8. Describe the difference between the philosophical terms *holism* and *reductionism*. Why are these concepts important to programming?
9. What is the difference between a *class* and an *object?*
10. Define the terms *subclass, superclass,* and *inheritance*.
11. What Java keyword is associated with the use of a constructor?
12. What is the difference between a **ConsoleProgram** and a **DialogProgram**?
13. True or false: The process of sending a message to a Java object is usually implemented by calling a method in that object.
14. In Java, how do you specify the object to which a message is directed?
15. What are the four **GObject** subclasses described in this chapter?
16. Which of these subclasses respond to the method **setFilled**? Which respond to the method **setFont**?
17. In Chapter 9, you will learn about several additional **GObject** subclasses beyond the four listed here. Will these classes respond to the method **setColor**?
18. In what ways does Java's coordinate system differ from the traditional Cartesian coordinate system?

Programming exercises

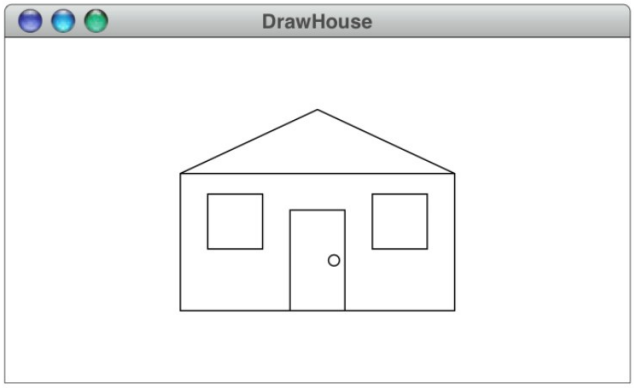
1. Type in the **HelloProgram.java** program exactly as it appears in this chapter and get it working. Change the message so that it reads "I love Java" instead. Add your name as a signature to the lower right corner.
2. The following program was written without comments or instructions to the user, except for a couple of input prompts:

```
import acm.program.*;

public class MyProgram extends ConsoleProgram {
   public void run() {
      double b = readDouble("Enter b: ");
      double h = readDouble("Enter h: ");
      double a = (b * h) / 2;
      println("a = " + a);
   }
}
```

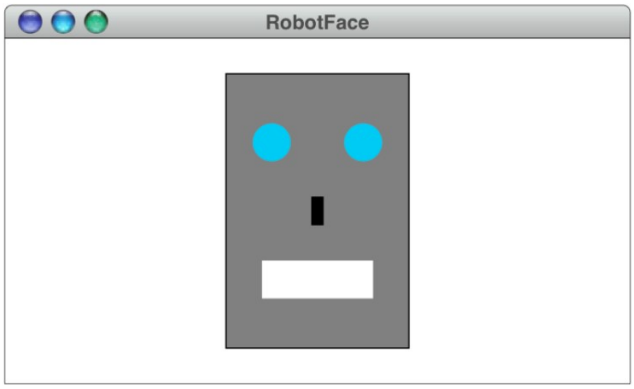
   Read through the program and figure out what it is doing. What result is it calculating? Rewrite this program so it is easier to understand, both for the user and for the programmer who must modify the program in the future.
3. Extend the **Add2Integers** program shown in Figure 2-2 so that it adds three integers instead.

4. Write a **`GraphicsProgram`** that generates the following simple picture of a house with a peaked roof, two windows, and a door with a circular doorknob:



As with each of the graphical exercises in this chapter, you should choose coordinate values for your programs that produce a reasonable approximation of the diagram shown. In Chapter 3, you will learn how to compute coordinate values so that, for example, the house is centered in the window, the door is centered in the house frame, and the windows are centered horizontally between the door and the side walls.

5. Write a **`GraphicsProgram`** that draws the following picture of a robot face:

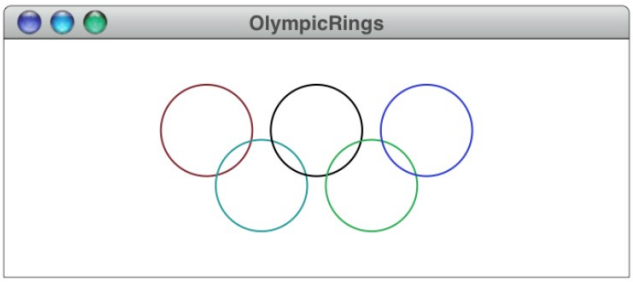


The eyes are orange, the nose is black, and the mouth is white. The face is filled in gray but outlined in black.

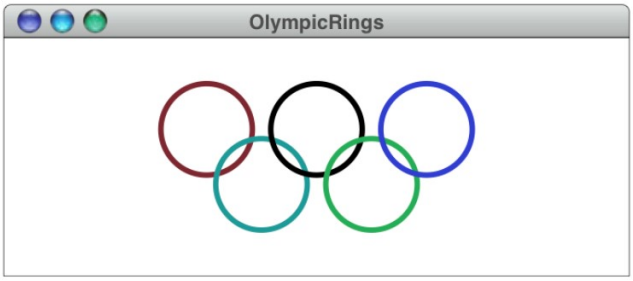
6. Write a **`GraphicsProgram`** that draws the following picture of an archery target, which also happens to be the logo of a large discount chain:

Both the outer and inner circles should be colored red.

7. Write a **`GraphicsProgram`** that draws the five interlocking rings (blue, yellow, black, green, and red) that form the symbol of the Olympic Games:



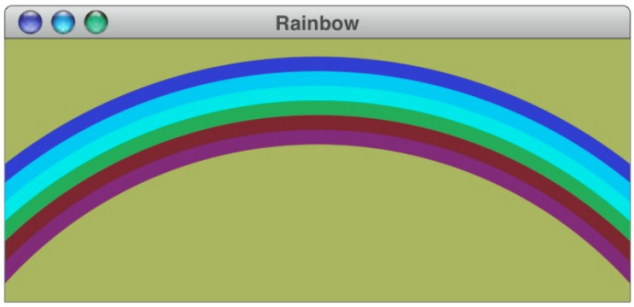
8. On most output devices, the Olympic Games logo from the preceding exercise doesn't show up all that well because the yellow circle (and to a lesser extent the green one) tends to disappear against the white background of the window. Part of the problem is that the outlines that Java draws for the **`GOval`** class are only one pixel wide, which doesn't show up well when drawn in lighter colors. It would be easier to see the rings if the borders were three pixels wide, like this:



Making this change would be easy if the **`GOval`** class included a method to set the width of the border, but, alas, it does not. Think about the problem for a bit

and see whether you can come up with a strategy for creating the three-pixel-wide display using only the tools you currently have.

9. Write a **`GraphicsProgram`** that draws a picture of a rainbow that looks something like this:



Starting at the top, the six stripes in the rainbow are red, orange, yellow, green, blue, and magenta, respectively; cyan makes a lovely color for the sky.

At first glance, it might seem as if this problem requires you to draw arcs. As it turns out, you can create the stripes in the rainbow using only circles, although seeing how this is possible forces you to think outside the box—in a literal as well as a figurative sense. The common center for each circle is some distance below the bottom of the window, and the diameters of the circles are wider than the screen. The **`GraphicsProgram`** shows only the part of the figure that actually appears in the window. This process of reducing a picture to the visible area is called **clipping.**