Chris Piech
CS 106A

Section #3
April 25, 2018

# Section Handout #3 — Parameters and Graphics Lab

Portions of this handout by Eric Roberts, Ruchir Rastogi, Guy Blanc, and Julia Daniel

**Warmup questions: True or False?**

a) The value of a local variable named `i` has no direct relationship with that of a variable named `i` in its caller.
b) The value of a parameter named `x` has no direct relationship with that of a variable named `x` in its caller.

## 1. Parameters

Consider the following Java code in a `GraphicsProgram`:

```
/*
 * File: MakeBoxes.java
 * --------------------
 * It's your job to figure out the output of the following code:
 */

import acm.program.*;

public class MakeBoxes extends GraphicsProgram {

      public void run() {
            for (int i = 0; i < 3; i++) {
                  GRect newBox = createBox(i + 1);
                  add(newBox, i * 100, i * 100);
            }
      }

      private GRect createBox(int i) {
            GRect box = new GRect(i * 20, i * 20);
            i += 5;
            return box;
      }
}
```

*Part A:*

Before experimenting with the code in Eclipse, discuss the following questions:

1. What is the value of `i` in `run` at the time `createBox` is first called?
2. What is the value of `i` in `createBox` after it is first called?
3. What is the value of `i` in `run` after `createBox` is first called but before `newBox` is added to the screen?
4. What do you expect the final output to look like?

*Part B:*
Download the lab starter code from the CS106A website and import it into Eclipse. Examine the above code, in `MakeBoxes.java`. Compare the output to what you expected in Part A. Is it the same? If not, discuss why the output is different from what you expected.

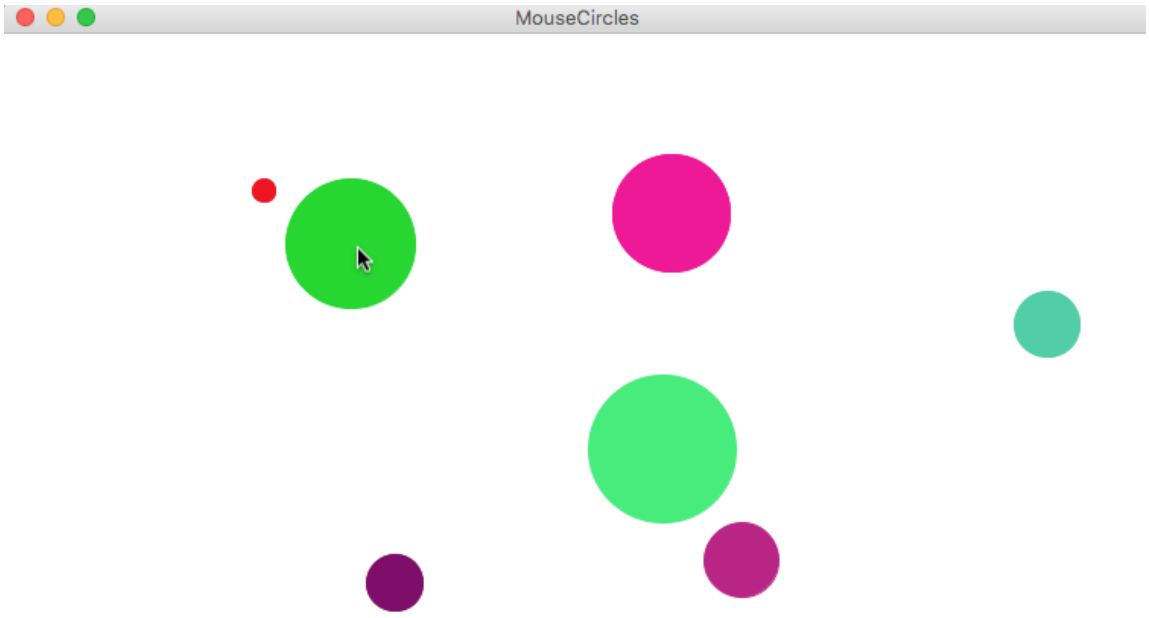
**2. Random Circles**
Write the `RandomCircles` GraphicsProgram, which should draw a set of ten circles with different sizes, positions, and colors. Each circle should have a randomly chosen color, a randomly chosen radius between `MIN_RADIUS` and `MAX_RADIUS` pixels, and a randomly chosen position on the canvas, subject to the condition that the entire circle must fit inside the canvas without extending past the edge. The following sample image shows one possible outcome of a run of this program:


We suggest the following milestones:

1. Implement a program that creates a single random circle, run it multiple times, and check that it works as expected. The following are important checks to verify:
    a. Do the circles generated during different runs have different colors?
    b. Do the circles appear to have radii that vary between `MIN_RADIUS` and `MAX_RADIUS`?
    c. Do the circles ever go off the screen? If the width and height of the window were both 200, and the radius was always, say, 100, where would you expect to see the circles appear? Can you adjust the constants to approximate and test this case?
2. Modify your program to create and draw 10 random circles. Run it a few times. Are you always guaranteed to see 10 circles? Discuss with your partner why or why not.

### 3. Mouse Circles

Write a GraphicsProgram that starts with a blank canvas. Whenever the mouse is clicked, your program should draw a circle centered at the location of the mouse click with a random color and random radius between `MIN_RADIUS` and `MAX_RADIUS`.




We suggest the following breakdown:

1. Write a method that takes in (x, y) coordinates and draws a circle centered at (x, y). For now, use a constant radius and ignore coloring. Check that the method draws circles as you expect it to. Feel free to reuse and modify code from problem 2 if relevant.
2. Use mouse listeners to call the above method when the mouse is clicked. Check that your program works correctly.
3. Modify your original method to choose a random radius between `MIN_RADIUS` and `MAX_RADIUS` and random color. Once again, check that your program works as intended.

*Optional extra challenge*: make the above program do other cool things! Here's one idea: Have a mouse press draw each circle, and then drag that circle around on the screen until the mouse is released.

**4. Hogwarts Tracing**

Consider the following Java code:

```
/*
 * File: Hogwarts.java
 * -------------------
 * This program is just testing your understanding of parameter
 * passing.
 */

import acm.program.*;

public class Hogwarts extends ConsoleProgram {

      public void run() {
            bludger(2001);

      }

      private void bludger(int y) {
            int x = y / 1000;
            int z = (x + y);
            x = quaffle(z, y);
            println("bludger: x = " + x + ", y = " + y + ", z = " + z);
      }

      private int quaffle(int x, int y) {
            int z = snitch(x + y, y);
            y /= z;
            println("quaffle: x = " + x + ", y = " + y + ", z = " + z);
            return z;
      }

      private int snitch(int x, int y) {
            y = x / (x % 10);
            println("snitch: x = " + x + ", y = " + y);
            return y;
      }
}
```

Run the `HogwartsChecker` program to go through the following questions and confirm whether your answer to each question is correct. It is recommended that you stay on a question until you get it right, as an early incorrect answer likely indicates a mistake in tracing that will cause later incorrect answers too.

1. What is the value of the parameter `y` in `bludger`?
2. What is the value of the first parameter, `x`, in `quaffle`?
3. What is the value of the second parameter, `y`, in `quaffle`?
4. What is the value of the first parameter, `x`, in `snitch`?
5. What is the value of the second parameter, `y`, in `snitch`?
6. What is the output of the first `println`?
7. What is the output of the second `println`?
8. What is the output of the third `println`?