Nick Troccoli
CS 106A

Section #7
August 9, 2017

# Section Handout #7: Interactors and Data Structures

Parts of this handout by Mehran Sahami, Eric Roberts, Marty Stepp, and Patrick Young

## 1. Colored Window

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

/* 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.
 */
public class ColoredWindow extends GraphicsProgram {
    /* Private constants */
    private static final int TEXT_FIELD_WIDTH = 16;
    private static final String COLORS_FILE = "res/colors.txt";

    /* Private fields */
    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() {
        add(new JLabel("Enter color: "), SOUTH);
        colorNameEntry = new JTextField(TEXT_FIELD_WIDTH);
        colorNameEntry.setActionCommand("Show");
        add(colorNameEntry, SOUTH);

        add(new JButton("Show"), SOUTH);
        add(new JButton("Random"), SOUTH);
        addActionListeners();

        colorNameEntry.addActionListener(this); // listen for ENTER
pressed
    }

    /* Triggered when the user enters a color or clicks "Random". */
    public void actionPerformed(ActionEvent e) {
        if (e.getActionCommand().equals("Random")) {
            // Pick a random color name - first convert all keys to an
ArrayList
            ArrayList<String> colorNames = new
ArrayList<String>(colors.keySet());
```

```
            int randomIndex = RandomGenerator.getInstance().nextInt(0,
                colorNames.size());
            String colorName = colorNames.get(randomIndex);
            colorNameEntry.setText(colorName);
            setBackground(colors.get(colorName));
        } else {
            // Get the (case-insensitive) color entered and display it,
if valid
            String colorName = colorNameEntry.getText().toLowerCase();
            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");
        }
    }
}
```

## 2. Word Cloud

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

/* This program allows the user to create a set of labels and then drag
 * them around in the window.
 */
public class WordCloud extends GraphicsProgram {
    /* Private constants */
    private static final int MAX_NAME = 25;

    /* Private fields */
    private HashMap<String,GLabel> contents;
```

```
    private JTextField nameField;
    private GLabel currentLabel;
    private GPoint last;

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

    /* Creates the control strip at the bottom of the window */
    private void addInteractors() {
        add(new JLabel("Name"), SOUTH);
        nameField = new JTextField(MAX_NAME);
        add(nameField, SOUTH);

        add(new JButton("Add"), SOUTH);
        add(new JButton("Remove"), SOUTH);
        add(new JButton("Clear"), SOUTH);

        addActionListeners();
    }

    /* Adds a label with the given name at the center of the window */
    private void addLabel(String name) {
        GLabel label = new GLabel(name);
        label.setFont(new Font("Helvetica", Font.BOLD, 18));
        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) {
        String labelName = nameField.getText();

        // Detect both clicks and ENTER for adding a new label
        if (e.getActionCommand().equals("Add")) {
            addLabel(labelName);
        } else if (e.getActionCommand().equals("Remove")) {
            if (contents.containsKey(labelName)) {
                remove(contents.get(labelName));
            }
        } else if (e.getActionCommand().equals("Clear")) {
            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());
        }
    }
}
```

### 3. Stanford Speak

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

/* Allows the user to type in a phrase, then translates the phrase to
its Stanford
 * abbreviation.
 */
public class StanfordSpeak extends Program {
    /* Private constants */
    private static final int TEXT_FIELD_WIDTH = 20;

    /* Private fields */
    private JTextField input;
    private JLabel label;

    public void init() {
        setSize(500, 100);
        add(new JLabel("Enter a phrase: "), NORTH);
        input = new JTextField(TEXT_FIELD_WIDTH);
        input.setActionCommand("Translate");
        input.addActionListener(this);
        add(input, NORTH);
        add(new JButton("Translate"), NORTH);

        label = new JLabel("");      // label for displaying "Stanford
Speak" later
        label.setFont(new Font("Courier", Font.BOLD, 40));
        add(label, SOUTH);
        addActionListeners();
    }

    /* Triggered when the user clicks "Translate" or presses ENTER. */
    public void actionPerformed(ActionEvent event) {
        String phrase = input.getText();
        String abbrev = createStanfordAbbreviation(phrase);
```

```
        label.setText(abbrev);
    }

    /* Translates phrase by individually translating each word. */
    private String createStanfordAbbreviation(String phrase) {
        String result = "";
        Scanner scan = new Scanner(phrase);
        while (scan.hasNext()) {
            String word = scan.next();
            result += abbreviateWord(word);
        }
        return result;
    }

    /* We said you can assume that this method is provided. The methods
     below are included for completeness. */

    private String abbreviateWord(String word) {
        word = word.toLowerCase();
        int vp = findFirstVowel(word);
        if (vp == -1) return word;  // return same word if no vowels
        String result = word.substring(0, vp + 1);
        if (vp + 1 < word.length() && isSonorant(word.charAt(vp + 1)))
            result += word.charAt(vp + 1);
        // convert the first letter in the word to uppercase
        result = Character.toUpperCase(result.charAt(0)) +
result.substring(1);
        return result;
    }

    private int findFirstVowel(String str) {
        for (int i = 0; i < str.length(); i++) {
            if (isEnglishVowel(str.charAt(i)))
                return i;
        }
        return -1;
    }

    private boolean isEnglishVowel(char ch) {
        return ch == 'a' || ch == 'e' || ch == 'i' || ch == 'o' || ch
== 'u';
    }

    private boolean isSonorant(char ch) {
        return ch == 'm' || ch == 'n';
    }
}
```

## 4. Data structure design

```
/*
 * File: ExpandableArray.java
 * --------------------------
 * This class provides methods for working with an array that expands
 * to include any positive index value supplied by the caller.
```

```
 */

public class ExpandableArray {

/**
 * Creates a new expandable array with no elements.
 */
   public ExpandableArray() {
      array = new Object[0];  // Allows us to check length of array
                              // even when no elements exist
   }


/**
 * Sets the element at the given index position to the specified.
 * value. If the internal array is not large enough to contain that
 * element, the implementation expands the array to make room.
 */
   public void set(int index, Object value) {
      if (index >= array.length) {

         // Create a new array that is large enough
         Object[] newArray = new Object[index + 1];

         // Copy all the existing elements into new array
         for (int i = 0; i < array.length; i++) {
            newArray[i] = array[i];
         }

         // Keep track of the new array in place of the old array
         array = newArray;
      }
      array[index] = value;
   }


/**
 * Returns the element at the specified index position, or null if
 * no such element exists.  Note that this method never throws an
 * out-of-bounds exception; if the index is outside the bounds of
 * the array, the return value is simply null.
 */
   public Object get(int index) {
      if (index >= array.length) return null;
      return array[index];
   }


/* Private instance variable */
   private Object[] array;

}
```