

## Solution to Section #3

---

Portions of this handout by Eric Roberts and Patrick Young.

### 1. True/False questions

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

**Answer: True**

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

**Answer: True**

### 2. Tracing method execution

The output of `Hogwarts.java` is:

```
snitch: x = 4004, y = 1001
quaffle: x = 2003, y = 1, z = 1001
bludger: x = 1001, y = 2001, z = 2003
```

### 3. Random circles

```
/*
 * 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.
 */

import acm.program.*;
import acm.graphics.*;
import acm.util.*;

public class RandomCircles extends GraphicsProgram {

    /** Number of circles */
    private static final int NCIRCLES = 10;

    /** Minimum radius */
    private static final double MIN_RADIUS = 5;

    /** Maximum radius */
    private static final double MAX_RADIUS = 50;

    public void run() {
        for (int i = 0; i < NCIRCLES; i++) {
            double r = rgen.nextDouble(MIN_RADIUS, MAX_RADIUS);
            double x = rgen.nextDouble(0, getWidth() - 2 * r);
            double y = rgen.nextDouble(0, getHeight() - 2 * r);
            GOval circle = new GOval(x, y, 2 * r, 2 * r);
            circle.setFilled(true);
            circle.setColor(rgen.nextColor());
            add(circle);
        }
    }

    /** Private instance variable */
    private RandomGenerator rgen = RandomGenerator.getInstance();
}

```

## 4. Drawing lines

```
/*
 * File: RubberBanding.java
 * -----
 * This program allows users to create lines on the graphics
 * canvas by clicking and dragging with the mouse. The line
 * is redrawn from the original point to the new endpoint, which
 * makes it look as if it is connected with a rubber band.
 */

import acm.graphics.*;
import acm.program.*;
import java.awt.event.*;

/** This class allows users to drag lines on the canvas */
public class RubberBanding extends GraphicsProgram {

    public void run() {
        addMouseListeners();
    }

    /** Called on mouse press to create a new line */
    public void mousePressed(MouseEvent e) {
        double x = e.getX();
        double y = e.getY();
        line = new GLine(x, y, x, y);
        add(line);
    }

    /** Called on mouse drag to reset the endpoint */
    public void mouseDragged(MouseEvent e) {
        double x = e.getX();
        double y = e.getY();
        line.setEndPoint(x, y);
    }

    /** Private instance variables */
    private GLine line;
}
```