Name_____

CS108-Section 2 Fall 2017

Grading Rubric Program 8

Criteria	Proficient	Competent	Novice	Unsatisfactory
Delivery 5	The program was delivered on time to edoras:/handin/Prog8 and an exact copy submitted in class.			Code was not delivered to edoras or in class.
Identification 5	The identification line displays in each program and contains program assignment, full name, and mascID.			The identification line is not present or is incomplete.
Documentation & Readability 10	The documentation is well written and clearly explains what the code is accomplishing and how. The code is exceptionally well organized and very easy to follow and conforms to best practices	The documentation consists of embedded comment and some simple header documentation that is somewhat useful in understanding the code. The code is fairly easy to read and conforms to most best practices	The documentation is simply comments embedded in the code with some simple header comments separating routines. The code is readable only to grader who has the assignment description.	The documentation is simply comments embedded in the code and does not help the reader understand the code. The code is quite difficult to read and poorly organized.
Correctness & Creativity 25	The program works and meets all of the requirements and sample output is provided that fully demonstrates the code works correctly**. Provides a creative solution. (25)**	The program works and meets all or most of the requirements and sample output is provided that demonstrates all or some code works correctly. **Solution is average. (24-15)**	The program works and meets some of the requirements and sample output is provided that demonstrates some of the correct functionality and/or no sample output provided. **Solution is just acceptable. (14-1)**	The program is producing incorrect results or no sample output provided. **Solution is unacceptable. (0)**

```java
import java.awt.GridBagConstraints;
import java.awt.GridBagLayout;
import java.awt.Insets;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import javax.swing.JButton;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JScrollPane;
import javax.swing.JTextArea;
import javax.swing.JTextField;

import java.util.List;
import realtimeweb.simplebusiness.domain.*;
import realtimeweb.simplebusiness.*;

/**
 * Program 8
 *
 * The program will simulating YELP application
 * by using data from online API
 *
 * CS108-02
 * 12.07.17
 *
 * @author Quang Trinh
 */
@SuppressWarnings("serial")
public class YelpByRating extends JFrame implements ActionListener {

    private JTextArea outputArea;                  // Displays search results
    private JButton searchButton;                  // Triggers search button
    private JButton sortButton;                   // Triggers sort button
    private JTextField cityField;                 // Holds city
    private JTextField kindField;                 // Holds kind

    /**
     * Constructor creates GUI components and add
     * GUI components using a GridBagLayout.
     */
    YelpByRating() {
        GridBagConstraints layoutConst = null; // Used to specify GUI component layout
        JScrollPane scrollPane = null;         // Container that adds a scroll bar
        JLabel cityLabel = null;               // Label for city
        JLabel kindLabel = null;               // Label for kind
        JLabel outputLabel = null;             // Label for search results

        // Set frame's title
        setTitle("Yelp simulator by Quang Trinh");

        // Create labels
        cityLabel = new JLabel("Location:");
        kindLabel = new JLabel("Term:");
        outputLabel = new JLabel("Results:");

        // Create output and add it to scroll pane
        outputArea = new JTextArea(30,25);
        scrollPane = new JScrollPane(outputArea);
        outputArea.setEditable(false);

        // Create search button
        searchButton = new JButton("Search");
        searchButton.addActionListener(this);
```

```
64.
65.         // Create sort button
66.         sortButton = new JButton("Sort");
67.         sortButton.addActionListener(this);
68.
69.         // Create city field
70.         cityField = new JTextField(10);
71.         cityField.setEditable(true);
72.         cityField.setText("City, State");
73.
74.         // Create kind field
75.         kindField = new JTextField(10);
76.         kindField.setEditable(true);
77.         kindField.setText("Looking for...");
78.
79.         // Use a GridBagLayout
80.         setLayout(new GridBagLayout());
81.
82.         // Cell(0,0)
83.         layoutConst = new GridBagConstraints();
84.         layoutConst.insets = new Insets(5, 10, 5, 1);
85.         layoutConst.anchor = GridBagConstraints.LINE_END;
86.         layoutConst.gridx = 0;
87.         layoutConst.gridy = 0;
88.         add(cityLabel, layoutConst);
89.
90.         // Cell(1,0)
91.         layoutConst = new GridBagConstraints();
92.         layoutConst.insets = new Insets(10, 1, 5, 10);
93.         layoutConst.fill = GridBagConstraints.HORIZONTAL;
94.         layoutConst.gridx = 1;
95.         layoutConst.gridy = 0;
96.         add(cityField, layoutConst);
97.
98.         // Cell(0,1)
99.         layoutConst = new GridBagConstraints();
100.                layoutConst.insets = new Insets(5, 10, 5, 1);
101.                layoutConst.anchor = GridBagConstraints.LINE_END;
102.                layoutConst.gridx = 0;
103.                layoutConst.gridy = 1;
104.                add(kindLabel, layoutConst);
105.
106.                // Cell(1,1)
107.                layoutConst = new GridBagConstraints();
108.                layoutConst.insets = new Insets(5, 1, 5, 10);
109.                layoutConst.fill = GridBagConstraints.HORIZONTAL;
110.                layoutConst.gridx = 1;
111.                layoutConst.gridy = 1;
112.                add(kindField, layoutConst);
113.
114.                // Cell(2,0)
115.                layoutConst = new GridBagConstraints();
116.                layoutConst.insets = new Insets(5, 5, 5, 10);
117.                layoutConst.fill = GridBagConstraints.BOTH;
118.                layoutConst.gridx = 2;
119.                layoutConst.gridy = 0;
120.                add(searchButton, layoutConst);
121.
122.                // Cell(2,1)
123.                layoutConst = new GridBagConstraints();
124.                layoutConst.insets = new Insets(2, 5, 2, 10);
125.                layoutConst.fill = GridBagConstraints.BOTH;
126.                layoutConst.gridx = 2;
```

```
127.                layoutConst.gridy = 1;
128.                add(sortButton, layoutConst);
129.
130.                // Cell(0,3)
131.                layoutConst = new GridBagConstraints();
132.                layoutConst.insets = new Insets(10, 10, 1, 10);
133.                layoutConst.fill = GridBagConstraints.HORIZONTAL;
134.                layoutConst.gridx = 0;
135.                layoutConst.gridy = 3;
136.                add(outputLabel, layoutConst);
137.
138.                // Cell(0,4)
139.                layoutConst = new GridBagConstraints();
140.                layoutConst.insets = new Insets(1, 10, 10, 10);
141.                layoutConst.fill = GridBagConstraints.HORIZONTAL;
142.                layoutConst.gridx = 0;
143.                layoutConst.gridy = 4;
144.                layoutConst.gridwidth = 3;
145.                add(scrollPane, layoutConst);
146.            }
147.
148.            @Override
149.            public void actionPerformed(ActionEvent event) {
150.                String cityInput = "";    // User city input
151.                String kindInput = "";   // User kind input
152.
153.                JButton sourceEvent = (JButton)event.getSource();
154.
155.                if (sourceEvent == searchButton) {
156.                    // Get input from fields
157.                    cityInput = cityField.getText();
158.                    kindInput = kindField.getText();
159.
160.                    // Clear the output text area
161.                    outputArea.setText("");
162.
163.                    // Get data from Online API
164.                    SimpleBusiness yelp = new SimpleBusiness();  // RealTime Data
165.
166.                    List<Business> businesses = yelp.search(kindInput, cityInput);
167.
168.                    for (Business b : businesses) {
169.                        System.out.println(b.getName() + " " + b.getRating());
170.                        outputArea.append(b.getName() + "  " + b.getRating() + "\n");
171.                        outputArea.append("-----------------------------------------\n");
172.                    }
173.                }
174.                else if (sourceEvent == sortButton) {
175.                    // Get input from fields
176.                    cityInput = cityField.getText();
177.                    kindInput = kindField.getText();
178.
179.                    // Clear the output text area
180.                    outputArea.setText("");
181.
182.                    // Get data from Online API
183.                    SimpleBusiness yelp = new SimpleBusiness();  // RealTime Data
184.
185.                    List<Business> businesses = yelp.search(kindInput, cityInput);
186.
187.                    // Sort the list
188.                    selectionSort(businesses, businesses.size() - 1);
189.
```

```
                    for (Business b : businesses) {
                        System.out.println(b.getName() + " " + b.getRating());
                        outputArea.append(b.getName() + "  " + b.getRating() + "\n");
                        outputArea.append("-----------------------------------------\n");
                    }
                }

                return;
            }

            /**
             * Sorting Business Object Rating descending by selection sort algorithm
             * @param businesses
             * @param size
             */
            private static void selectionSort(List<Business> businesses, int size) {
                int i = 0;
                int j = 0;
                int indexLargest = 0;
                Business temp = null;

                for (i = 0; i < size; ++i) {
                    indexLargest = i;
                    for (j = i + 1; j < size; ++j) {
                        if(businesses.get(j).getRating() > businesses.get(indexLargest).getRating()) {

                            indexLargest = j;
                        }
                    }
                    // Swap 2 objects
                    temp = businesses.get(i);
                    businesses.set(i, businesses.get(indexLargest));
                    businesses.set(indexLargest, temp);
                }
            }

            /**
             * Return author's identification
             * @return
             */
            public static String getIdentificationString() {
                return ("Quang Trinh - Program 8");
            }

            public static void main(String[] args) {

                YelpByRating myFrame = new YelpByRating();

                myFrame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
                myFrame.pack();
                myFrame.setVisible(true);

                return;
            }

        }
```

SCREENSHOTS

Button Pressed