Section Handout #3
Parameters, Random Numbers, and Simple Graphics

1. True/False questions

For each of the following statements below, indicate whether it is true or false in Java:

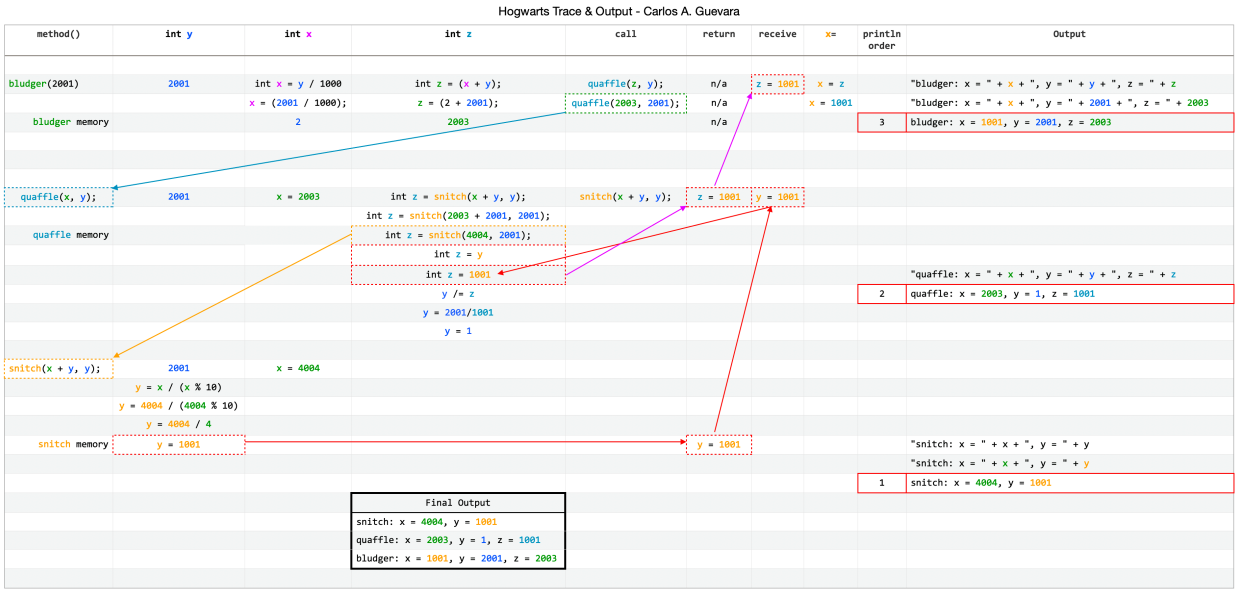
a) The value of a local variable named `i` has no direct relationship with that of a variable named `i` in its caller. **TRUE**

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

2. Tracing method execution

(The following PDF file stored in remote repository proj_assigns/assignment3)

For the program below, trace through its execution by hand to show what output is produced when it runs.



3. **Random circles**
(Code uploaded to remote repository: proj_assigns/assignment3)

Write a GraphicsProgram that draws a set of ten circles with different sizes, positions, and colors. Each circle should have a randomly chosen color, a randomly chosen radius between 5 and 50 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.

On some runs of this program you might not see ten circles. Why?
The RandomGenerator may select WHITE as a color and it will blend in with the background, rendering it invisible. Another possibility is that one circle may be drawn on top of another circle either covering it up or blending in with the same color.

```
package week2;
/*
 * File: RandomCircles.java
 * ------------------------
 * This program draws a set of 10 circles with different sizes,
 * positions, and colors. Each circle has a randomly chosen
 * color, a randomly chosen radius between 5 and 50 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.
 *
 * acm.util.* pkg: https://cs.stanford.edu/people/eroberts/jtf/rationale/
UtilPackage.html
*/
import acm.program.*;
import acm.graphics.*;
import acm.util.*;
import java.awt.*;

public class RandomCirclesFinal extends GraphicsProgram {
	private static final double TOTAL_CIRCLES = 10;
	private static final double MIN_RADIUS = 5;
	private static final double MAX_RADIUS = 50;

	private RandomGenerator randgen = RandomGenerator.getInstance();

	public void run() {
	/*
	 * (x, y) = coordinates for GOval
	 * (r, r) = radius of a circle
	 */
		for (int i = 1; i <= TOTAL_CIRCLES; i++) {
			double r = randgen.nextDouble(MIN_RADIUS, MAX_RADIUS);
			double x = randgen.nextDouble(0, getWidth() - 2 * r);
			double y = randgen.nextDouble(0, getHeight() - 2 * r);

			GOval circle = new GOval(x, y, 2 * r, 2 * r);

			circle.setFilled(true);
			circle.setColor(randgen.nextColor());
			add(circle);
		}
	}
}
```

4. **Drawing lines**
(Code uploaded to remote repository: proj_assigns/assignment3)

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: DrawLines 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: DrawLines Because the original point and the mouse position appear to be joined by some elastic string, this technique is called rubber-banding. Although this program may seem quite powerful, it is also simple to implement. The entire program requires fewer than 20 lines of code.