Eric Roberts
CS 106A

Handout #27
April 23, 2012

Section Handout #3
Class Definitions and Interactive Graphics

1. Writing a class definition

Using the `Student` class from Chapter 6 (or last Friday's class) as an example, write a class definition for an `Employee` class that keeps track of the following information:

- The name of the employee
- The employee's nine-digit tax id number
- The employee's job title
- A flag indicating whether the employee is still active
- The employee's annual salary

The first two fields should be set as part of the constructor, and it should not be possible for the client to change these values after that. For the other fields, your class definition should provide getters and setters that manipulate those fields. Your class should also define a `toString` method that generates a readable version of the object.

Once you have made this definition, write the code necessary to initialize the following `Employee` objects using the variable names shown:

`ceo`
`Ebenezer Scrooge`
`161803399`
`CEO`
active
£1000

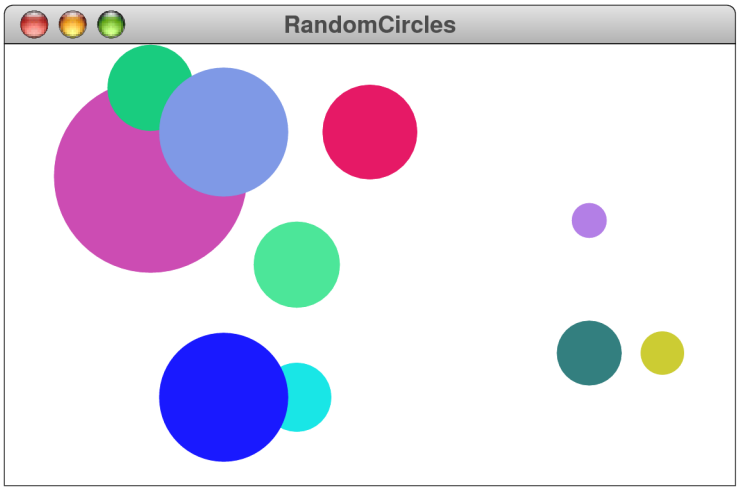
`partner`
`Jacob Marley`
`271828182`
`Former Partner`
inactive
£0

`clerk`
`Bob Cratchit`
`314159265`
`Clerk`
active
£25

Finally, write the code necessary to double Bob Cratchit's salary.

2. Animating circles

In Friday's class, I wrote a program to place 10 random circles on the screen, as follows:



Your mission for this section is to extend that program so that the circles don't simply appear all at once, but instead grow outward from their center so that the radius of the circle increases by one pixel in each time step. For example, if the largest circle near the upper left corner is the first one to appear, it should start as a dot with the same center and then grow to its final size. The program should then grow the next circle, continuing in this fashion until all ten circles are complete.

Before writing this problem, it is probably worth thinking about how you could do a better job of decomposing the program from class.

3. Drawing rectangles

The `DrawLines` program on page 357 shows just how easy it is to write a program to draw lines by dragging the mouse. Interactive drawing programs typically allow you to create other shapes as well. In a typical drawing application, for example, you can create a rectangle by pressing the mouse at one corner and then dragging it to the opposite corner. Thus, if you press the mouse at the location in the left diagram and then drag it to the position in which you see it in the right diagram, the program would create the rectangle shown:

The rectangle grows as you drag the mouse. When you release the mouse button, the rectangle is complete and stays where it is. You can then go back and add more rectangles in the same way.

Although the code for this exercise is quite short, there is one important consideration that you will need to take into account. In the example above, the initial mouse click is in the upper left corner of the rectangle. Your program, however, has to work just as well if you drag the mouse in some other direction besides to the right and down. For example, you should also be able to draw a rectangle by dragging to the left, as shown in the following illustration: