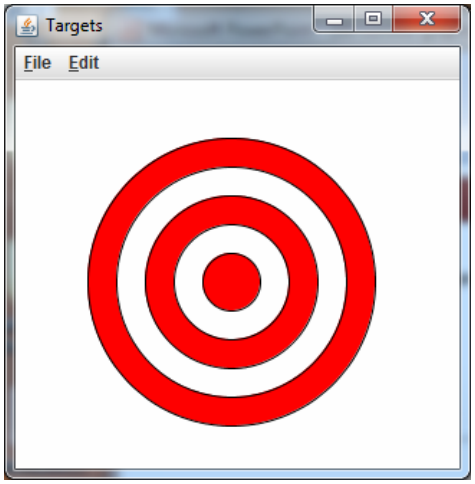
CS 106A, Lecture 11
Graphics

reading:

Art & Science of Java, 4.5, 3.2

Graphics programs

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

public class Name extends GraphicsProgram {
    public void run() {
        statements;
    }
}
```



- A GraphicsProgram draws 2D shapes, colors, lines, etc. instead of text console output.

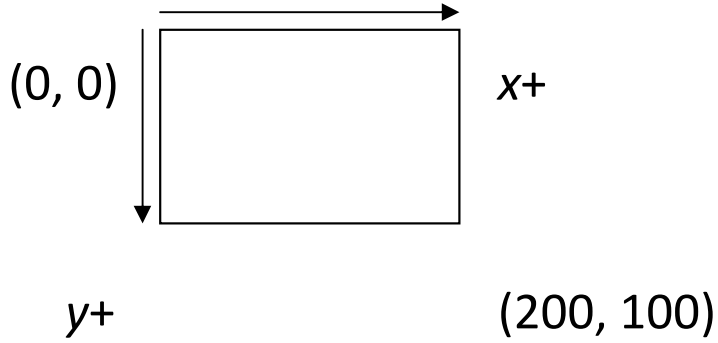
Coordinate system

- Each (x, y) position is a *pixel* ("picture element").

- Position (0, 0) is at the window's top-left corner.

  x increases rightward.
  y increases downward.

- The rectangle from (0, 0) to (200, 100) looks like this:



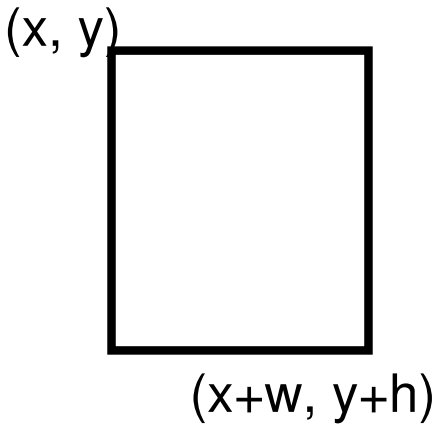
GraphicsProgram methods

- The `GraphicsProgram` contains these useful methods:

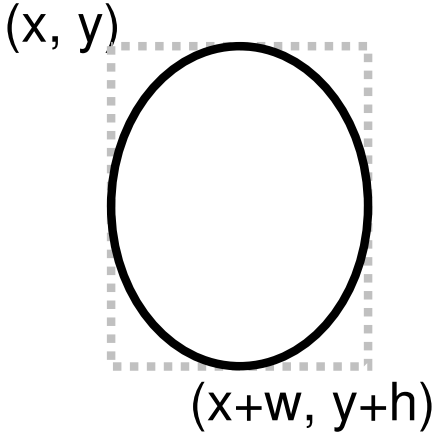
Method	Description
`add(gobj);` `add(gobj, x, y);`	adds a graphical object to the window
`getElementAt(x, y)`	return the object at the given (x,y) position(s)
`getElementCount()`	return number of graphical objects onscreen
`getWidth(), getHeight()`	return dimensions of window
`remove(gobj);`	removes 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
`setSize(w, h);`	set size of entire window (incl. bars/borders)
`setTitle("text");`	sets window's title bar text

Graphical objects

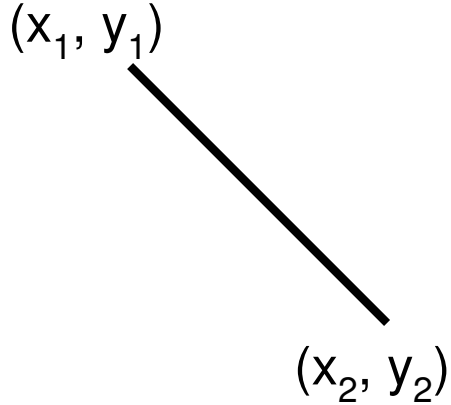
GRect



GOval



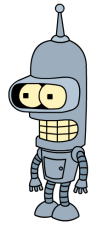
GLine



GLabel

Hello there!

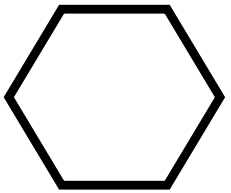
GImage



GArc

GRoundRect

GPolygon



Graphical objects

Graphical object	Description
`new GImage("`***filename***`", `***x***`, `***y***`)`	image from the given file, drawn at *(x, y)*
`new GLabel("`***text***`", `***x***`, `***y***`)`	text with bottom-left at *(x, y)*
`new GLine(`***x1***`, `***y1***`, `***x2***`, `***y2***`)`	line between points (*x1*, *y1*), (*x2*, *y2*)
`new GOval(`***x***`, `***y***`, `***w***`, `***h***`)`	largest oval that fits in a box of size *w* * *h* with top-left at (*x*, *y*)
`new GRect(`***x***`, `***y***`, `***w***`, `***h***`)`	rectangle of size *w* * *h* with top-left at (*x*, *y*)

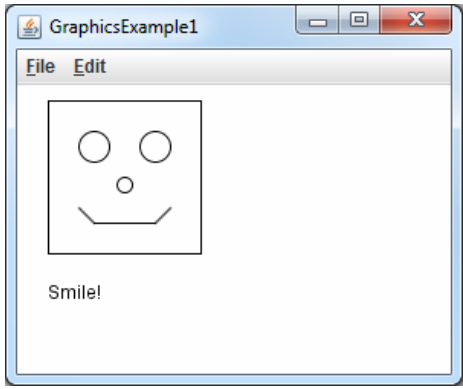
- to place an object in the window, use the add method.

```
add(new Type(parameters));
```

- for others, see:
  - http://cs.stanford.edu/people/eroberts/jtf/javadoc/student/

Graphics example

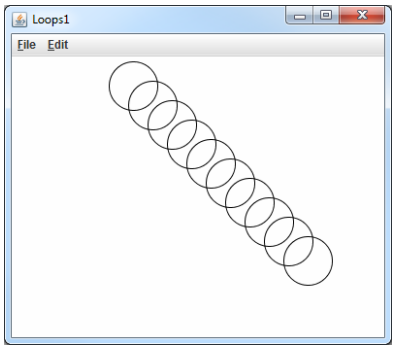
```
import acm.graphics.*;
import acm.program.*;
import java.awt.*;

public class SmileyFace extends GraphicsProgram {
    public void run() {
        setCanvasSize(300, 250);
        add(new GRect(20, 10, 100, 100));    // head
        add(new GOval(40, 30, 20, 20));      // left eye
        add(new GOval(80, 30, 20, 20));      // right eye
        add(new GOval(65, 60, 10, 10));      // nose
        add(new GLine(40, 80, 50, 90));      // mouth
        add(new GLine(50, 90, 90, 90));      // mouth
        add(new GLine(90, 90, 100, 80));     // mouth
        add(new GLabel("Smile!", 20, 140));
    }
}
```



Drawing with loops

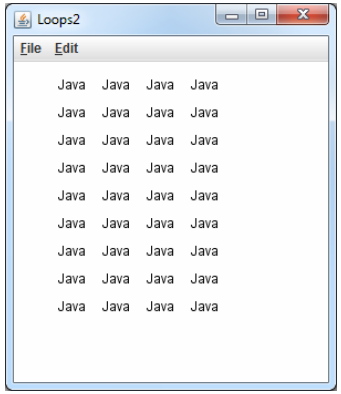
- The *x*,*y*,*w*,*h* expressions can use the loop counter variable:

```
for (int i = 0; i < 10; i++) {
    add(new GOval(100 + 20 * i, 5 + 20 * i, 50, 50));
}   //                  x            y        w   h
```



- Nested loops can be used with graphics:

```
for (int x = 1; x <= 4; x++) {
    for (int y = 1; y <= 9; y++) {
        add(new GLabel("Java", x * 40, y * 25));
    }
}
```

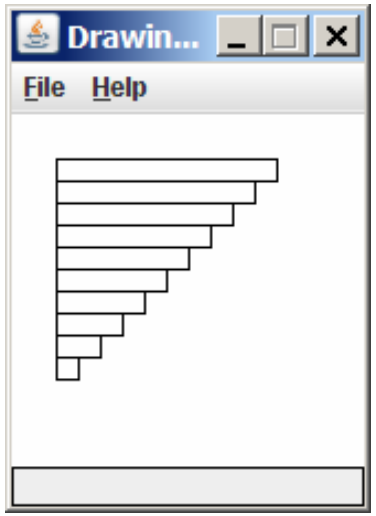


Drawing w/ loops exercise

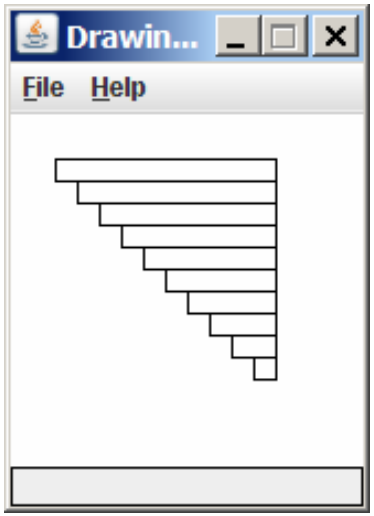
- **Q:** What is the output of the following code?

```
for (int i = 0; i < 10; i++) {
    add(new GRect(20 + 10 * i,  20 + 10 * i,
                  100 - 10 * i, 10));
}
```

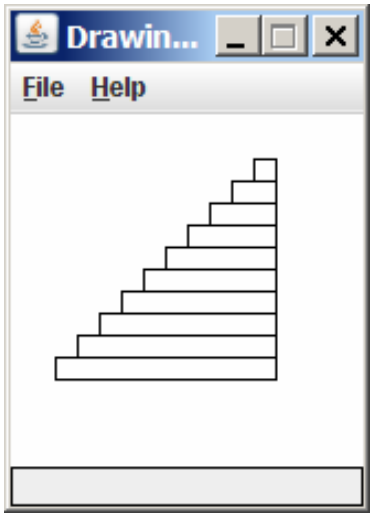
A.



B.



C.



D.

none

– *(How would we modify the code above to produce each output?)*

Graphical object methods

- All graphical objects have these methods inside them (and more):

Method	Description
obj.move(*dx*, *dy*)	adjusts location by the given amount
obj.setBackground(*Color*)	sets overall window's background color
obj.setFilled(*boolean*)	whether to fill the shape with color
obj.setFillColor(*Color*)	what color to fill the shape with
obj.setColor(*Color*)	what color to outline the shape with
obj.setLocation(*x*, *y*)	change the object's x/y position
obj.setSize(*w*, *h*)	change the objects width*height size

- To call these methods, you must capture the object in a variable:

```
GRect rect = new GRect(20, 30, 60, 25);
rect.setColor(Color.RED);
rect.setFilled(true);
add(rect);
```

Object method example

```
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);
    }
}
```

label

hello, world

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

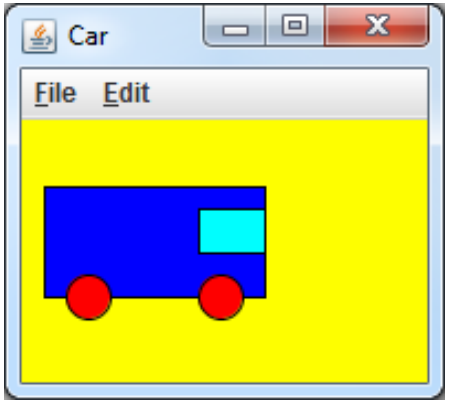
  `new Color(`***`red, green, blue`***`)`

  - Example:

```
rect.setColor(new Color(192, 128, 64));
```

"Collage" model

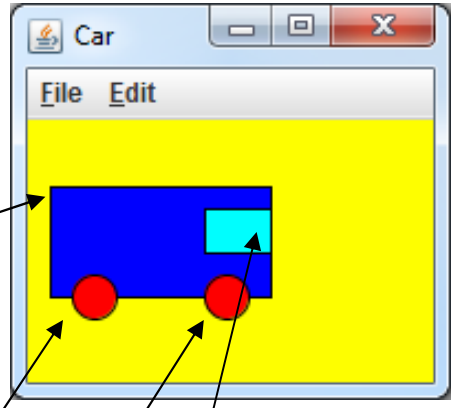
- When two shapes occupy the same pixels, the last one drawn "wins" and is shown "on top."



- Write a graphical program named **`Car`** that draws a figure that looks like a (crappy) car.
  - Red wheels at (20, 70) and (80, 70), size 20x20
  - Cyan windshield at (80, 40), size 30x20
  - Blue body at (10, 30), size 100x50
  - yellow background

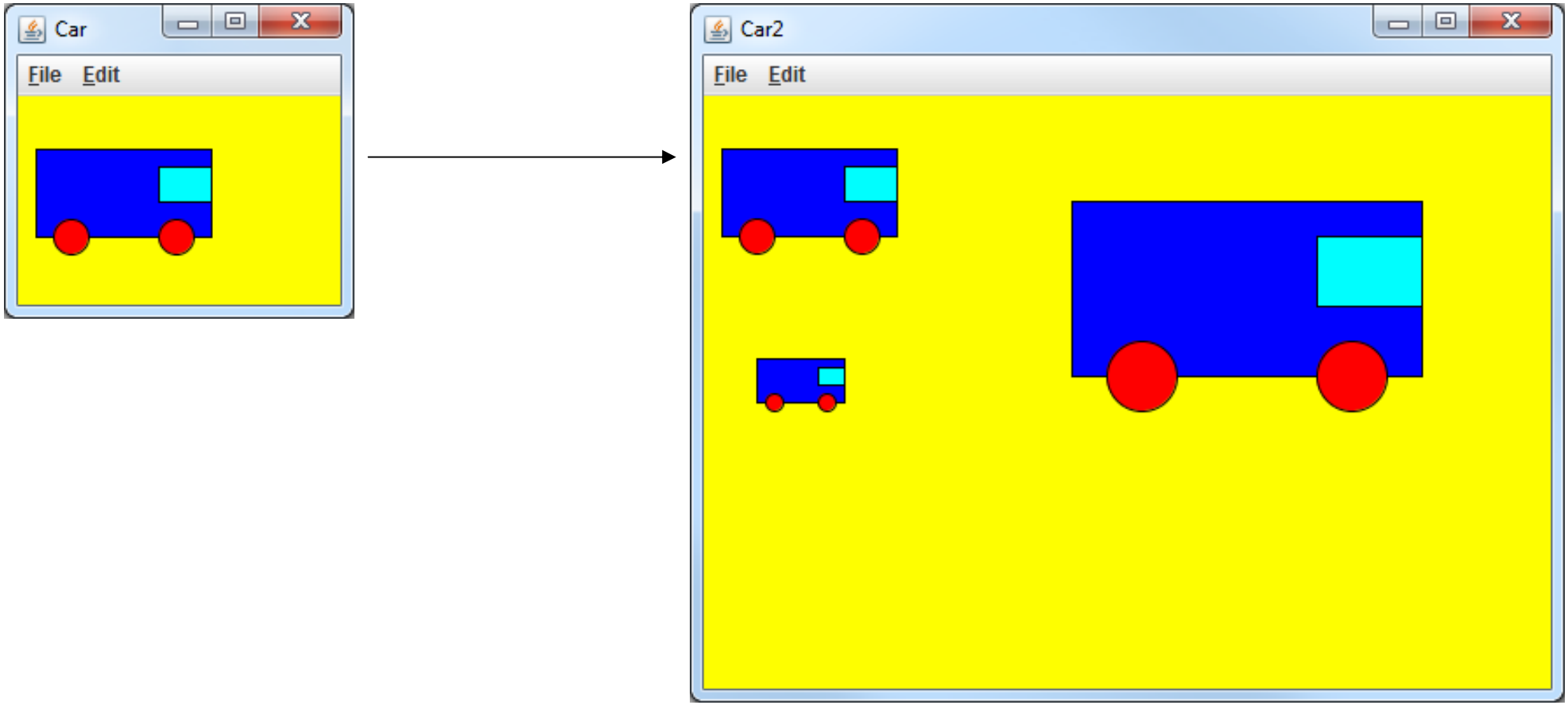
Car solution

```
// When 2 shapes occupy the same pixels, the last one drawn "wins"
public class Car extends GraphicsProgram {
    public void run() {
        setCanvasSize(200, 180);
        setBackground(Color.YELLOW);

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

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

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

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



Parameterized graphics

- In the last lecture we saw code to draw a graphical car.
- How would you make a method for drawing cars of different locations and sizes?



Parameterized solution

```
public class Car2 extends GraphicsProgram {
  public void run() {
    setBackground(Color.YELLOW);
    drawCar(10, 30, 100);
    drawCar(210, 60, 200);
    drawCar(30, 150, 50);
  }

  public void drawCar(int x, int y, int size) {
    GRect body = new GRect(x, y, size, size/2);
    body.setFilled(true);
    body.setFillColor(Color.BLUE);
    add(body);

    GOval wheel1 = new GOval(x+size/10, y+2*size/5, size/5, size/5);
    wheel1.setFilled(true);
    wheel1.setFillColor(Color.RED);
    add(wheel1);

    GOval wheel2 = new GOval(x+7*size/10, y+2*size/5, size/5, size/5);
    wheel2.setFilled(true);
    wheel2.setFillColor(Color.RED);
    add(wheel2);

    GRect windshield = new GRect(x+7*size/10, y+size/10, 3*size/10, size/5);
    windshield.setFilled(true);
    windshield.setFillColor(Color.CYAN);
    add(windshield);
} }
```

Graphics and returns

- Methods of graphical objects that return values:

Method	Description
obj.getColor()	the color used to color the shape outline
obj.getFillColor()	the color used to color the shape interior
obj.getX()	the left *x*-coordinate of the shape
obj.getY()	the top *y*-coordinate of the shape
obj.getWidth()	number of pixels wide the shape is
obj.getHeight()	number of pixels tall the shape is

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

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

Graphics exercise

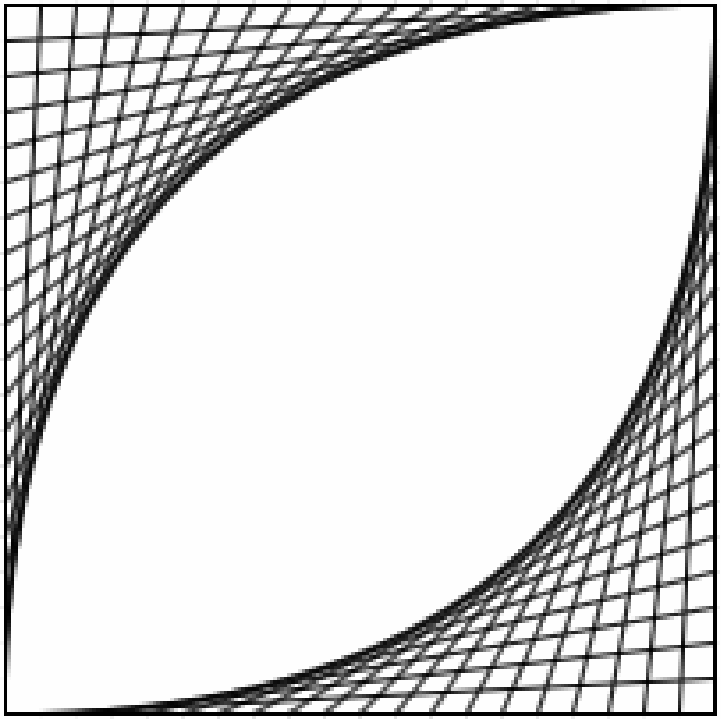
- Write a graphical program Football that draws a series of lines:
  - Outer square is at (10, 30) and size 200x200
  - each line is 10px apart in each dimension

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:

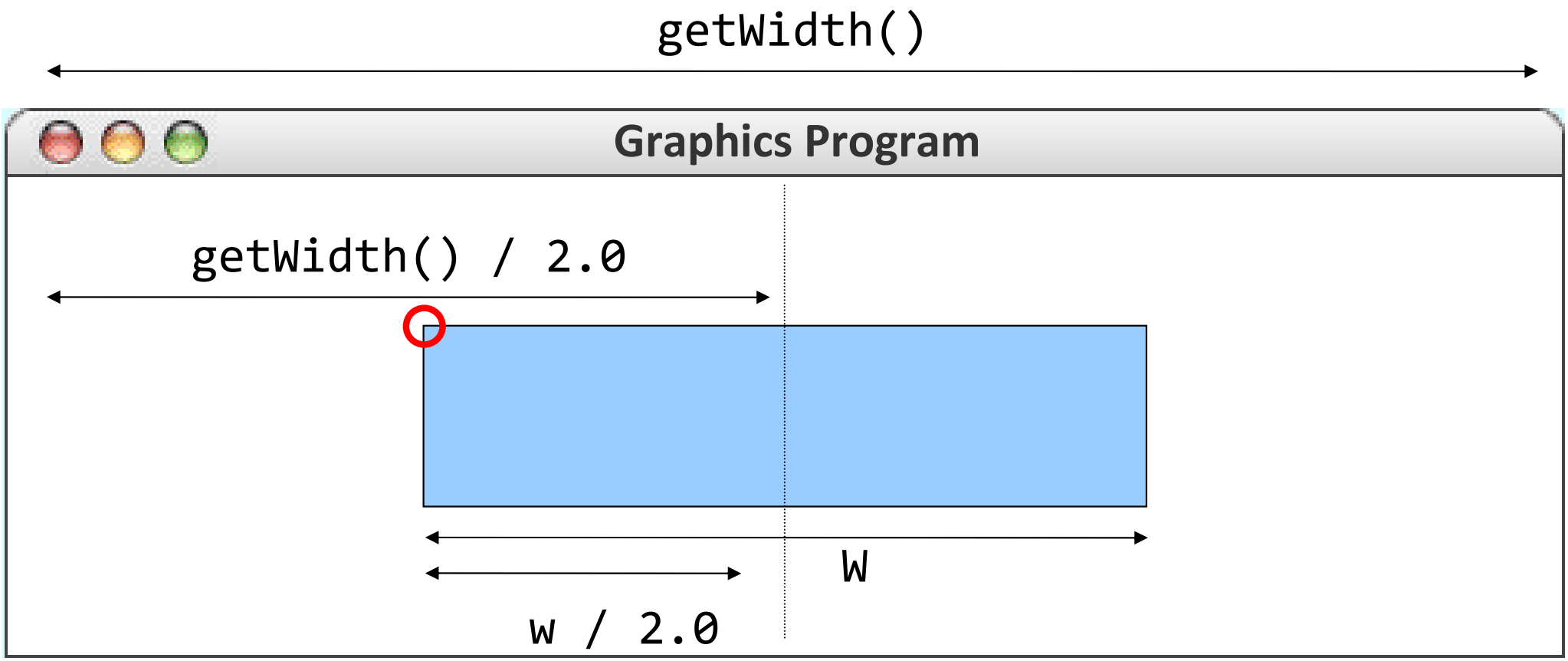
- (210, 30) to (210, 230)
- (210, 40) to (200, 230)
- ...
- (210, 220) to (20, 230)



Exercise solution

```
import acm.graphics.*;
import acm.program.*;

public class Football extends GraphicsProgram {
    public void run() {
        add(new GRect(10, 30, 200, 200));

        // top-left lines
        for (int i = 0; i < 20; i++) {
            add(new GLine(210 - i*10, 30, 10, 30 + i*10));
        }

        // bottom-right lines
        for (int i = 0; i < 20; i++) {
            add(new GLine(210, 30 + i*10, 210 - i*10, 230));
        }
    }
}
```

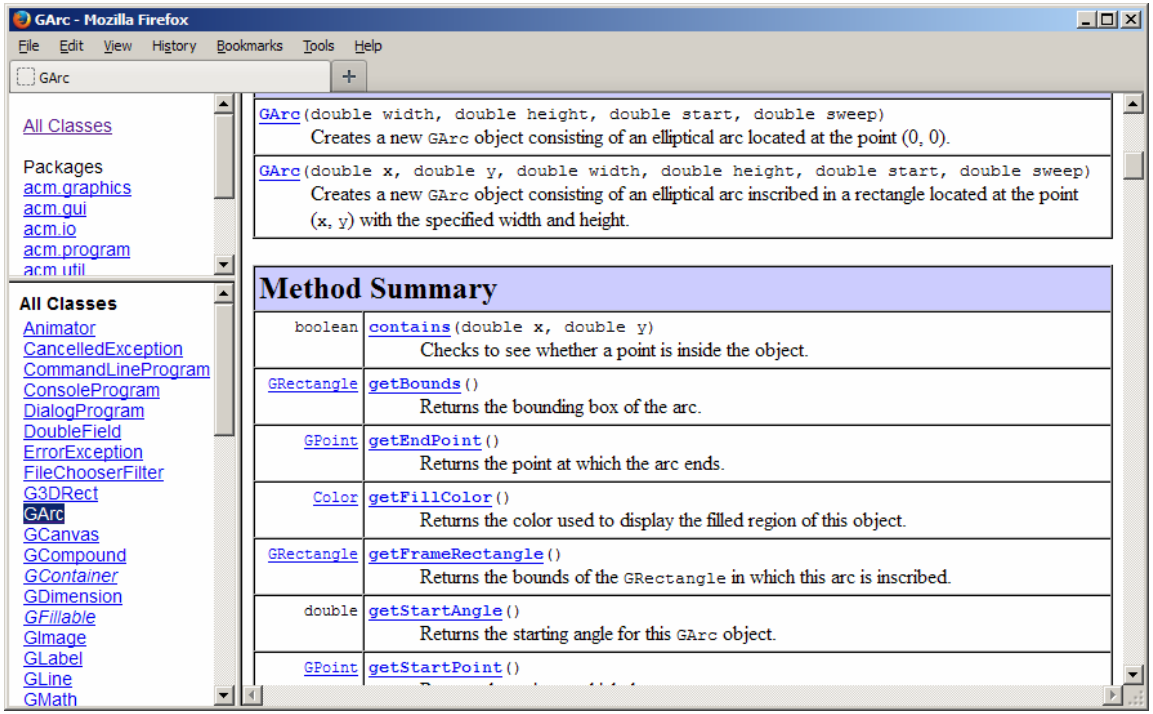
Centering



rectangle's x value = ???

Learning more

- Click the "**Stanford Java Lib**" link on the class web site.
  - 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.



http://cs.stanford.edu/people/eroberts/jtf/javadoc/student/