Chris Piech
CS 106A

Section #7
Feburary 28, 2018

Solution to Section #7

Portions of this handout by Eric Roberts and Nick Troccoli

1. Word Cloud

```
/**
 * File: WordCloud.java
 * --------------------
 * This program allows the user to create a set of labels and then drag
 * them around in the window.
 */

import acm.graphics.*;
import acm.program.*;
import java.util.*;
import java.awt.event.*;
import javax.swing.*;

public class WordCloud extends GraphicsProgram {

      public void init() {
            contents = new HashMap<String,GLabel>();
            createController();
            addActionListeners();
            addMouseListeners();
      }

      /* Creates the control strip at the bottom of the window */
      private void createController() {
            nameField = new JTextField(MAX_NAME);
            nameField.addActionListener(this);   // Detects ENTER key pressed
            addButton = new JButton("Add");
            removeButton = new JButton("Remove");
            clearButton = new JButton("Clear");
            add(new JLabel("Name"), SOUTH);
            add(nameField, SOUTH);
            add(addButton, SOUTH);
            add(removeButton, SOUTH);
            add(clearButton, SOUTH);
      }

      /* Adds a label with the given name at the center of the window */
      private void addLabel(String name) {
            GLabel label = new GLabel(name);
            double labelX = getWidth() / 2.0 - label.getWidth() / 2.0;
            double labelY = getHeight() / 2 + label.getAscent() / 2.0;
            add(label, labelX, labelY);
            contents.put(name, label);
      }

      /* Removes all labels in the contents table */
      private void removeContents() {
            for (String labelName : contents.keySet()) {
                  remove(contents.get(labelName));
            }
            contents.clear();       // Clear all entries in the hashmap
      }

      /* Called in response to button actions */
      public void actionPerformed(ActionEvent e) {
            Object source = e.getSource();
            // Detect both clicks and ENTER for adding a new label
            if (source == addButton || source == nameField) {
                  addLabel(nameField.getText());
            } else if (source == removeButton) {
                  String text = nameField.getText();
                  if (contents.containsKey(text)) {
                        remove(contents.get(text));
                        contents.remove(text);
                  }
            } else if (source == clearButton) {
                  removeContents();
            }
      }

      /* Called on mouse press to record the coordinates of the click */
      public void mousePressed(MouseEvent e) {
            last = new GPoint(e.getPoint());
            currentLabel = (GLabel)getElementAt(last);
      }

      /* Called on mouse drag to reposition the object */
      public void mouseDragged(MouseEvent e) {
            if (currentLabel != null) {
                  currentLabel.move(e.getX() - last.getX(),
                        e.getY() - last.getY());
                  last = new GPoint(e.getPoint());
            }
      }

      /* Private constants */
      private static final int MAX_NAME = 25;

      /* Private instance variables */
      private HashMap<String,GLabel> contents;
      private JTextField nameField;
      private JButton addButton;
      private JButton removeButton;
      private JButton clearButton;
      private GLabel currentLabel;
      private GPoint last;
}
```

2. Interactive Karel

```
/*
 * File: InteractiveKarel.java
 * --------------------
 * This program lets the user control Karel as it moves and turns
 * within the canvas window.
 */

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

/* Simulates a simplified Karel the Robot through use of GUI interactors. */
public class InteractiveKarel extends GraphicsProgram {

    /* The number of pixels wide/tall for the Karel images */
    private static final int KAREL_SIZE = 64;

    /* The image of Karel currently displayed on the canvas. */
    private GImage karel;

    /* The direction (NORTH, SOUTH, EAST, WEST) Karel is facing. */
    private String direction;

    /* Sets up GUI components and Karel's initial image. */
    public void init() {
        add(new JButton("move"), SOUTH);
        add(new JButton("turnLeft"), SOUTH);
        addActionListeners();
    }

    /* Add our graphics once the canvas is onscreen. */
    public void run() {
        karel = new GImage("KarelEast.jpg");
        direction = EAST;
        add(karel, 0, 0);
    }

    /* When we get an interaction, update Karel accordingly. */
    public void actionPerformed(ActionEvent event) {
        String command = event.getActionCommand();
        if (command.equals("move")) {
            moveKarel();
        } else if (command.equals("turnLeft")) {
            turnLeftKarel();
        }
    }

    /* Moves Karel one step in the current direction. */
    private void moveKarel() {
        double newX = karel.getX();
        double newY = karel.getY();
        if (direction.equals(NORTH)) {
            newY -= KAREL_SIZE;
        } else if (direction.equals(SOUTH)) {
            newY += KAREL_SIZE;
        } else if (direction.equals(EAST))  {
            newX += KAREL_SIZE;
        } else if (direction.equals(WEST))  {
            newX -= KAREL_SIZE;
        }

        if (isKarelOnScreen(newX, newY)) {
            karel.setLocation(newX, newY);
        }
    }

    /* Causes Karel to turn 90 degrees to the left (counter-clockwise). */
    private void turnLeftKarel() {
        if (direction.equals(NORTH)) {
            direction = EAST;
        } else if (direction.equals(EAST)) {
            direction = SOUTH;
        } else if (direction.equals(SOUTH))  {
            direction = WEST;
        } else if (direction.equals(WEST))  {
            direction = NORTH;
        }

        karel.setImage("Karel" + direction + ".jpg");
    }

    /* Returns whether Karel would be on-screen at the given x/y position. */
    private boolean isKarelOnScreen(double x, double y) {
        return x >= 0 && y >= 0 && x + KAREL_SIZE <= getWidth()
                && y + KAREL_SIZE <= getHeight();
    }
}
```

3. The `Employee` Class

```
/*
 * File: Employee.java
 * -------------------
 * Class which describes the Employee variable type.
 * An Employee has the following information:
 *      - name
 *      - title
 *      - annual salary
 *
 * They may be given a promotion, which adds the word "Senior"
 * to their job title and doubles their salary.
 */
public class Employee {

    public Employee(String newName, String newTitle) {
        name = newName;
        title = newTitle;
    }

    public String getTitle() {
        return title;
    }

    public void setTitle(String title) {
        this.title = title;
    }

    public int getSalary() {
        return salary;
    }

    public void setSalary(int salary) {
        this.salary = salary;
    }

    public String getName() {
        return name;
    }

    // Adds "Senior" to the front of our job title, and doubles our salary
    public void promote() {
        title = "Senior " + title;
        salary *= 2;
    }

    /* Employee instance variables */
    private String name;
    private String title;
    private int salary;
}
```

4. Subclassing `GCanvas`

```
/*
* File: RandomCirclesCanvas.java
* ------------------------
* This GCanvas subclass adds the ability to also draw random circles.
* 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.graphics.*;
import acm.util.*;

public class RandomCirclesCanvas extends GCanvas {

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

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

   public void drawRandomCircle() {
      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);   // adds it to ourself!
   }

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