COE 593 – COE Application

Welcome to the Midterm Exam
Tuesday November 20, 2012

Instructor: Wissam F. Fawaz

Name: ___Solution Key_____

Student ID: _____

Instructions:

1. This exam is **Closed Book**. Please do not forget to write your name and ID on the first page.
2. You have exactly **90 minutes** to complete the 4 required problems.
3. Read each problem carefully. If something appears ambiguous, please write your assumptions.
4. Points allocated to each problem are shown in square brackets.
5. Put your answers in the space provided only. No other spaces will be graded or even looked at.

Good Luck!!

Problem 1: Event handling (**20 minutes**) [30 points]

1. Design and implement a Java program that displays a button and a label. Every time the button is pushed, the label should display a random number between 1 (inclusive) and 100 (inclusive).

```
//********************************************************
//  RandomGUI.java       Author: Lewis/Loftus
//********************************************************
import java.awt.Dimension;
import java.awt.event.*;
import javax.swing.*;
import java.util.Random;
public class RandomGUI
{
   private int WIDTH = 100;
   private int HEIGHT = 75;
   private int MAX_NUMBER = 100;
   private JFrame frame;
   private JPanel panel;
   private JLabel numberLabel;
   private JButton randomButton;
   private Random generator;

   //-----------------------------------------------------------------
   //  Sets up the GUI.
   //-----------------------------------------------------------------
   public RandomGUI()
   {
      frame = new JFrame ("Random Display");
      frame.setDefaultCloseOperation (JFrame.EXIT_ON_CLOSE);
      generator = new Random();

      numberLabel              =              new              JLabel
(String.valueOf(generator.nextInt(MAX_NUMBER)+1));

      randomButton = new JButton("New Number");
      randomButton.addActionListener (new RandomListener());

      panel = new JPanel();
      panel.setPreferredSize (new Dimension(WIDTH, HEIGHT));
      panel.add (numberLabel);
      panel.add(randomButton);

      frame.getContentPane().add (panel);
   }

   //-----------------------------------------------------------------
   //  Displays the primary application frame.
   //-----------------------------------------------------------------
   public void display()
   {
      frame.pack();
      frame.setVisible(true);
   }
```

```
   //*****************************************************************
   //  Represents an action listener for the random button.
   //*****************************************************************
      private class RandomListener implements ActionListener
      {
         //--------------------------------------------------------------
         //  Generates and displays a new random number
         //--------------------------------------------------------------
         public void actionPerformed (ActionEvent event)
         {
            numberLabel.setText(Integer.toString
            (generator.nextInt(MAX_NUMBER)+1));
         }
      }
   }

   //*****************************************************************
   //  RandomDisplay.java       Author: Lewis/Loftus
   //*****************************************************************

   public class RandomDisplay
   {
      //--------------------------------------------------------------
      //  Creates and displays the random number GUI.
      //--------------------------------------------------------------
      public static void main (String[] args)
      {
         RandomGUI rand = new RandomGUI();
         rand.display();
      }
   }
```

2. Design and implement an application that presents two buttons and a label to the user. Label the buttons “Increment” and “Decrement”, respectively. Display a numeric value (initially 50) using the label. Each time the increment button is pushed, increment the value displayed. Likewise, each time the decrement button is pressed decrement the value displayed.

```
//*****************************************************************
//  IncDecGUI.java       Author: Lewis/Loftus
//*****************************************************************

import java.awt.Graphics;
import java.awt.Dimension;
import java.awt.event.*;
import javax.swing.*;

public class IncDecGUI
{
   private int WIDTH = 100;
   private int HEIGHT = 90;
   private int INITIAL_NUMBER = 50;
```

```
private JFrame frame;
private JPanel panel;
private JLabel numberLabel;
private JButton incButton, decButton;
private int number;

//-----------------------------------------------------------------
//  Sets up the GUI.
//-----------------------------------------------------------------
public IncDecGUI()
{
   frame = new JFrame ("Random Numbers");
   frame.setDefaultCloseOperation (JFrame.EXIT_ON_CLOSE);

   number = INITIAL_NUMBER;

   numberLabel = new JLabel (String.valueOf(number));

   incButton = new JButton("Increment");
   incButton.addActionListener (new IncListener());

   decButton = new JButton("Decrement");
   decButton.addActionListener (new DecListener());

   panel = new JPanel();
   panel.setPreferredSize (new Dimension(WIDTH, HEIGHT));
   panel.add (numberLabel);
   panel.add(incButton);
   panel.add(decButton);

   frame.getContentPane().add (panel);
}

//-----------------------------------------------------------------
//  Displays the primary application frame.
//-----------------------------------------------------------------
public void display()
{
   frame.pack();
   frame.setVisible(true);
}

//*****************************************************************
//  Represents an action listener for the increment button.
//*****************************************************************
private class IncListener implements ActionListener
{
   //--------------------------------------------------------------
   //  Increments and displays the current number
   //--------------------------------------------------------------
   public void actionPerformed (ActionEvent event)
   {
      numberLabel.setText (Integer.toString (++number));
   }
}
```

```
   //*****************************************************************
   //  Represents an action listener for the decrement button.
   //*****************************************************************
   private class DecListener implements ActionListener
   {
      //--------------------------------------------------------------
      //  Decrements and displays the current number
      //--------------------------------------------------------------
      public void actionPerformed (ActionEvent event)
      {
         numberLabel.setText (Integer.toString (--number));
      }
   }
}

//*****************************************************************
//  IncDec.java       Author: Lewis/Loftus
//*****************************************************************

public class IncDec
{
   //--------------------------------------------------------------
   //  Creates and displays the increment / decrement  GUI.
   //--------------------------------------------------------------

   public static void main (String[] args)
   {
      IncDecGUI id = new IncDecGUI();
      id.display();
   }
}
```

Problem 2: Choices (**20 minutes)** [30 points]

1. Write an application with three radio buttons labeled “Red”, “Green”, and “Blue” as well as a combo box containing three items labeled “Red”, “Green”, and “Blue”. The radio buttons and the combo box items should enable the user to change the background color of a panel displayed in the center of the frame to red, green, or blue.

```
import java.swing.*;
import java.awt.*;
import java.awt.event.*;

public class MyFrame extends JFrame {
   private String[] colors; private Color[] c;
   private JRadioButton[] radios;
   private JComboBox<String> box;
   private JPanel colorPanel;

   public MyFrame() {
         super(“Color Panel”);
         MyListener ml = new MyListener();
         colors = new String[] {“Red”, “Green”, “Blue”};
         box = new JComboBox<String>(colors);
         ButtonGroup group = new ButtonGroup();
         colorPanel = new JPanel();
         radios = new JRadioButton[colors.length];

         for(int i=0; i < colors.length; i++) {
         radios[i] = new JRadioButton(colors[i];
         group.add(radios[i]);
         radios[i].addActionListener(ml);
         }

         box.addActionListener(ml);
         c = new Color[] {Color.RED, Color.GREEN, Color.BLUE};

         radios[0].setSelected(true);
         box.setSelectedIndex(0);
         colorPanel.setBackground(c[0]);

         JPanel m = new JPanel();
         m.setLayout(new BorderLayout());
         m.add(colorPanel, BorderLayout.CENTER);
         JPanel sub = new JPanel();
         sub.setLayout(new GridLayout(2,3));
         for(JRadioButton rb : radios)
               sub.add(rb);
         sub.add(box);
         m.add(sub, BorderLayout.SOUTH);
         getContentPane().add(m);
   }
```

```
	private class MyListener implements ActionListener {
		public  void actionPerformed(ActionEvent e) {
			Object source = e.getSource();
			if(source instance of JComboBox) {
				int index = box.getSelectedIndex();
				radios[index].setSelected(true);
				colorPanel.setBackground(c[index]);
			}

			if(source instance of JRadioButton) {
				for(int i=0; i < radios.length; i++)
					if(radios[i].isSelected) {
					box.setSelectedIndex(i);
					colorPanel.setBackground(c[i]);
					break;
					}
			}
		}
	}
}
```

Problem 3: DOM Parser (**25 minutes)** [20 points]

Consider the following XML document that shows insurance policy details:

```
<policies>
      <policy>
            <policy-type>buildings</policy-type>
            <start-date>2007-12-12</start-date>
            <annual-premium>45.6</annual-premium>
            <number-of-claims>5</number-of-claims>
            <paid-up>true</paid-up>
      </policy>
      <policy>
            <policy-type>contents</policy-type>
            <start-date>2005-02-03</start-date>
            <annual-premium>84.0</annual-premium>
            <number-of-claims>0</number-of-claims>
            <paid-up>false</paid-up>
      </policy>
      <policy>
            <policy-type>buildings</policy-type>
            <start-date>1998-06-30</start-date>
            <annual-premium>59.0</annual-premium>
            <number-of-claims>1</number-of-claims>
            <paid-up>false</paid-up>
      </policy>
</policies>
```

The information given above is stored in an **XML file** called “policies.xml” that resides inside a **folder** named “XML” on the “http://www.wissamfawaz.com” **webserver**. So, “policies.xml” can be identified through the following URL: “http://www.wissamfawaz.com/XML/policies.xml”. In what follows, “policies.xml” is used as the source XML for parsing purposes.

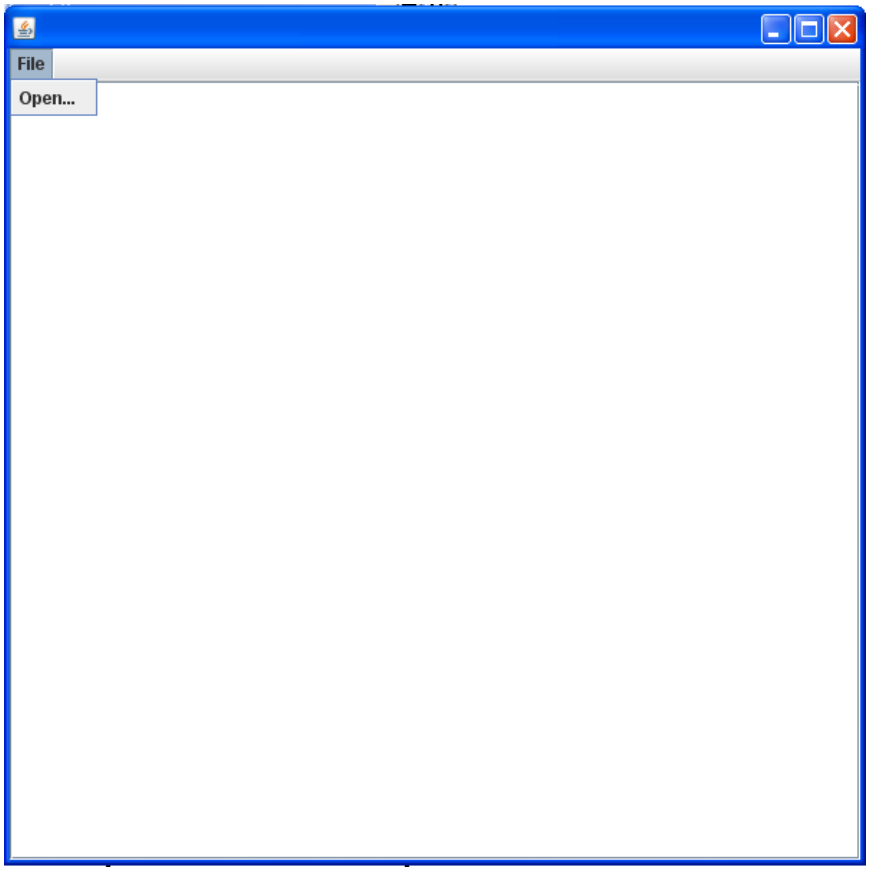
1. Draw a **tree** that captures the structure of the **Document Object Model** (DOM) corresponding to the considered XML document.

2. Using the **DOM parser**, write a Java program that:
    a. outputs the number of “policy” **elements** that have already been paid up (i.e., “paid-up” equals true) along with their associated “annual-premium” information, and
    b. then outputs the “policy-type” and “start-date” details pertaining to the policy having the smallest “number-of-claims” associated with it.

```
import java.io.InputStream;
import java.util.ArrayList;
import java.net.URL;
import javax.xml.parsers.DocumentBuilder;
import javax.xml.parsers.DocumentBuilderFactory;
import org.w3c.dom.Document;
import org.w3c.dom.Element;
import org.w3c.dom.Node;
import org.w3c.dom.NodeList;
public class ParsingPolicyInformation {
     public static void main(String[] args) {
     URL url = new URL(“http://.../policies.xml”);
     DocumentBuilderFactory dbf =
     DocumentBuilderFactory.newInstance();
     DocumentBuilder db = dbf.newDocumentBuilder();
     Document doc = db.parse(url.openStream());
     int paidPolicyCount = 0;
     StringBuilder annualPremium = new StringBuilder();
     int smallestClaims = -1;
     String policyType = null, String startDate = null;
     NodeList list = doc.getElementsByTagName(“policy”);
     for(int i=0; i<list.getLength(); i++) {
          Element e = (Element) list.item(i);
     NodeList nl = e.getElementsByTagName(“policy-type”);
          Node n = nl.item(0);
          String pt = n.getFirstChild().getNodeValue();
          l = e.getElementsByTagName(“start-date”);
          n = l.item(0);
          String sd = n.getFirstChild().getNodeValue();
          l = e.getElementByTagName(“annual-premium”);
          n = l.item(0);
          String ap = n.getFirstChild().getNodeValue();
          l = e.getElementsByTagName(“number-of-claims”);
          n = l.item(0);
          String noc = n.getFirstChild().getNodeValue();
          l = e.getElementsByTagName(“paid-up”);
          n = l.item(0);
          String pu = n.getFirstChild().getNodeValue();
```

```
          boolean paid = Boolean.parseBoolean(pu);
          if(paid) {
               paidPolicyCount++;
               annualPremium.append(ap + “\n”);
          }

          int claimCount = Integer.parseInt(noc);

     if(smallestClaims==-1 || smallestClaims > claimCount){

          smallestClaims = claimsCount;
          policyType = pt; startDate = sd;
     }

     System.out.println(“P1-“ + paidPolicyCount +
annaulPremium);
     System.out.println(“P2-“+smallestClaims + “-“ +
policyType + “-“+startDate);


     } // end main method
} // end ParsingPolicyInformation class
```

Problem 4: Menu, fileChooser, editorPane (**25 minutes)** [20 points]



Design and implement a Java application that uses the GUI shown above. This application enables the user to select both **text** and **image** files from the file system, retrieve the selected file and then display its content in an editor pane. As illustrated through the figure above, the GUI is made up of the following components: 1) **File menu** containing an **Open...** menu item that creates a **JFileChooser** when it gets selected, 2) **JEditorPane** used for the purpose of showing the content of the file obtained through the file chooser, and 3) **JScrollPane** providing scrolling for the editor pane.

It is important to note that the way the content of the file is displayed in the editor pane depends on whether the selected file is an image or a text file. As such, make sure that the type of the file is properly identified before its content is incorporated into the JEditorPane GUI component.

A program skeleton is provided next and your job is to simply complete the implementation of the program as per the guidelines conveyed through the comments that are highlighted in bold.

```
import java.awt.*;
import java.awt.event.*;
import java.io.File;
import java.net.URL;
import javax.swing.*;
public class OpenFilesFrame extends JFrame {
     // Include instance variables below
     private JEditorPane editor;
     private JScrollPane scroll;
     private JFileChooser fileChooser;
     private File selectedFile;



     // Complete the implementation of the constructor
     public OpenFilesFrame() {
          editor = new JEditorPane();
          scroll = new JScrollPane(editor);
          add(scroll, BorderLayout.CENTER);
          JMenuBar menu = new JMenuBar();
          JMenu f = new JMenu(“File”);
          JMenuItem open = new JMenuItem(“Open…”);
          open.addActionListener(new OpenListener());
          f.add(open); menu.add(f);
          setMenuBar(menu);
     }
     private class OpenListener implements ActionListener {
          @Override
          public void actionPerformed(ActionEvent e) {
               selectedFile = null;
               // create filechooser and show its dialog
               fileChooser = new JFileChooser();
     fileChooser.setSelectionMode(JFileChooser.FILES_ONLY);
     fileChooser.showInputDialog(OpenFilesFrame.this);

               // retrieve selected file
               selectedFile=fileChooser.getSelectedFile();
               if(selectedFile == null)
                    return;

               if(selectedFile != null)
                         showFileContent(selectedFile);
          }
     }
     // method used to display content of file in
     // editor pane.
     private void showFileContent(File file){
```

```
          name = file.getName();
          name = name.substring(name.lastIndexOf('.')+1);
          if(name.equalsIgnoreCase("txt")) {
               editor.setContentType("text/plain");
               editor.setPage(file.toURI().toURL());
          }
          else if(name.equalsIgnoreCase("jpg") ||
                  name.equalsIgnoreCase("png") ||
                  name.equalsIgnoreCase("gif")) {
               editor.setContentType("text/html");
               StringBuilder sb = new StringBuilder();
               sb.append("<HTML><BODY>");
               sb.append("<IMG           src=\""           +
file.toURI().toURL() + "\"/>");
               sb.append("</BODY></HTML>");
               editor.setText(sb.toString());
          }


     } // end showFileContent method
     // Complete the implementation of the main method
     public static void main(String[] args) {
          OpenFilesFrame frame = new OpenFilesFrame();
     frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
          frame.setSize(mew Dimension(400, 400));
          frame.setVisible(true);



     } // end main method
} // end OpenFilesFrame class
```

Appendix: Classes and Methods

1. DOM parser related classes along with their associated methods:

<table>
<tr><td>Document

Element getDocumentElement()
Element getElementById(String)
NodeList getElementsByTagName(String)</td><td>NodeList

int getLength()
Node item(int index)</td></tr>
<tr><td>Node

NodeList getChildNodes()
Node getFirstChild()
Node getLastChild()
String getNodeValue()
Node getParentNode()
boolean hasChildNodes()</td><td colspan="2">Element

NodeList getElementsByTagName(String)
String getTagName()
String getAttribute(String)
void removeAttribute(String)
boolean hasAttribute(String)</td></tr>
</table>

2. Classes related to GUI-based applications:

<table>
<tr><td>JFileChooser

static int APPROVE_OPTION
static int CANCEL_OPTION
int showInputDialog(Component)
File getSelectedFile()</td><td>JButton

JButton(String)
void setEnabled(boolean)
void setMnemonic(char)</td></tr>
<tr><td>JLabel

JLabel()
JLabel(String)
String getText()
void setText(String)</td><td>JEditorPane

JEditorPane()
void setPage(URL)
void setText(String)
void setContentType(String)</td></tr>
<tr><td>JRadioButton

JRadioButton(String)
boolean isSelected()
String getText()
void setSelected(boolean)</td><td>JComboBox

JComboBox(Object[])
void addItem(Object)
int getSelectedIndex()
void setSelectedIndex(int)</td></tr>
<tr><td>ActionEvent
Object getSource()
String getActionCommand()

ActionListener
void actionPerformed(ActionEvent)</td><td>File
String getName()
URI toURI()

URI
URL toURL()</td></tr>
</table>