Alisha Adam & Rohit Talreja
CS 106A – Summer 2016

Solutions for Section #7

If you'd like to try running these programs, download a starter project (including any necessary data files/images) here: http://web.stanford.edu/class/cs106a/section7-code.zip

1. Remove Even Length

```
private void removeEvenLength(Iterator<String> it) {
    while (it.hasNext()) {
        String str = it.next();
        if (str.length() % 2 == 0) {
            it.remove();
        }
    }
}
```

Note: You may recall that we wrote a similar method last week by looping over an ArrayList, rather than using an iterator. We wrote this version just to demonstrate the use of iterators, which might come in handy soon (hint: on the last homework assignment).

2. Window Coloring

```
/* This program allows the user to type a color name and have that become the
 * background color of the window. It uses a large data file of color names.
 */
import acm.program.*;
import java.io.*;
import java.util.*;
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;

public class ColoredWindow extends GraphicsProgram {
    private static final int NUM_COLUMNS = 16;   // number of columns in text box
    private static final String COLORS_FILE = "res/colors.txt";  // color file to read

    private JTextField colorNameEntry;       // text field used for data entry
    private HashMap<String, Color> colors;   // color data from file

    public void init() {
        readColors();
        addInteractors();
    }

    /* Adds the interactors and event listeners to the window. */
    private void addInteractors() {
        colorNameEntry = new JTextField(NUM_COLUMNS);
        add(new JLabel("Enter color: "), SOUTH);
        add(colorNameEntry, SOUTH);

        // listen for when the user presses enter while in the box
        colorNameEntry.addActionListener(this);
    }
```

```java

    /* When the user types Enter, look up the current color. */
    public void actionPerformed(ActionEvent e) {
        String colorName = colorNameEntry.getText().toLowerCase(); // case-insensitive
        Color chosenColor = colors.get(colorName);
        if (chosenColor != null) {
            setBackground(chosenColor);
        }
    }

    /* Read the color data from the file into a map of (name -> Color) */
    private void readColors() {
        colors = new HashMap<String, Color>();
        try {
            Scanner sc = new Scanner(new File(COLORS_FILE));
            while (sc.hasNext()) {
                String colorName = sc.nextLine().toLowerCase(); // normalize case
                String rgbValues = sc.nextLine();
                Scanner tokens = new Scanner(rgbValues);
                int r = tokens.nextInt();
                int g = tokens.nextInt();
                int b = tokens.nextInt();
                Color c = new Color(r, g, b);
                colors.put(colorName, c);
            }
        } catch (FileNotFoundException e) {
            println("Couldn't load color file");
        }
    }
}
```

3. DIY Karel

```java
/* Simulates a simplified Karel the Robot through use of GUI interactors. */
import acm.graphics.*;
import acm.program.*;
import java.awt.event.*;
import javax.swing.*;

public class InteractiveKarel extends GraphicsProgram {
    private static final int KAREL_SIZE = 64;   // Karel's size in px
    private GImage karel;   // the current Karel image
    private String dir;     // Karel's direction: “North”, “South”, “East”, or “West”

    /* Sets up GUI components and Karel's initial image. */
    public void init() {
        karel = new GImage("res/KarelEast.png", 0, 0);
        dir = “East”;
        add(karel);
        add(new JButton("move"), SOUTH);
        add(new JButton("turnLeft"), SOUTH);
        addActionListeners();
    }

    /* When we get a command, update Karel's position/direction accordingly. */
    public void actionPerformed(ActionEvent event) {
        String command = event.getActionCommand();
        if (command.equals("move")) {
            move();
        } else if (command.equals("turnLeft")) {
            turnLeft();
        }
    }

    /* Moves Karel one step in the right direction. */
    private void move() {
        double newX = karel.getX();
        double newY = karel.getY();
        if (dir.equals(“North”))      { newY -= KAREL_SIZE; }
        else if (dir.equals(“South”)) { newY += KAREL_SIZE; }
        else if (dir.equals(“East”))  { newX += KAREL_SIZE; }
        else                          { newX -= KAREL_SIZE; }
        if (isKarelOnScreen(newX, newY)) {
            karel.setLocation(newX, newY);
        }
    }

    /* Causes Karel to turn 90 degrees to the left (counter-clockwise). */
    private void turnLeft() {
        if (dir.equals(“North”))      { dir = “West”; }
        else if (dir.equals(“East”))  { dir = “North”; }
        else if (dir.equals(“South”)) { dir = “East”; }
        else                          { dir = “South”; }
        karel.setImage("res/Karel" + dir + ".png");   // e.g. "KarelNorth.png"
    }

    /* 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();
    }
}
```