Eric Roberts
CS 106A

Handout #27A
April 23, 2012

Solution to Section #3

1. Employee class

```
/*
 * File: Employee.java
 * -------------------
 * This file defines the Employee class.
 */

/**
 * The Employee class keeps track of the following pieces of
 * data about an employee: the name, the tax id number, the
 * job title, whether the employee is active, and the annual
 * salary.
 */

public class Employee {

/**
 * Creates a new Employee object with the specified name and
 * tax id number.
 * @param name The employee's name as a String
 * @param taxId The employee tax id number as an int
 */
   public Employee(String name, int taxId) {
      employeeName = name;
      taxIdNumber = taxId;
   }

/**
 * Gets the name of this employee.
 * @return The name of this employee
 */
   public String getName() {
      return employeeName;
   }

/**
 * Gets the tax id number of this employee.
 * @return The tax id number of this employee
 */
   public int getTaxIdNumber() {
      return taxIdNumber;
   }

/**
 * Sets the employee's job title.
 * @param title The new job title
 */
   public void setJobTitle(String title) {
      jobTitle = title;
   }

/**
 * Gets the employee's job title.
 * @return The job title
 */
   public String getJobTitle() {
      return jobTitle;
   }
```

```
/**
 * Sets whether the employee is active.
 * @param flag The value true or false indicating active status
 */
   public void setActive(boolean flag) {
      active = flag;
   }

/**
 * Returns whether the employee is active.
 * @return Whether the employee is active
 */
   public boolean isActive() {
      return active;
   }

/**
 * Sets the employee's salary.
 * @param salary The new salary
 */
   public void setSalary(double salary) {
      annualSalary = salary;
   }

/**
 * Gets the annual salary for this employee.
 * @return The annual salary for this employee works
 */
   public double getSalary() {
      return annualSalary;
   }

/**
 * Creates a string identifying this employee.
 * @return The string used to display this employee
 */
   public String toString() {
      String str = employeeName + " (#" + taxIdNumber + ")";
      if (!active) str += " [inactive]";
      return str;
   }

/* Private instance variables */
   private String employeeName; /* The employee's name            */
   private int taxIdNumber;     /* The employee's tax id number   */
   private String jobTitle;     /* The employee's job title       */
   private boolean active;      /* Whether the employee is active */
   private double annualSalary; /* The employee's annual salary   */
}
```

```
/*
 * File: ScroogeAndMarley.java
 * ---------------------------
 * This program tests the implementation of the Employee class.
 */

import acm.program.*;

public class ScroogeAndMarley extends ConsoleProgram {

   public void run() {

      Employee ceo = new Employee("Ebenezer Scrooge", 161803399);
      ceo.setJobTitle("CEO");
      ceo.setActive(true);
      ceo.setSalary(1000);

      Employee partner = new Employee("Jacob Marley", 271828182);
      partner.setJobTitle("Former Partner");
      partner.setActive(false);
      partner.setSalary(0);

      Employee clerk = new Employee("Bob Cratchit", 314159265);
      clerk.setJobTitle("Clerk");
      clerk.setActive(true);
      clerk.setSalary(25);

      println("ceo = " + ceo);
      println("partner = " + partner);
      println("clerk = " + clerk);

      clerk.setSalary(2 * clerk.getSalary());

      println(clerk.getName() + "'s salary = " + clerk.getSalary());
   }

}
```

2. Growing circles

```
/*
 * File: GrowingCircles.java
 * -------------------------
 * This program draws 10 randomly colored circles on the screen, where each
 * circle starts as a dot and then grows until it reaches its final size.
 * The size of the circle is chosen randomly subject to the condition that
 * the entire circle must fit inside the window.
 */

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

public class GrowingCircles extends GraphicsProgram {

   public void run() {
      for (int i = 0; i < N_CIRCLES; i++) {
         animateCircle();
      }
   }

   private void animateCircle() {
      GOval circle = createRandomCircle();
      double radius = circle.getWidth() / 2;
      circle.setSize(0, 0);
      circle.move(radius, radius);
      add(circle);
      while (circle.getWidth() / 2 < radius) {
         growCircle(circle);
         pause(PAUSE_TIME);
      }
   }

   private GOval createRandomCircle() {
      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());
      return circle;
   }

   private void growCircle(GOval circle) {
      double radius = circle.getWidth() / 2 + DELTA_RADIUS;
      circle.setSize(2 * radius, 2 * radius);
      circle.move(-DELTA_RADIUS, -DELTA_RADIUS);
   }

/* Instance variable */
   private RandomGenerator rgen = RandomGenerator.getInstance();

/* Private constants */
   private static final int N_CIRCLES = 10;
   private static final double MIN_RADIUS = 5;
   private static final double MAX_RADIUS = 150;
   private static final double DELTA_RADIUS = 1;
   private static final double PAUSE_TIME = 20;

}
```

3. Drawing rectangles

```
/*
 * File: DrawRectangle.java
 * ------------------------
 * This program allows users to create rectangles on the canvas
 * by clicking and dragging with the mouse.
 */

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

/** This class allows users to drag rectangles on the canvas */
public class DrawRectangle extends GraphicsProgram {

   public void run() {
      addMouseListeners();
   }

/* Called on mouse press to record the starting coordinates */
   public void mousePressed(MouseEvent e) {
      startX = e.getX();
      startY = e.getY();
      currentRect = (GRect) getElementAt(startX, startY);
      dragging = (currentRect != null);
      if (!dragging) {
         currentRect = new GRect(startX, startY, 0, 0);
         currentRect.setFilled(true);
         add(currentRect);
      }
   }

/* Called on mouse drag to reshape the current rectangle */
   public void mouseDragged(MouseEvent e) {
      double x = e.getX();
      double y = e.getY();
      if (dragging) {
         currentRect.move(x - startX, y - startY);
         startX = x;
         startY = y;
      } else {
         x = Math.min(x, startX);
         y = Math.min(y, startY);
         double width = Math.abs(e.getX() - startX);
         double height = Math.abs(e.getY() - startY);
         currentRect.setBounds(x, y, width, height);
      }
   }

/* Private instance variables */
   private GRect currentRect;    /* The current rectangle        */
   private boolean dragging;     /* True if dragging a rectangle */
   private double startX;        /* The initial mouse X position */
   private double startY;        /* The initial mouse Y position */

}
```