Assignment A4 CS100J Spring 2005 Due Sunday, 13 March, 23:59

This assignment introduces you to graphics. You will write procedures that draw stars, spirals, and bouncing balls in a JFrame. You may work with one other person. If you do so, please form a group for this assignment on the CMS WELL BEFORE YOU SUBMIT YOUR FILES. At the end of this document, we tell you what to submit. You will not use a JUnit testing program because you will be looking at visual output (graphics) to determine correctnesss.

Download class `Turtle` (from here or from the course website), put it in its own directory, and open it in DrJava. A `Turtle` is a pen of a certain color at a pixel `(x, y)` that is pointing in some direction, given by the angle (0 degrees is to the right; 90 degrees, up; 180 degrees, left; and 270 degrees, down). When the turtle is moved to another spot using procedure move, a line is drawn if the pen is currently "down" --if it is "up", nothing is drawn. The pen is initially black, but its color, of class `java.awt.Color`, can be changed. A footnote on page 1.5 of the ProgramLive CD contains information about class `Color`. Study the specification of class `Turtle` (the javadoc files).

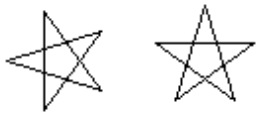
In DrJava, create another file with the following class in it (use the indent-line feature of DrJava to indent the lines appropriately) and save it in the same directory with file <code>Turtle.java</code>.

```
import java.awt.*;
/** Assignment A4: using a Turtle */
public class MyTurtle extends Turtle {
/** Draw a black line 30 pixels to the right and then
a red line 35 pixels down. */
public void drawTwo() {
move(30); // draw a line 30 pixels to the right
addAngle(270); // add 270 degrees to the angle
setColor(Color.red);
move(35);
}
}
```

We give you one method in this class as an example of how graphics works. After compiling class `MyTurtle`, in DrJava's interaction pane, create an instance of class `MyTurtle` and then execute a call on this method and see what happens. A `JFrame` should be created and two lines should be drawn on it.

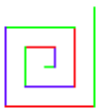
In the interactions pane (or in a method in class `MyTurtle`), draw some things to familiarize yourself with class `Turtle`. After that, perform the following tasks. Put a precise and complete specification on any method you write as a javadoc comment --points will be deducted if you don't. The specification should allow anyone to know precisely what a call on the method does. It must mention all parameters and say what they are for. Look at the javadoc spec to make sure the javadoc comments are appropriate. As usual, your methods must have our names, exactly, and have the same parameters.

Task 1. Write a procedure `drawStar(int d)` that draws a star whose lines have length `d` with the current turtle. An example appears to the right in this paragraph. Don't worry about what angle the turtle is facing when you start drawing the star; just begin by drawing a line of length `d`. The angle between the two lines that make a point is 36 degrees. However, because of the way the drawing is done, to get that angle of 36 degrees in a point, after drawing a line, you have to add 144 degrees to the angle.



Task 2. The orientation of the star drawn by `drawStar` depends on the angle at which the turtle is facing when the drawing starts. Write a procedure `drawStarUp(int d)` that draws a star at the current position of the turtle with the point straight up (so that one line is horizontal), as in the second star on the right.

Task 3: Draw a spiral. The first picture to the right is done by drawing 10 lines, as follows. The first one has length 5; the second, 10; the third, 15, etc. After each line, 90 degrees is added to the angle. The lines alternate among three colors: green, blue, red.



Write a procedure `spiral(int n, int a, int d, int sec)`; that draws `n` lines, adding angle `a` after each one. Line 1 is `d` pixels long, line 2 is `2*d` pixels long, ..., line `i` is `d*i` pixels long. The lines alternate among green, blue, and red. Pause `sec` microseconds after drawing each line.

Then write a procedure `spiralm(int n, int a, int d, int sec)` that does the same thing as `spiral` but starts drawing at the midpoint of the panel, facing east (right).

When you first test your method, use 5 for `d` and 0 for `sec`. Try different angles --90 degrees, 92 degrees, 88 degrees, etc. You can also use `sec` = 500 or `sec` =1000 in order to see the lines drawn one at a time.

You will be amazed at what method `spiral` does. Find out by trying these calls (clear the panel, using procedure `clear()`, before each one --assume `x` is a `MyTurtle` folder):

```
x.spiralm(500, 90, 1, 0);    x.spiralm(500, 135, 1, 0);    x.spiralm(500, 60, 1, 0);
x.spiralm(500, 121, 1, 0);   x.spiralm(500, 89, 1, 0);     x.spiralm(500, 150, 1, 0);
x.spiralm(500, 120, 1, 0);   x.spiralm(500, 119, 1, 0);
```

Task 4: Bouncing balls. Procedure `fillCircle` in `Turtle` lets you draw a disk –a filled-in circle. You can use this method to draw a bouncing ball. Suppose the ball is at some position `(x, y)`. To make it look like the ball is moving, repeat the following process over and over again:

1. Pause for 100 microseconds.
2. Move the ball: (1) Draw the ball at its current position using color white, thus erasing it, (2) change the position of the turtle, and (3) draw the ball in its own color.

(a) Start a new .java file for class `Ball`, which extends `Turtle`. This class needs three (private) fields: `radius` gives the radius of the ball and variables `vx` and `vy` will be the acceleration --when the ball is moved, it moves `vx` pixels in the horizontal direction and `vy` pixels in the vertical direction. Write getter methods for the three fields.

(b) Write a constructor `Ball(x, y, vx, vy, r, c)` that initializes a new `Ball` folder so that the turtle starts at `(x, y)`, has acceleration `(vx, vy)`, has radius `r`, and is drawn with color `c`. You know the constructor works properly when you see the ball in the panel. Create several balls, with different starting points, radii, and colors, before going on to the next step. To make things easier, you might also want to write a constructor `Ball(vx, vy, r)` that starts the turtle at the midpoint of the panel, has acceleration `(vx, vy)`, has radius `r`, and is a black ball.

(c) Write a procedure `moveBallOnce()` that moves the ball once, as given by the acceleration `(vx, vy)` of the ball. Here are some things to think about:

1. To erase the ball, you have to draw it with a white pen. Then you have to move it. And finally you have to draw the ball in its original color. So, you have to remember the original color. Have a local variable `save` to contain the original color and, after drawing the ball white, set the turtle color back to the value of variable `save`.
2. We want the ball to bounce off the sides of the wall. So, if the ball gets too near the top (i.e. the `y`-coordinate of the turtle becomes less than the radius of the ball), then before moving the ball negate the `y`-coordinate acceleration (using `vy= - vy;`). You have to do the same kind of thing for the other three walls.

Test method `moveBallOnce()` carefully. Here is an example of how to test it. In the interactions pane, create a ball `d` of radius 40 that starts in the middle of the panel and has velocity `(0, -30)`. Then, repeatedly execute `d.moveBallOnce()` and watch what happens. Makes sure that it bounces properly off the top and bottom walls. Then do the same kind of test for the left and right walls.

(d) Write a method `inMotion()` that puts the ball perpetually in motion. Its body should be a loop that does not terminate and that has a repetend that (1) pauses for 100 microseconds and then (2) moves the ball once. In the interactions pane, create a new `Ball d`, call `d.inMotion();`, and watch the ball move. The best way to stop this execution is to hit the DrJava reset button.

(e) Write a static method `inMotion(Ball b1, Ball b2)` that puts both balls `b1` and `b2` in motion forever.

Task 5. Something of your own. Write a procedure that does something that you find interesting. Make sure you say what it does in its specification. Place it either in `MyTurtle` or in `Ball`, whichever is most appropriate. We don't care what it does. You could draw a face whose size depends on a parameter. You could make some interesting design with a few stars. You could have the two bouncing balls change direction when they collide (when they occupy the same space). You could have more bouncing balls, and when two collide, the smaller one blows up (goes completely off the screen). You could change the initial color of the panel to something other than white and then put a ball in motion, so that you see the path it takes. How about placing some rectangles at the top of the panel; when a ball hits them, the rectangle disappears and the ball changes direction.We will make the most interesting procedures available on the course website.

What to submit. At the top of class `MyTurtle`, put a comment that says what your interesting procedure is and which class it is in. Submit files `MyTurtle.java` and `Ball.java`.