Assignment 2: Welcome to Java!

Based on an assignment by Eric Roberts and Mehran Sahami

Having helped Karel the Robot through the challenges of Assignment 1, it's time to transition into the wonderful world of Java programming!

In this assignment, you will work through six small programs, each giving you a feel for how to use variables, methods, control structures, the `acm.graphics` package, and other features of Java not present in Karel's world. By the time you're done, you'll be ready to start building larger, more elaborate Java programs.

Good luck, and have fun!

Due Monday, January 26 at 3:15PM

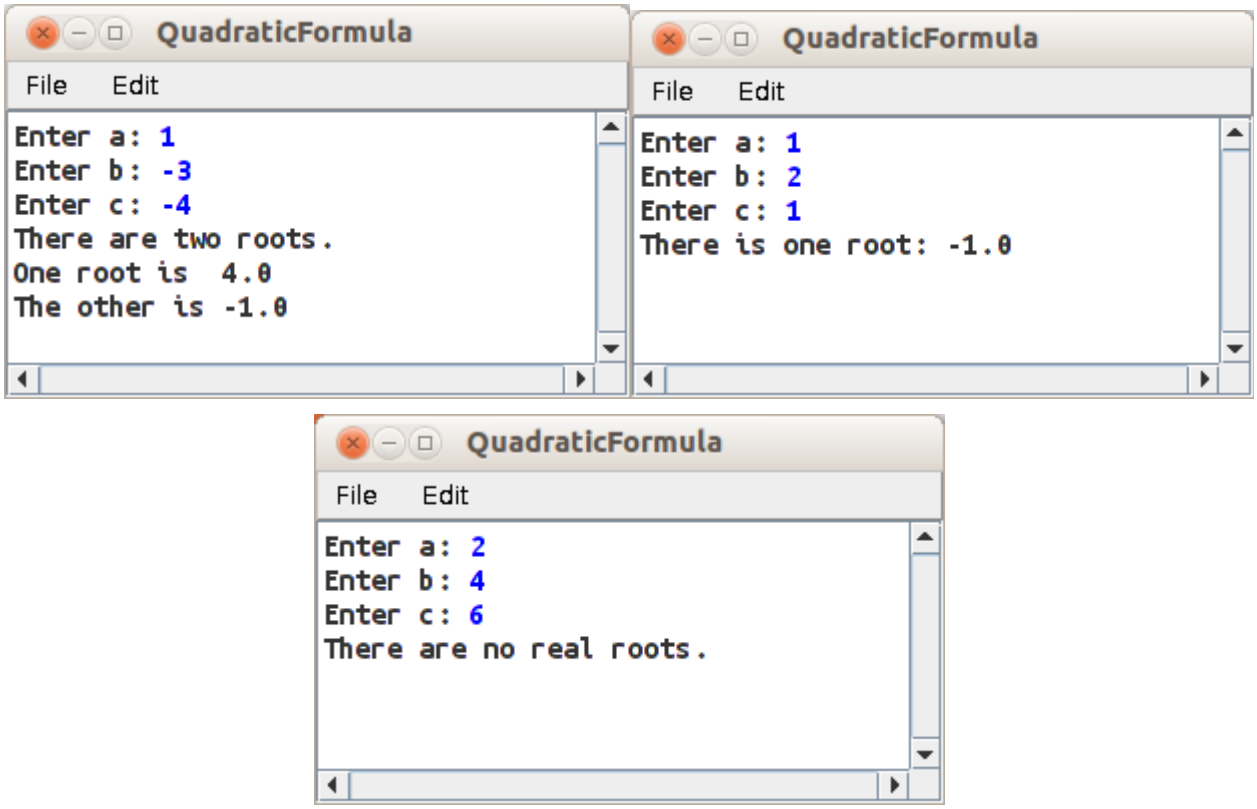
Part One: The Quadratic Formula

A *quadratic equation* is an equation of the form $ax^2 + bx + c = 0$, where a is nonzero. Given the values of a, b, and c, the *quadratic formula* says that the roots of the quadratic equation are given by

$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

The quantity $b^2 - 4ac$ is called the *discriminant*. If it's greater than zero, there are two different roots of the quadratic equation, which are given by the above formula. If it's exactly zero, there's just one root, given in two different ways by the quadratic formula. If it's negative, there are no (real) roots to the equation.

Your job is to write a program that prompts the user for the values of a, b, and c, then prints out the roots of the quadratic equation $ax^2 + bx + c = 0$. Your program should be able to duplicate the following sample run, plus runs with other values:



For reference, to compute the square root of some number x, you can use the `Math.sqrt` function. For example, the code

```
double y = Math.sqrt(x);
```

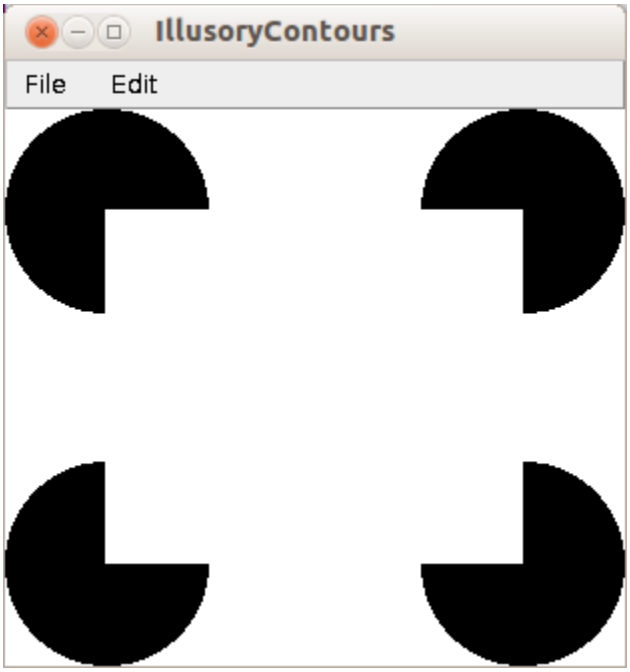
sets `y` to the square root of `x`.

Some notes:

- You can assume that the user doesn't enter 0 as their value for a.
- Aside from the above restriction, the values of a, b, and c can be any real numbers.

Part Two: Illusory Contours

The *illusory contours illusion* is a famous optical illusion that looks like this:



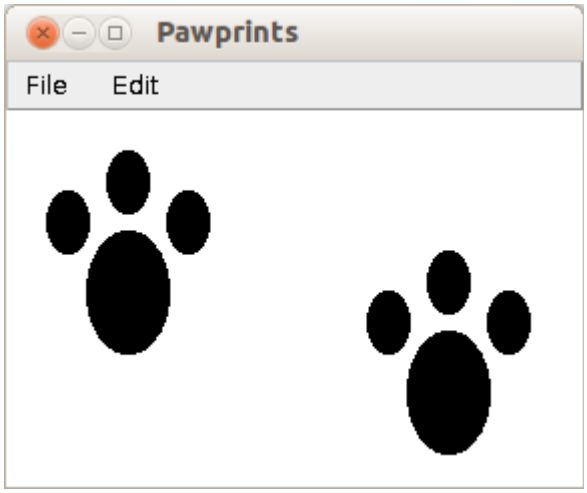
Your job in this part of the assignment is to create a program that draws this illusion. The three-quarters circles should all be the same size and should be flush up against the edges of the window. Additionally, your program should be designed such that it's easy to adjust the sizes of the circles; think about using constants. There are many approaches you can take to draw this figure, some of which are easier than others. You're welcome to choose any approach you'd like.

Part Three: Pawprints

Your task in this part of the assignment is to implement a method that draws a pawprint. In the starter code, we've given you a program with an incomplete implementation of a method called

```
private void drawPawprint(double x, double y)
```

The `x` and `y` parameters to this method give the coordinates of the upper-left corner of the bounding box of the pawprint (similar to how the *x* and *y* coordinates of a `GOval` give the coordinates of the upper-left corner of the oval's bounding box). The pawprint should consist of three toes and a heel, as shown in the picture (which has two pawprints drawn).



The provided starter code contains some constants that specify how big each toe is, how big the heel is, and the relative offset from the upper-left corner of the bounding box to each toe and to the heel. For example, the first toe would be drawn `FIRST_TOE_OFFSET_X` pixels to the right to the specified *x* coordinate and `FIRST_TOE_OFFSET_Y` pixels below the specified *y* coordinate.

The provided starter code's `run` method is set up to call `drawPawprint` in two different locations. If you see the pawprint drawn correctly at those locations, then your method probably works!

Part Four: Tricolor Flags

Many countries, regions, territories, and states have tricolor flags, flags purely consisting of three evenly-sized vertical or horizontal bars. Armenia, Austria, Belgium, Bulgaria, Chad, Estonia, France, Gabon, Germany, Guinea, Hungary, Ireland, Italy, Ivory Coast, Lithuania, Luxembourg, Mali, the Netherlands, Nigeria, Peru, Romania, Russia, South Ossetia, Sierra Leone, and Yemen all have tricolor flags. (I apologize if I missed anything – if I did, let me know!)

Your job is to write a program that draws a tricolor flag of your own choosing centered both vertically and horizontally in the window. It can be a tricolor flag of an actual state or region, or it can be of your own invention. The width and height of the flag should be controlled by the two named constants `FLAG_WIDTH` and `FLAG_HEIGHT`.

You should then add a `GLabel` in the bottom-right corner of the window with the text "Flag of ***x***," where ***x*** is which country or region the flag represents (whether real or fictional). Your `GLabel` should be flush against the bottom-right corner of the window and none of the text should be cut off. To guarantee this, you'll need to compute the position for the label. This will require you to have an understanding of a few different text measurements; check pages 307 – 311 of *The Art and Science of Java* for details, and in particular look into ascent and descent heights.

In the course of this assignment, you might want to use colors not given by the enumerated color constants `Color.RED`, `Color.BLUE`, etc. In Java, you can create custom colors by specifying the amount of red, green, and blue that make up that color. For example, the following code will define a global constant called `CARDINAL_RED`:

```
private static final Color CARDINAL_RED = new Color(196, 30, 58);
```

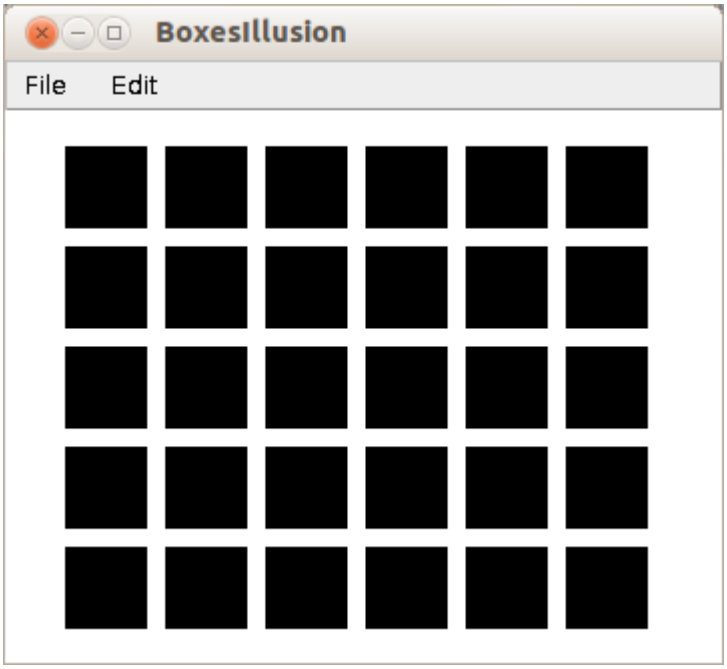
This constant is called `CARDINAL_RED` rather than `Color.CARDINAL_RED`, so if you wanted to set the color of an object to `CARDINAL_RED`, you'd write something to the effect of.

```
obj.setColor(CARDINAL_RED);
```

If you visit Wikipedia's "List of Colors" articles, you can get the amounts of red, green, and blue for all sorts of different colors.

Part Five: The Boxes Illusion

In this problem, you'll write a program that produces an optical illusion. By drawing a grid of black squares with small amounts of spacing in-between them, your brain will trick you into thinking there are small grayish areas in the corners between those squares. You can see this here:

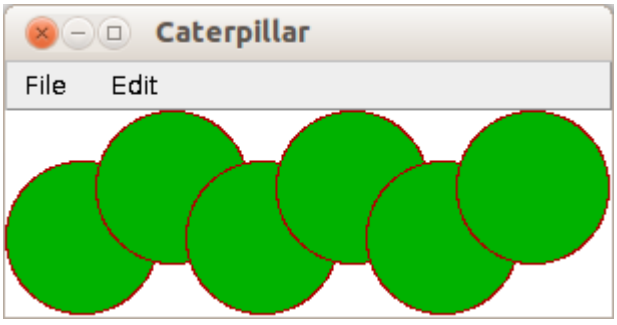


We'd like you to structure your program so that the figure can easily be modified by changing some of the constants in the starter code. Specifically:

- The number of of rows and columns in the figure should be controlled by the `NUM_ROWS` and `NUM_COLS` constants.
- The width and height of each box should be controlled by the `BOX_SIZE` constant.
- The spacing between the boxes should be controlled by the `BOX_SPACING` constant.
- The entire figure should be centered both horizontally and vertically in the window.

Part Six: Caterpillars

Your job in this part of the assignment is to draw a caterpillar made out of a bunch of `GOval`s. You can get a nice caterpillar effect by slightly overlapping the ovals horizontally and shifting half of the ovals down a slight amount from the rest. Notice that the body segments are stacked so that the leftmost oval is at the bottom and the rightmost is at the top.



In the example picture to the right, we fiddled with the `APPLICATION_WIDTH` and `APPLICATION_HEIGHT` constants to get the caterpillar to precisely fill the window. You're not required to do this – you can draw the caterpillar anywhere you want in the window. You're not required to fill the entire canvas, and you don't need to make the caterpillar's size proportional to the size of the window.

The caterpillar you draw should have each circle in the body filled in. Each circle's fill color must be different than its border color. Other than that, you're welcome to choose the colors, sizes, and relative spacing of the body segments however you'd like. We did a fair amount of experimentation to come up with the values used in the image shown here.

As in Part Five of this assignment, you should use named constants in your program to make it easy to change various aspects of the drawing. Specifically, you should use constants to control the horizontal and vertical spacing of the caterpillar's body segments, the caterpillar's colors, and the total number of body segments in the caterpillar. One particular note – your section leader should be able to change the number of body segments in the caterpillar to be either even or odd.

(Optional) Part Seven: Extensions!

Can you think of a way to make one of the programs you wrote more exciting? If so, for extra credit, you're welcome to go above and beyond what we've asked you to do in this assignment.

If you'd like to implement extensions on top of any of these programs, please feel free to do so. To make it easier to grade your assignments, if you do choose to add extensions, please create separate programs for the "base" version of the assignment (what we asked you to do in this handout) and the "extended" version of the assignment (what you chose to do on top.) For example, if you wanted to add extensions to the pawprints program, you could submit your assignment with both a `Pawprints.java` program and an `ExtendedPawprints.java` program.

Advice, Tips, and Tricks

Many of the programs you'll be writing need to work for a variety of user inputs. For example, the quadratic formula program should be able to handle real numbers as inputs (except for $a = 0$). Some of the other programs that you'll be writing need to reference constants defined in your program. One of the major points of defining constants in a program is to make it easier to change your program's behavior simply by adjusting the value assigned to that constant. During testing, we will change the values of the constants in your programs to check whether you have correctly and consistently used constants. Before submitting, check whether or not your programs behave correctly when you vary the values of the constants in the program. It would be a shame if your section leader dropped you from a ✓+ to a ✓ because you had forgotten to test your program on some particular input.

Style is just as important as ever in this assignment. Be sure to follow the style guidelines set out from the Karel assignment – use a top-down design, comment your code as you go, have intelligent method names, and indent your code properly. In addition to this, now that we've added variables, methods, and constants into the mix, you should check your code for the following stylistic issues:

- **Do you have clear names for your variables?** In Java, the convention is that variables should be written in `lowerCamelCase` and should clearly describe what values they represent. Avoid single-letter variable names except in `for` loops or when the single letter actually is an appropriate variable name. Try to be descriptive about what values will be stored in the variable.
- **Do you use methods appropriately?** In many of these programs you will end up writing similar code multiple times. Whenever possible, factor this similar code out into a method and introduce parameters if necessary. This makes code easier to read, maintain, test, and debug.
- **Do you have appropriate inline comments?** Method comments are a great way to make your intentions easier to follow, but it is also important to comment the bodies of your methods as well. Use comments to indicate what task different pieces of the code are trying to perform, or to clarify logic that is not immediately obvious.
- **Do you make appropriate use of constants?** Several of the programs you'll write – especially the graphics programs – will require values that will not be immediately evident from context. When appropriate, introduce constants to increase readability and customizability.
- **Did you update the file comments appropriately?** Java files should begin with a comment describing who wrote the file and what the file contains. Did you update the comments at the top of each Java file with information about your program?

This is not an exhaustive list of stylistic conventions, but it should help you get started. As always, if you have any questions on what constitutes good style, feel free to stop on by the Tresidder LaIR with questions, come visit us during office hours, or email your section leader with questions.

Good luck!