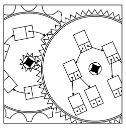
Laboratory 2: A Simple Drawing Program

Objective. To play with some of the features of the `element` package.



SimpleDraw

Discussion. In this lab we enter a pre-written program that allows us to experiment with the features of the `element` package. The program we will be experimenting with allows us to draw a single "curve" in the drawing window. This may seem a little simplistic, but there are many things to be learned even from scribbling on a computer's screen.

Here is the program we will start with:

```
// Lab: Experimenting with interaction in the drawing window.

import element.*;
import java.awt.Color;

public class SimpleDraw
{
    public static void main(String args[])
    {
        DrawingWindow d = new DrawingWindow(200,400);
        Pt mouse;
        final int radius = 2;
        Circle nib = new Circle(0,0,radius);

        d.setForeground(Color.red);
        d.awaitMousePress();

        while (d.mousePressed())
        {
            mouse = d.getMouse();
            nib.center(mouse);
            d.fill(nib);
        }
    }
}
```

First, this program does not use a `ConsoleWindow`. Increasingly, programs in today's world are graphically oriented. Instead, we use the `DrawingWindow` referred to by `d`. Skipping down we see there is the loop

```
while (d.mousePressed())
```

Strictly speaking, we don't know about loops, but this one is simple: it executes the `getMouse`, `nib.center`, and `d.fill` methods as fast as possible, as long as the mouse button is being pressed. In fact, all of the external evidence of the program running is the result of executing those three statements: the program simply draws circles at each point.

Procedure. Enter, run, and modify the `draw` program:

1. Start your programming environment and type in the above program.
2. Compile and run the program. When the program begins to run, the drawing window appears. With a curve in mind, press the mouse and sketch out the curve with the mouse down. When you let go, the program should stop.
3. Run the program and draw your curve by moving the mouse fast. You should see the individual circles that make up the curve. This is a physical representation of the very short length of time it takes for Java to execute one iteration of the loop! If your machine is faster, the circles will be closer together, and the curve will be smoother.
4. Change the `radius` declaration so that it has the value 5. This allows you to draw with a thicker pen.
5. Change the program so that the nib is a thin rectangle whose center point is `mouse`. (Hint: you will have to declare a `Rect` of the appropriate dimensions, and frequently `center` it at `mouse`.).
6. Experiment further with changes to the program (see below), remembering to save your program at the end.

Thought questions. Consider the following questions as you complete the lab:

1. What happens if you put a `d.invertMode()` at the top of your program and change the foreground color to `Color.black`? (Since black is the usual color, you can just comment out the `setForeground` command.)
2. What happens if you use *relative* mouse commands (like `move` and `line`) after having moved *absolutely* to the point `mouse`? Is it possible to have, for example, as slanted nib for italic calligraphy?
3. How would you get the compter to report (say, in a console window), the number of points drawn?
4. How would this help you determine the average length of time it takes for Java to draw a `Circle` at the point `mouse`?
5. What happens if you type `main(args)` as the last line of your program?