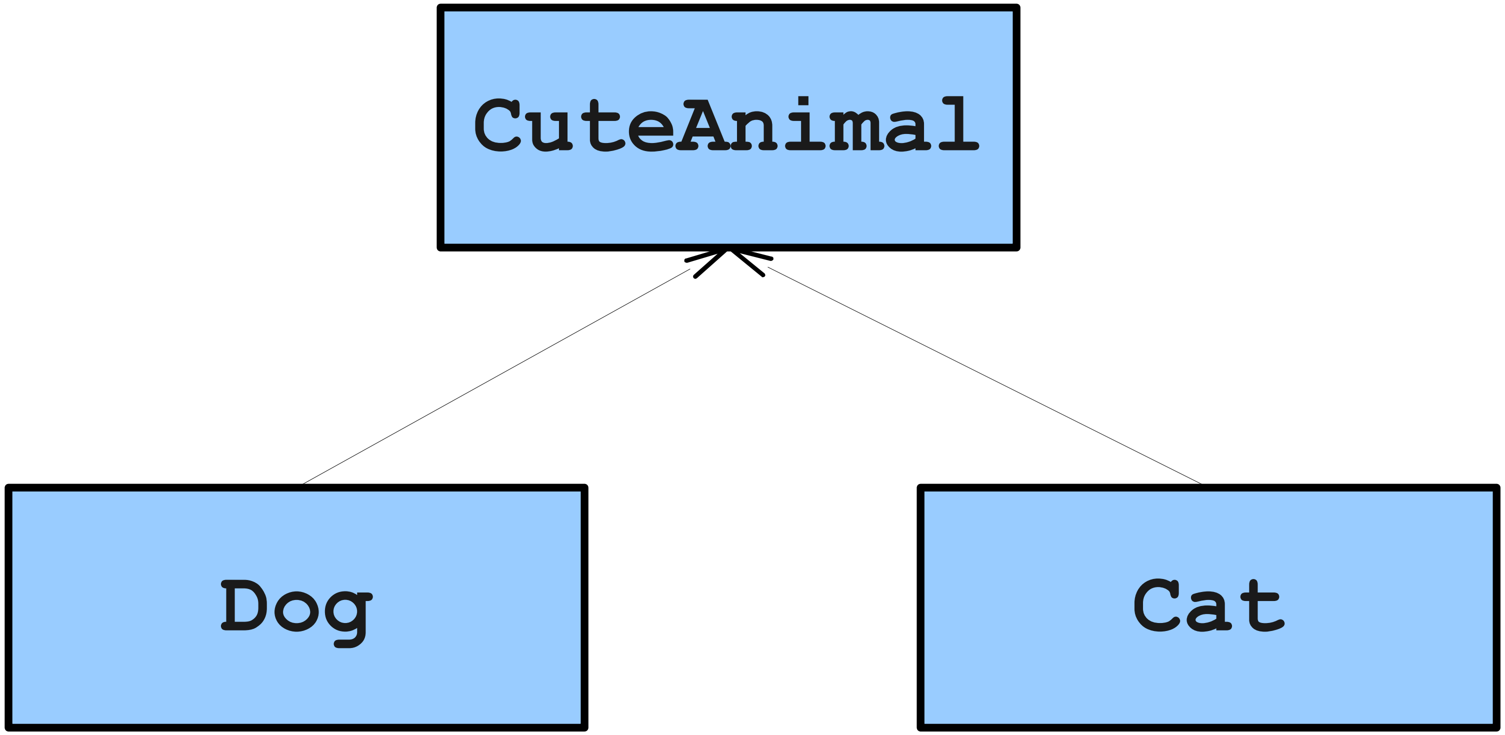
Has a fur color.
Has an energy level.
Has a level of cuteness.
Can purr.
Can haz cheezburger?

Dog

Has a fur color.
Has an energy level.
Has a level of cuteness.
Can sit.
Can stay.
Can be your friend.

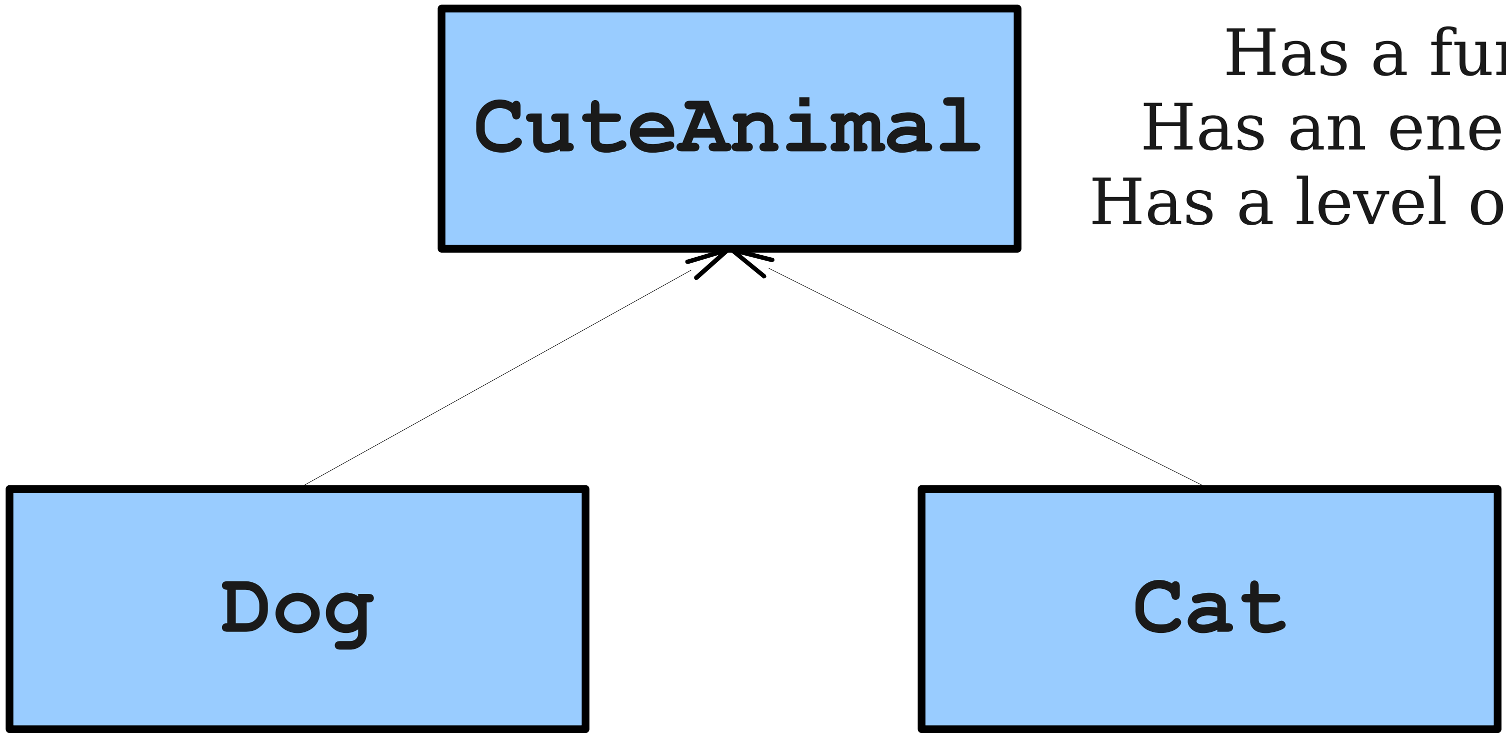
Cat

Has a fur color.
Has an energy level.
Has a level of cuteness.
Can purr.
Can haz cheezburger?



Has a fur color.
Has an energy level.
Has a level of cuteness.
Can sit.
Can stay.
Can be your friend.

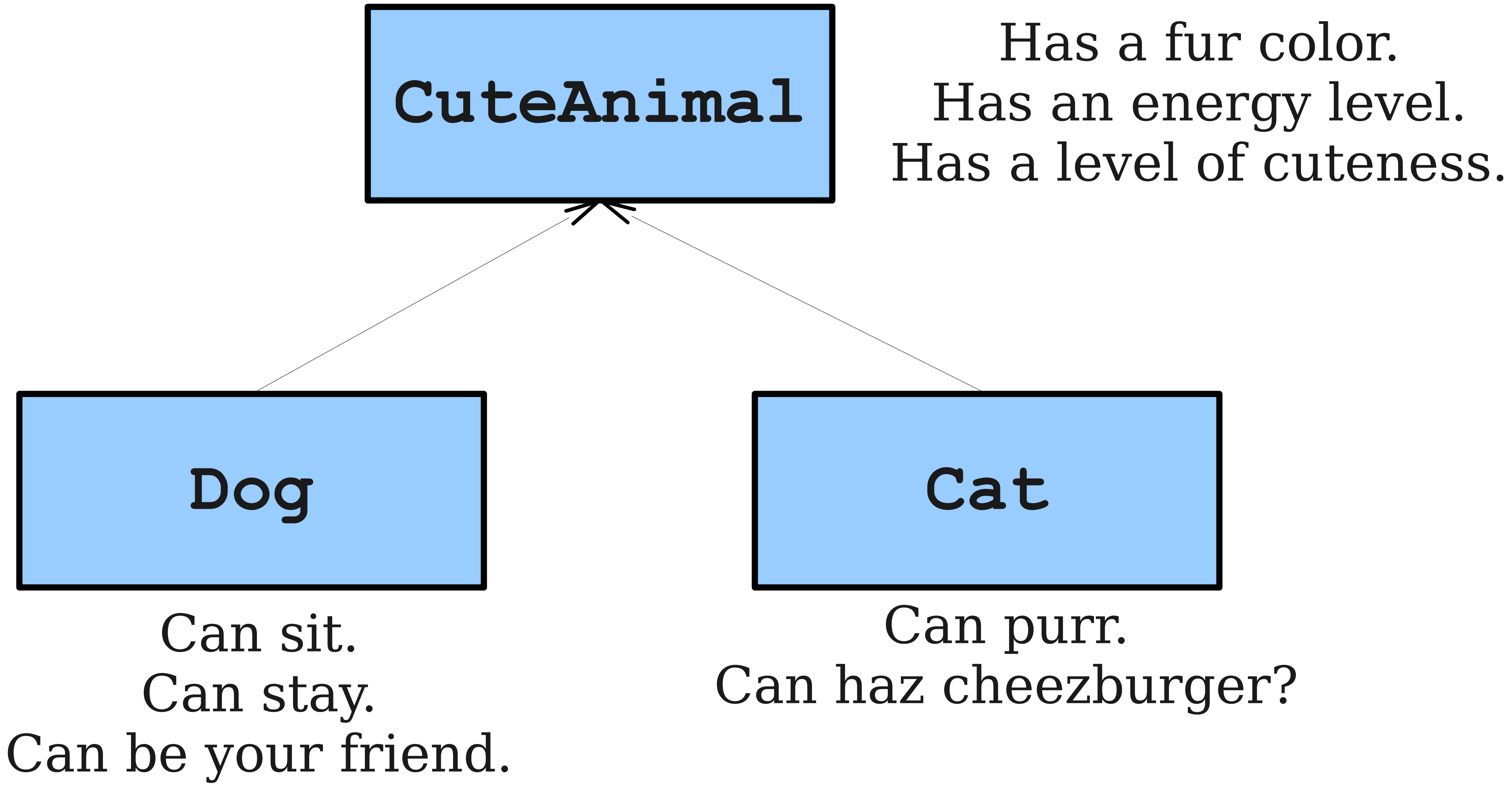
Has a fur color.
Has an energy level.
Has a level of cuteness.
Can purr.
Can haz cheezburger?



Has a fur color.
Has an energy level.
Has a level of cuteness.

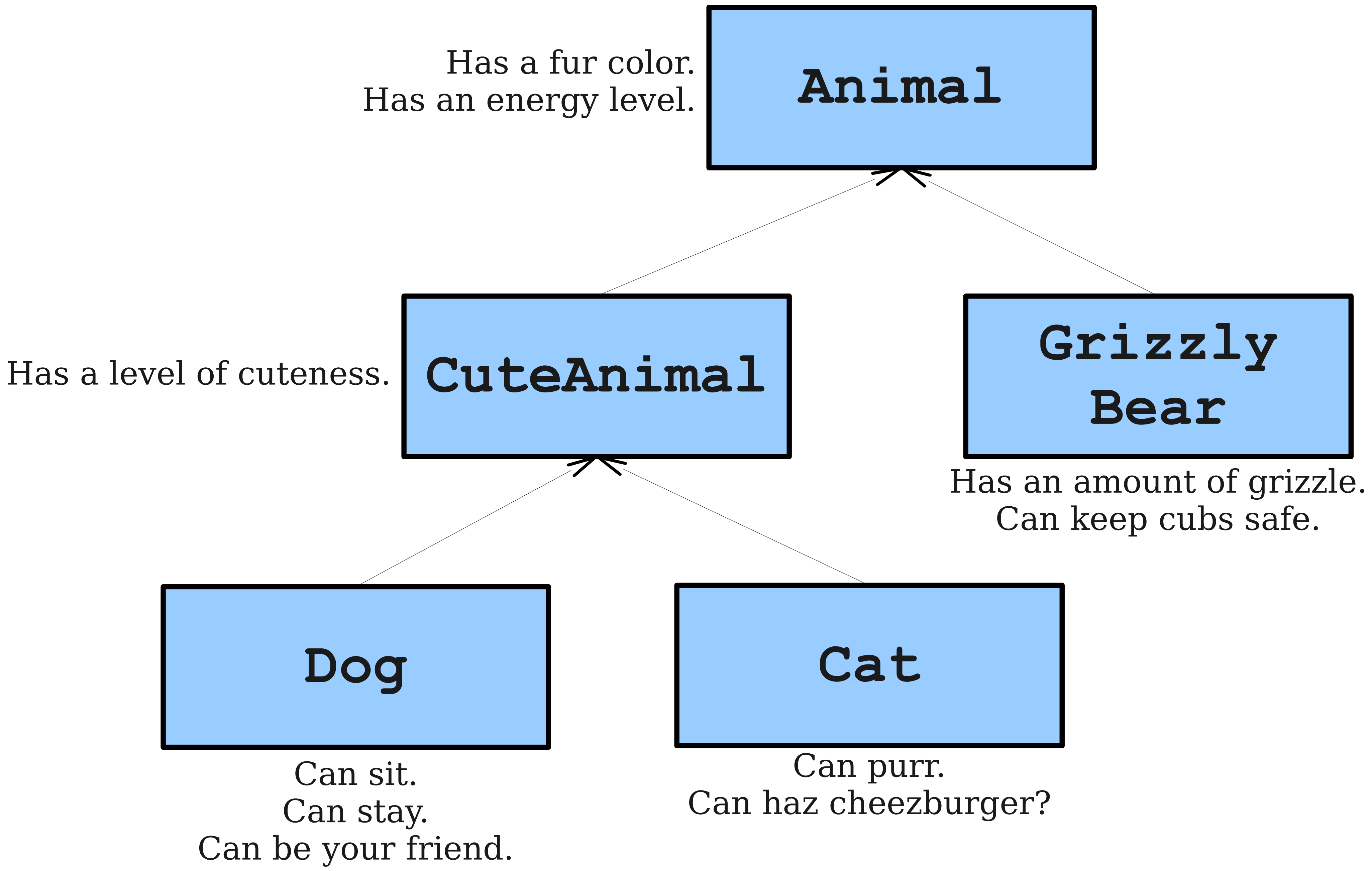
Has a fur color.
Has an energy level.
Has a level of cuteness.
Can sit.
Can stay.
Can be your friend.

Has a fur color.
Has an energy level.
Has a level of cuteness.
Can purr.
Can haz cheezburger?

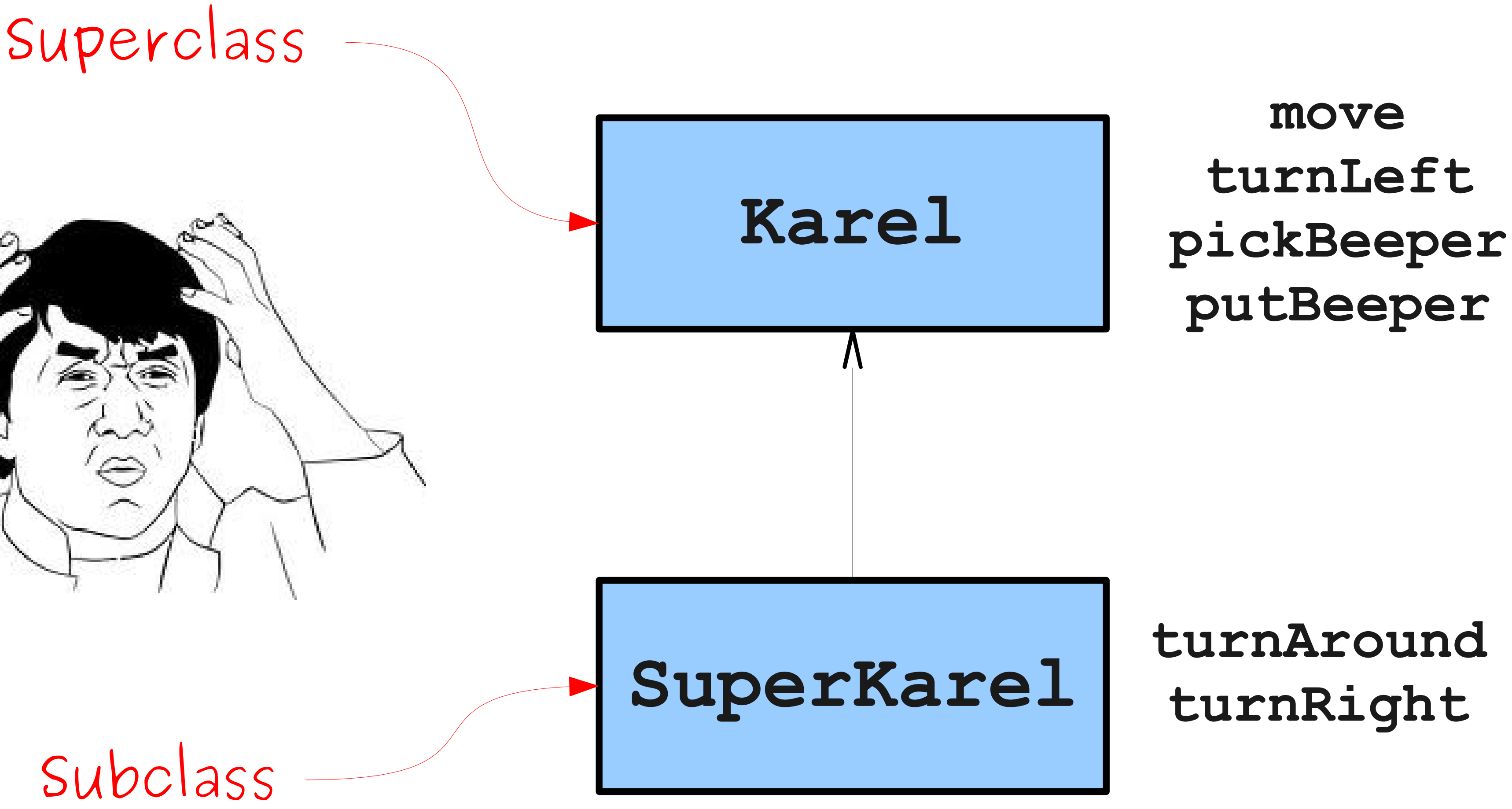


The class **`Dog`** and **`Cat`** classes are **subclasses** of the **`CuteAnimal`** class.

A Class Hierarchy



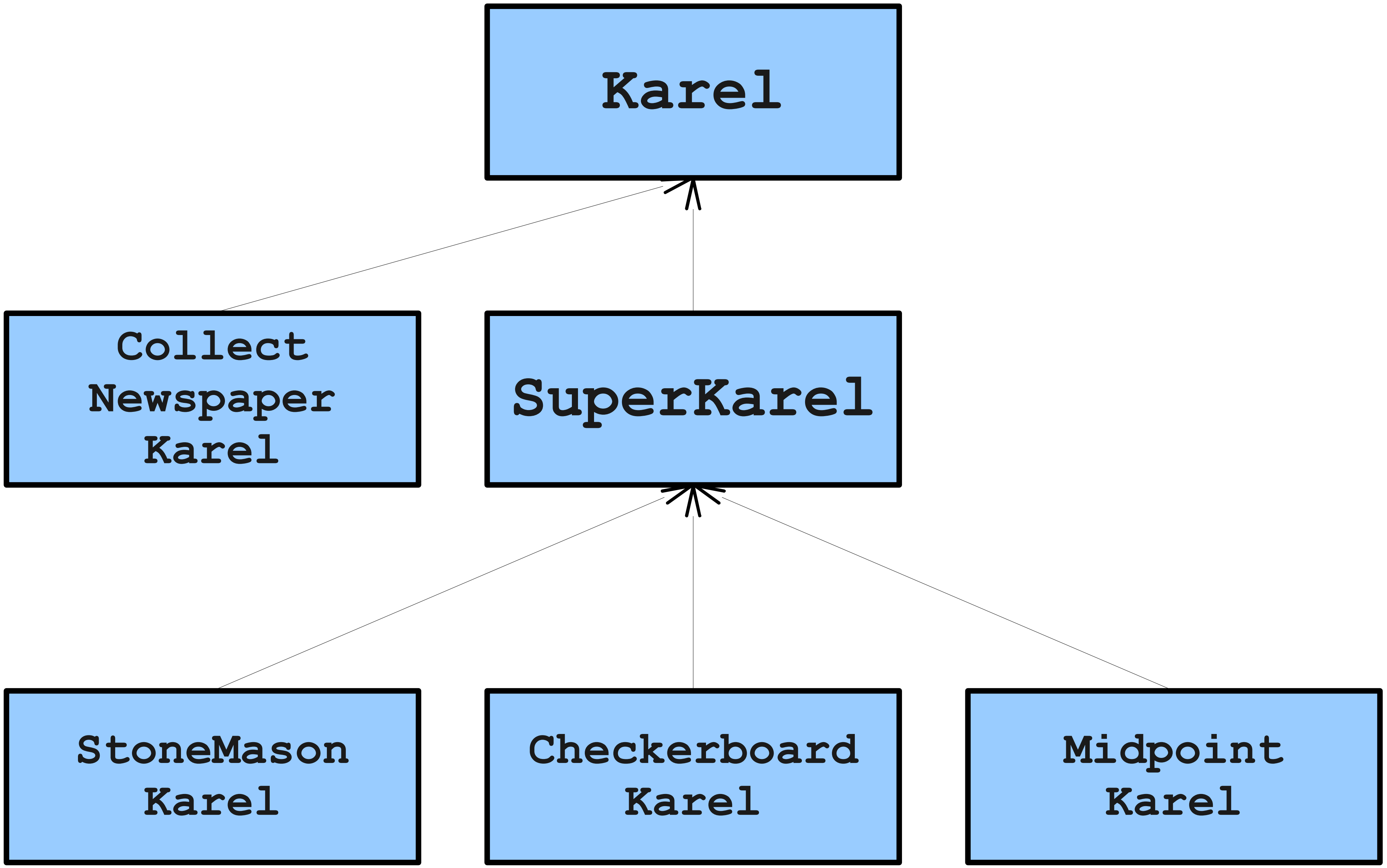
Classes so Far



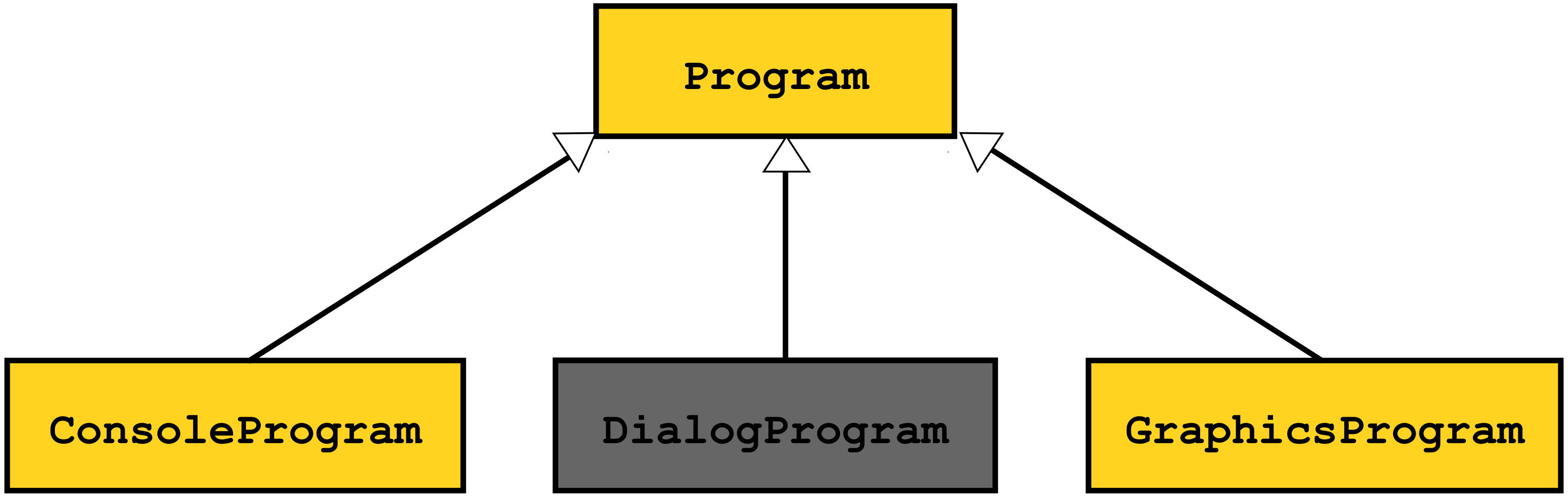
```java
/* File: RoombaKarel.java
 *
 * A Karel program in which Karel picks up all the beepers in a
 * square world.
 */

import stanford.karel.*;

public class RoombaKarel extends SuperKarel {
    public void run() {
        sweepRow();
        while (leftIsClear()) {
            moveToNextRow();
            sweepRow();
        }
    }

    /* Precondition:  Karel is facing East at the start of a row.
     * Postcondition: Karel is facing East at the start of a row,
     *                but the row has all beepers cleared from it
     */
    private void cleanOneRow() {
        sweepToEnd();
        comeHome();
    }

    /* ... */
}
```

```
/* File: RoombaKarel.java
 *
 * A Karel program in which Karel picks up all the beepers in a
 * square world.
 */

import stanford.karel.*;

public class RoombaKarel extends SuperKarel {
    public void run() {
        sweepRow();
        while (leftIsClear()) {
            moveToNextRow();
            sweepRow();
        }
    }

    /* Precondition:  Karel is facing East at the start of a row.
     * Postcondition: Karel is facing East at the start of a row,
     *                but the row has all beepers cleared from it
     */
    private void cleanOneRow() {
        sweepToEnd();
        comeHome();
    }

    /* ... */
}
```

How Does Karel Fit In?



`acm.program` Hierarchy

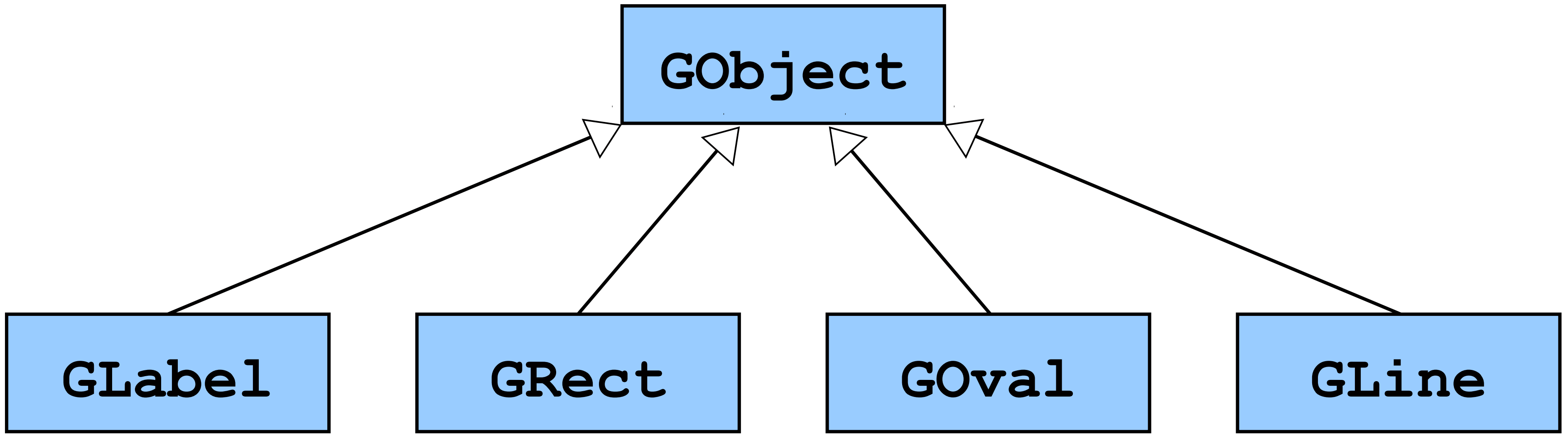


Graphic courtesy of Eric Roberts

Programming with Graphics

The `GObject` Hierarchy

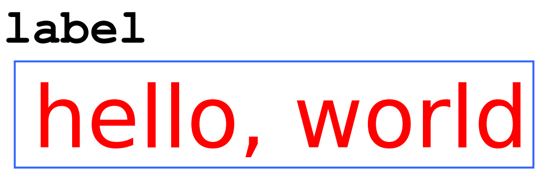
The classes that represent graphical objects form a hierarchy, part of which looks like this:

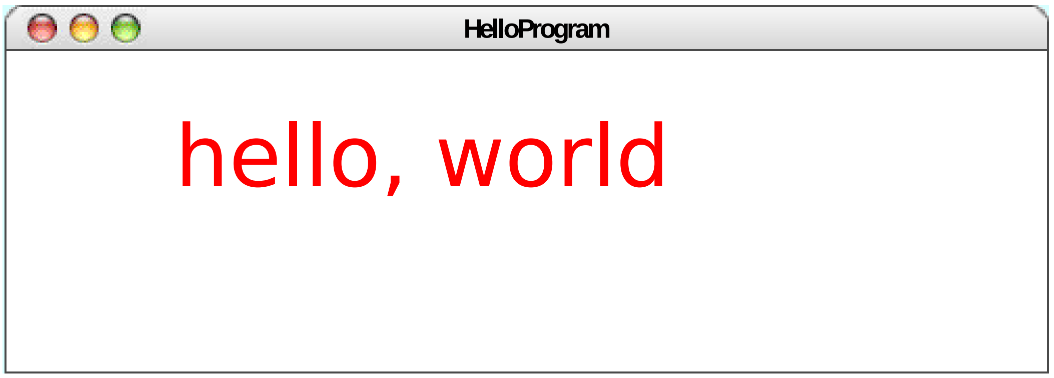


Graphic courtesy of Eric Roberts

Sending Messages to a **`GLabel`**

```
public class HelloProgram extends GraphicsProgram {
   public void run() {
      GLabel label = new GLabel("hello, world", 100, 75);
      label.setFont("SansSerif-36");
      label.setColor(Color.RED);
      add(label);
   }
}
```





Graphic courtesy of Eric Roberts

Objects and Variables

- Variables can be declared to hold objects.
- The type of the variable is the name of the class:
  - `GLabel label;`
  - `GOval oval;`
- Instances of a class can be created using the **`new`** keyword:
  - `GLabel label = new GLabel("Y?", 0, 0);`

Sending Messages

- To call a method on an object stored in a variable, use the syntax

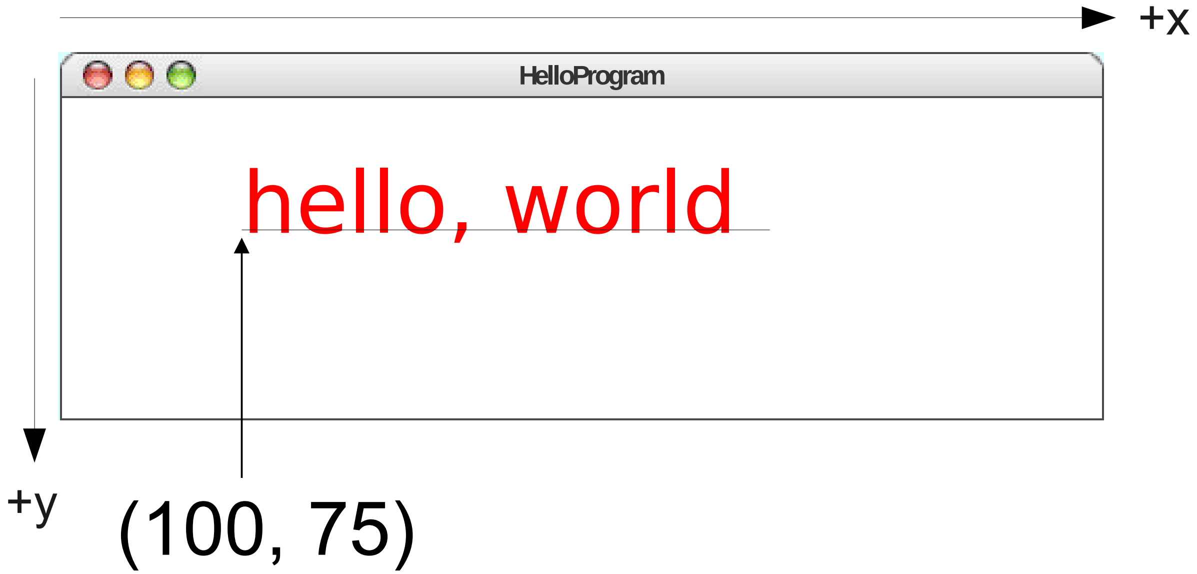
***object*.*method*(*parameters*)**

- For example:

```
label.setFont("Comic Sans-32");
label.setColor(Color.ORANGE);
```

Graphics Coordinates

- Origin is upper left.
- *x* coordinates increase from left to right.
- *y* coordinates increase from top to bottom.
- Units are **pixels** (dots on the screen).
- **`GLabel`** coordinates are baseline of first character.



Graphic courtesy of Eric Roberts

Operations on the **`GObject`** Class

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.	

Standard color names defined in the **`java.awt`** package:

`Color.BLACK`	**`Color.RED`**	**`Color.BLUE`**
`Color.DARK_GRAY`	**`Color.YELLOW`**	**`Color.MAGENTA`**
`Color.GRAY`	**`Color.GREEN`**	**`Color.ORANGE`**
`Color.LIGHT_GRAY`	**`Color.CYAN`**	**`Color.PINK`**
`Color.WHITE`		

Graphic courtesy of Eric Roberts

Operations on the **`GLabel`** Class

Constructor

`new GLabel(`*text*, *x*, *y*`)`

Creates a label containing the specified text that begins at the point (*x*, *y*).

Methods specific to the **`GLabel`** class

label`.`**`setFont(`***font*`)`

Sets the font used to display the label as specified by the font string.

The font is specified as

`"`*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.

Graphic courtesy of Eric Roberts

Drawing Geometrical Objects

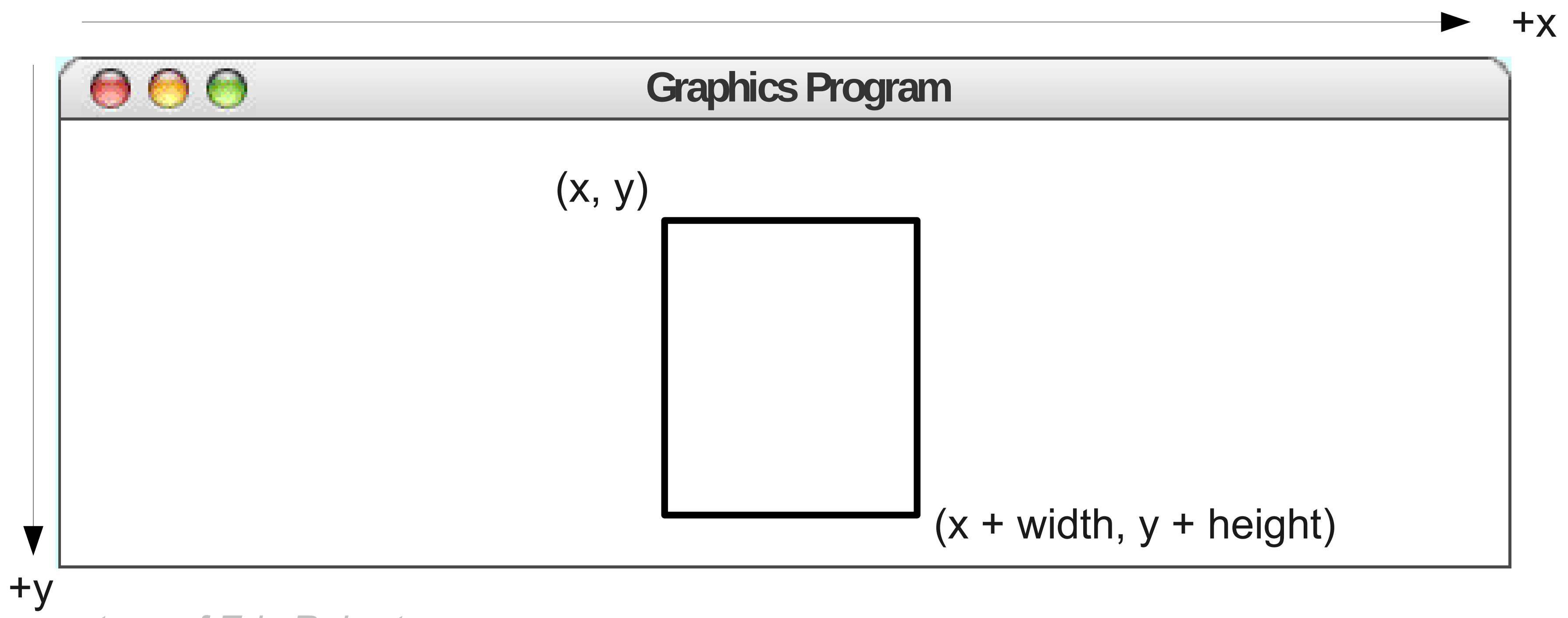
Graphic courtesy of Eric Roberts

Drawing Geometrical Objects

Constructors

new GRect(*x* **,** *y* **,** *width* **,** *height* **)**

Creates a rectangle whose upper left corner is at (*x*, *y*) of the specified size



Graphic courtesy of Eric Roberts

Drawing Geometrical Objects

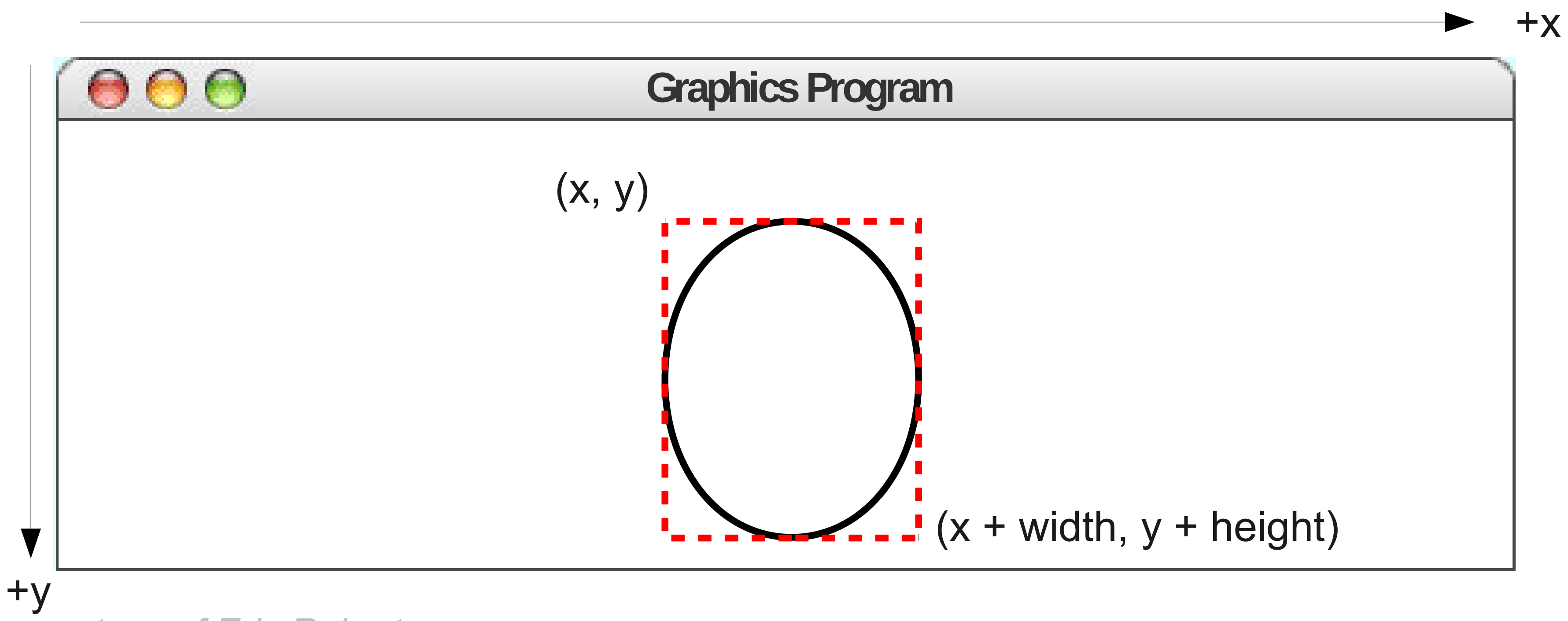
Constructors

`new GRect(` *x*, *y*, *width*, *height***`)`**

Creates a rectangle whose upper left corner is at (*x*, *y*) of the specified size

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

Creates an oval that fits inside the rectangle with the same dimensions.



Graphic courtesy of Eric Roberts

Drawing Geometrical Objects

Constructors

`new GRect(` *x*, *y*, *width*, *height***`)`**

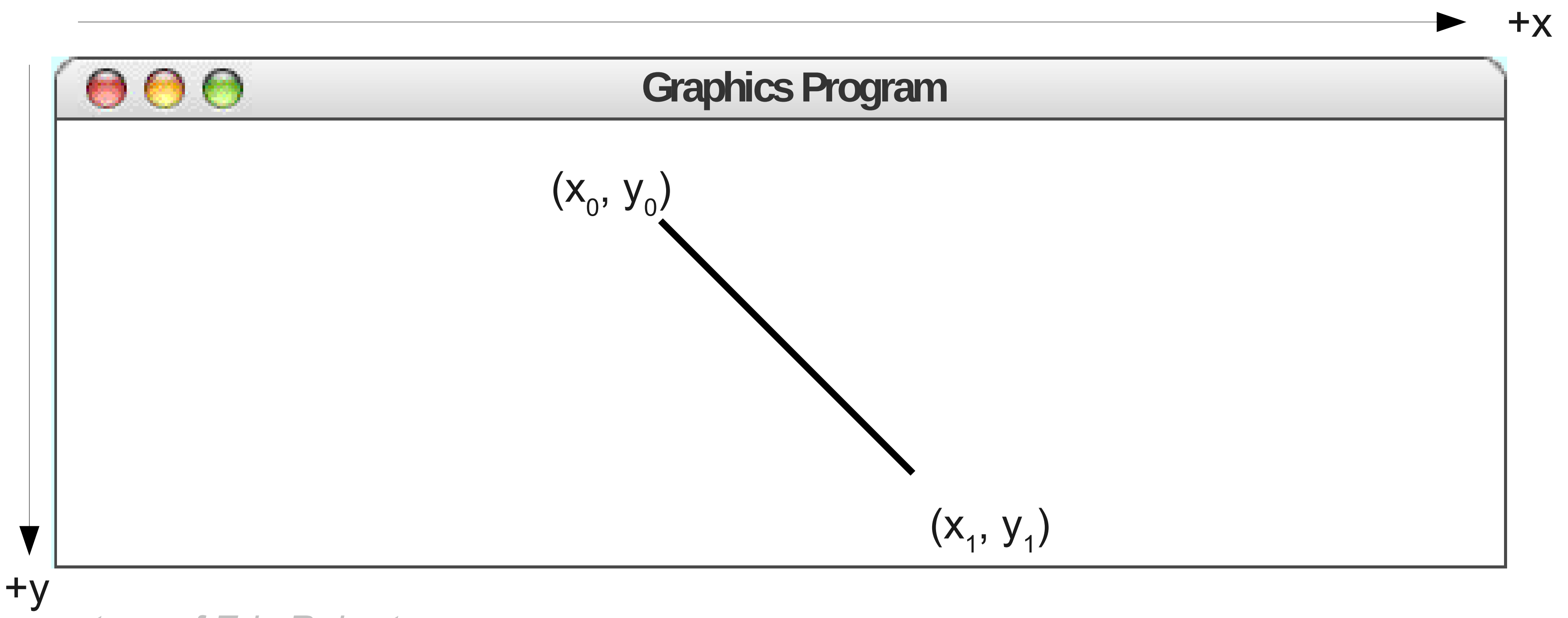
Creates a rectangle whose upper left corner is at (*x*, *y*) of the specified size

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

Creates an oval that fits inside the rectangle with the same dimensions.

`new GLine(` x_0, y_0, x_1, y_1**`)`**

Creates a line extending from (x_0, y_0) to (x_1, y_1).



Graphic courtesy of Eric Roberts

Drawing Geometrical Objects

Constructors

`new GRect(` *x*, *y*, *width*, *height***`)`**
Creates a rectangle whose upper left corner is at (*x*, *y*) of the specified size

`new GOval(` *x*, *y*, *width*, *height***`)`**
Creates an oval that fits inside the rectangle with the same dimensions.

`new GLine(` x_0, y_0, x_1, y_1**`)`**
Creates a line extending from (x_0, y_0) to (x_1, y_1).

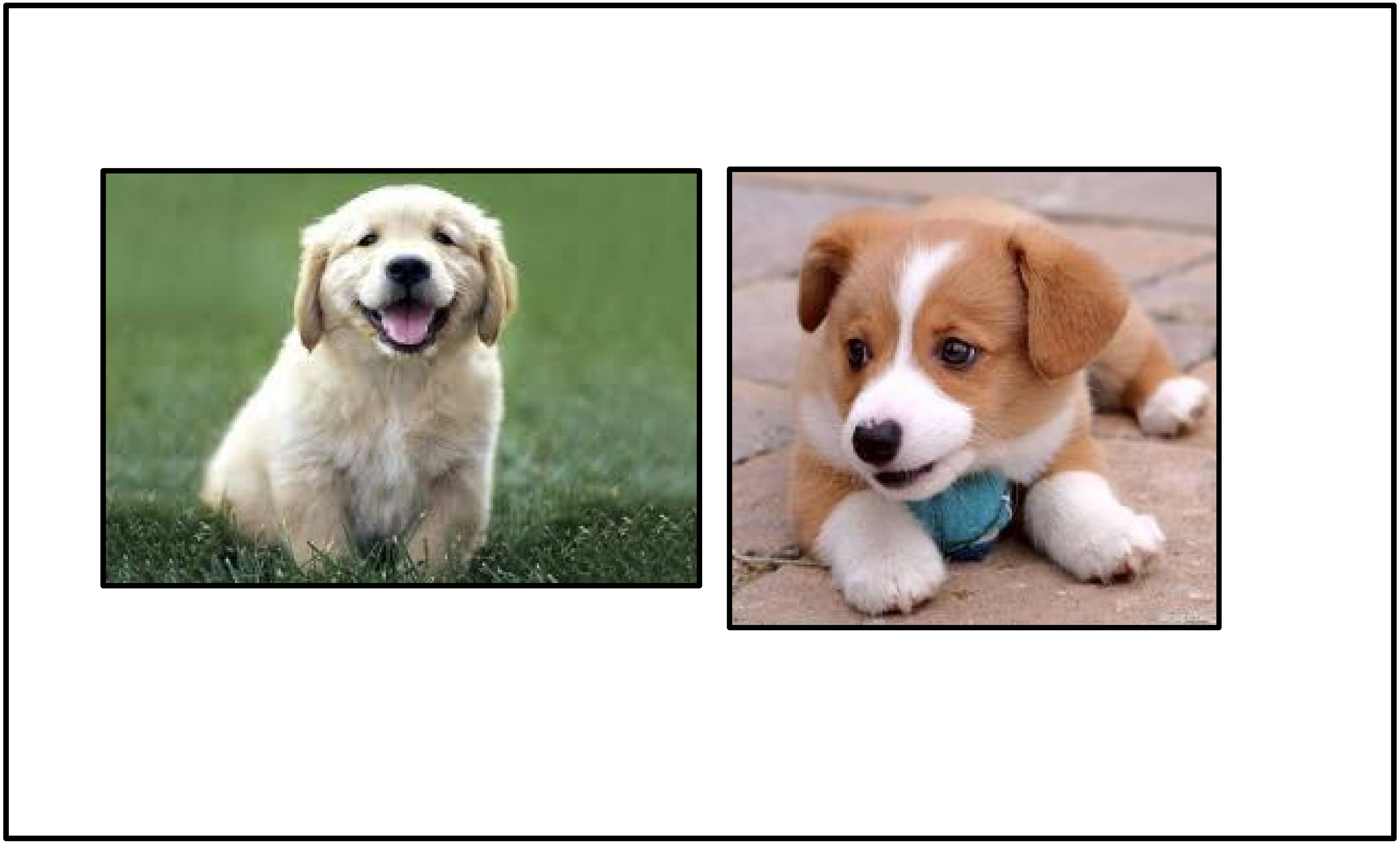
Methods shared by the **`GRect`** and **`GOval`** classes

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.

Graphic courtesy of Eric Roberts

The Collage Model



The Collage Model