Section Handout 4

Based on handouts by Eric Roberts, Patrick Young, and Jeremy Keeshin

Problem One: The Wizard of Java

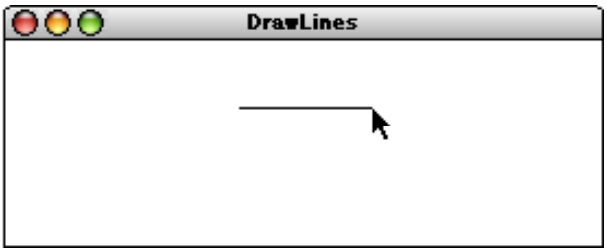
Below is a program that explores what happens when you combine parameter passing, return values, and objects. Trace through the execution of this program. What does it print out?

For convenience, we've added line numbers into this program.

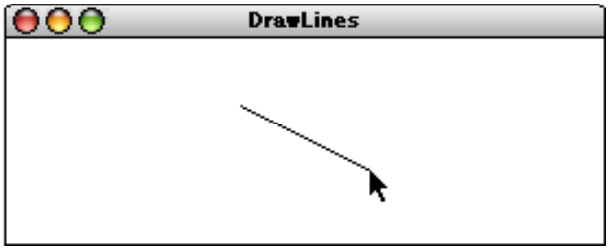
```
/*
 * File: TheWizardOfJava.java
 * A program that explores parameters, return values, primitives, and objects.
 */
import acm.program.*;  // For ConsoleProgram
import acm.graphics.*; // For GPoint

public class TheWizardOfJava extends ConsoleProgram {
 1:     public void run() {
 2:         String dorothy = "Somewhere over the rainbow...";
 3:         GPoint toto = new GPoint(2.718, 3.141);
 4:         int scarecrow = 137;
 5:
 6:         cowardlyLion(dorothy, toto);
 7:         println("dorothy = " + dorothy);
 8:         println("toto.getX() = " + toto.getX());
 9:
10:         tinMan(dorothy, scarecrow);
11:         println("dorothy = " + dorothy);
12:         println("scarecrow = " + scarecrow);
13:
14:         toto = overTheRainbow(dorothy, toto);
15:         println("dorothy = " + dorothy);
16:         println("scarecrow = " + scarecrow);
17:         println("toto.getX() = " + toto.getX());
18:     }
19:
20:     private void cowardlyLion(String dorothy, GPoint toto) {
21:         dorothy += " way up high";
22:         toto = new GPoint(1.61, 98.6);
23:     }
24:
25:     private String tinMan(String dorothy, int scarecrow) {
26:         dorothy = "The scarecrow said " + scarecrow;
27:         scarecrow = 14;
28:         return dorothy;
29:     }
30:
31:     private GPoint overTheRainbow(String dorothy, GPoint toto) {
32:         toto = new GPoint(99.9, 44.4);
33:         dorothy.toUpperCase();
34:         return toto;
35:     }
}
```

Problem Two: Rubber-Banding

Write a `GraphicsProgram` that allows the user to draw lines on the canvas. Pressing the mouse button sets the starting point for the line. Dragging the mouse moves the other endpoint around as the drag proceeds. Releasing the mouse fixes the line in its current position and gets ready to start a new line. For example, suppose that you press the mouse button somewhere on the screen and then drag it rightward an inch, holding the button down. What you'd like to see is the following picture:



If you then move the mouse downward without releasing the button, the displayed line will track the mouse, so that you might see the following picture:



Because the original point and the mouse position appear to be joined by some elastic string, this technique is called *rubber-banding*.

You may find it useful to use the `setEndpoint(double x, double y)` method on the `GLine` type, which sets the endpoint of a line.

Problem Three: Adding Commas to Numeric Strings (Chapter 8, Exercise 13, page 290)

When large numbers are written out on paper, it is traditional (at least in the United States) to use commas to separate the digits into groups of three. For example, the number one million is usually written as 1,000,000. You've probably noticed, though, that if you try to `println(1000000)`, you'll see the number printed as `1000000`, without commas.

To make it easier for programmers to display numbers in this fashion, implement a method

```
private String addCommasToNumericString(String digits)
```

that takes a string of decimal digits representing a number and returns the string formed by inserting commas at every third position, starting on the right. For example, if you were to execute the main program

```
public void run() {
   while (true) {
      String digits = readLine("Enter a numeric string: ");
      if (digits.length() == 0) break;
      println(addCommasToNumericString(digits));
   }
}
```

your implementation of the `addCommasToNumericString` method should be able to produce the following sample run: