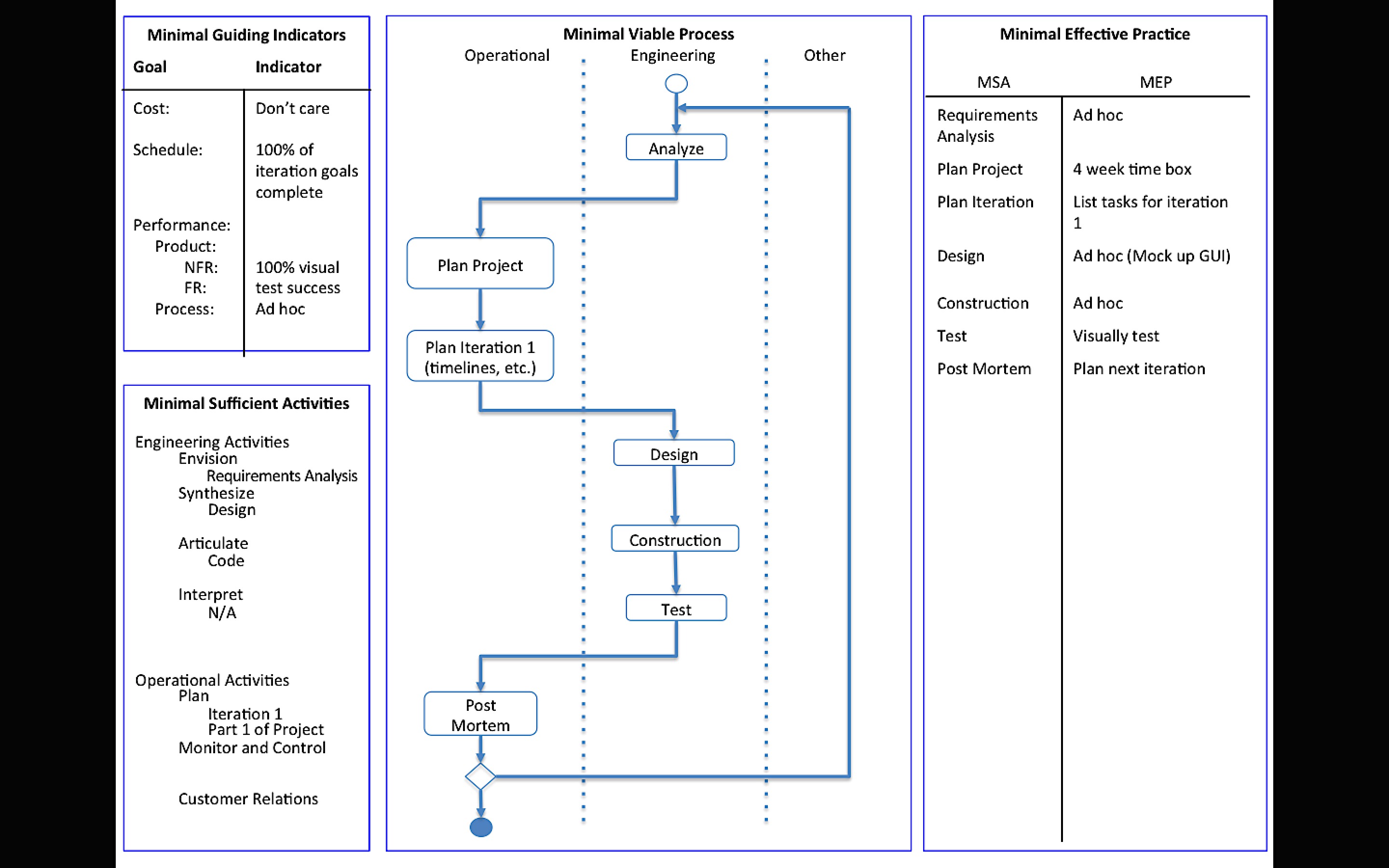
Iteration 1

# Process for Iteration 1



# Requirements Analysis

Given a secret image file and two innocent image files, the tool should

* Be able to read in image files and store the pixel information
* Use the extended visual cryptography scheme to encode the secret image pixels into the two innocent images
* Store the encoded images in new image files
  + The filenames and location can be specified by the user. If not, the files are named share1 and share2 and gets stored on the Desktop.

Given two encoded image files, the tool should

* Be able to read in the files and store the pixel information
* Use the extended visual cryptography scheme to decode the secret image from the encoded images (similar to super imposing them)
* The image revealing the secret gets stored in a new image file
  + The filename and location can be specified by the user. If not, the file is named secretMsg and gets stored on the Desktop.

The visual cryptography tool will only work with PNG and JPEG images. The images involved with the encoding process must have the same dimensions. The tool can handle images of any coloring.

# Plans for Project

Iteration 1:

* Create a graphical user interface
* Get the tool working for strictly black and white images
* Test the tool to check the quality of the encoded shares and the decoded message

Iteration 2:

* Research and implement the visual cryptography scheme with gray scale images
* Add features to project to help boost robustness (i.e. add in checks to keep the user from breaking the tool easily)

Iteration 3:

* Add the ability to encode and decode multicolor images

Iteration 4:

* Analyze the tool and look for ways to improve efficiency (performance and memory storage)

# Plans for Iteration 1

* Clean up the code developed over Summer 2015 and keep only the material relevant to this project.
* Add a graphic user interface to increase the quality of the project.
* Analyze the results of current black and white extended visual cryptography scheme to look for ways of improvement or prepare for it to evolve into handling gray scale images.

# Design

The design portion of Iteration 1 was focused on designing a graphical user interface for the visual cryptography tool. While thinking about how I wanted the user to work with the tool, I decided that three designs were needed: main/welcome page, encode page, and decode page. Then, I created mockups on engineering paper to plan the elements to be placed in the java frames. Figure 1 shows the mockups and Figure 2 displays the current user interface for the visual cryptography tool.

|  |
| --- |
|  |
| Figure 1: Design of Main/Welcome Page (left), Design of Encode Page (middle), Design of Decode Page (right) |

|  |
| --- |
|  |
| Figure 2: Main/Welcome GUI (top), Encode GUI (center), Decode GUI (bottom) |

# Construction

The construction for Iteration 1 involved creating a graphical user interface and merging my previous developed code with the new interface. In order to better design the GUI, I used the NetBeans IDE 8.0.2. This IDE allows you to drag and drop items from the javax.swing Library into your frame and more easily align objects. Once all those elements were functioning, I added in the classes necessary for the encoding and decoding processes. See source code at the end of documentation.

# Test

Due to time constraints, I was only able to perform one encoding test and one decoding test. Figure 3 shows the PNG image files I used for encoding.

|  |
| --- |
|  |
| Figure 3: Secret Image to be Encoded (left), Innocent Image 1 (middle), Innocent Image 2 (right) |

The input for encoding can be seen in Figure 4, and it shows how the graphical user interface works with the encoding process.

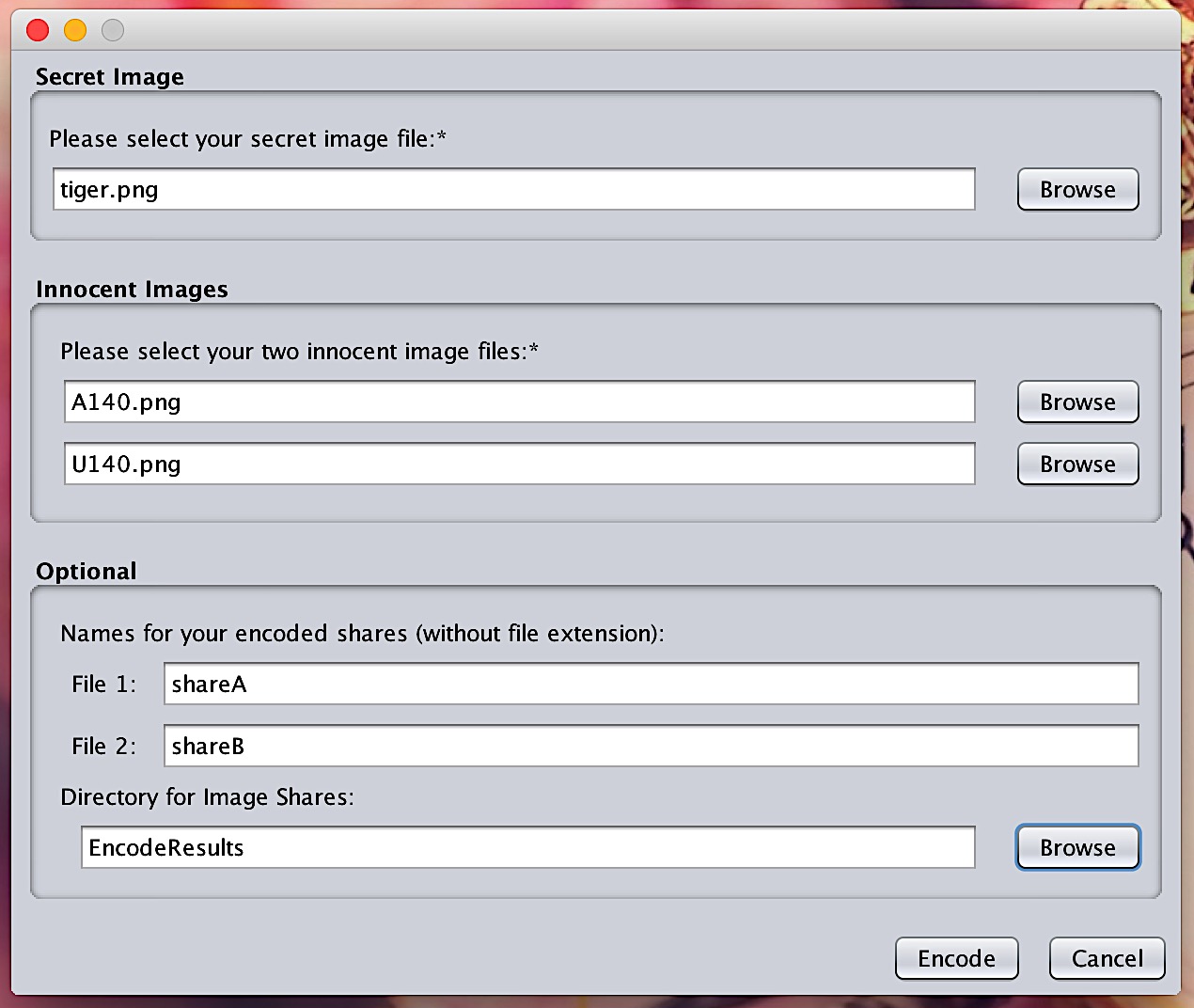


Figure 4: GUI before encoding the images in Figure 3.

Figure 5 displays the results of the encoding process. These images were used as the inputs for testing the decoding process as well.

|  |
| --- |
|  |
| Figure 5: Encoded Share A (left), Encoded Share B (right) |

The input for encoding can be seen in Figure 6, and it shows how the graphical user interface works with the decoding process.

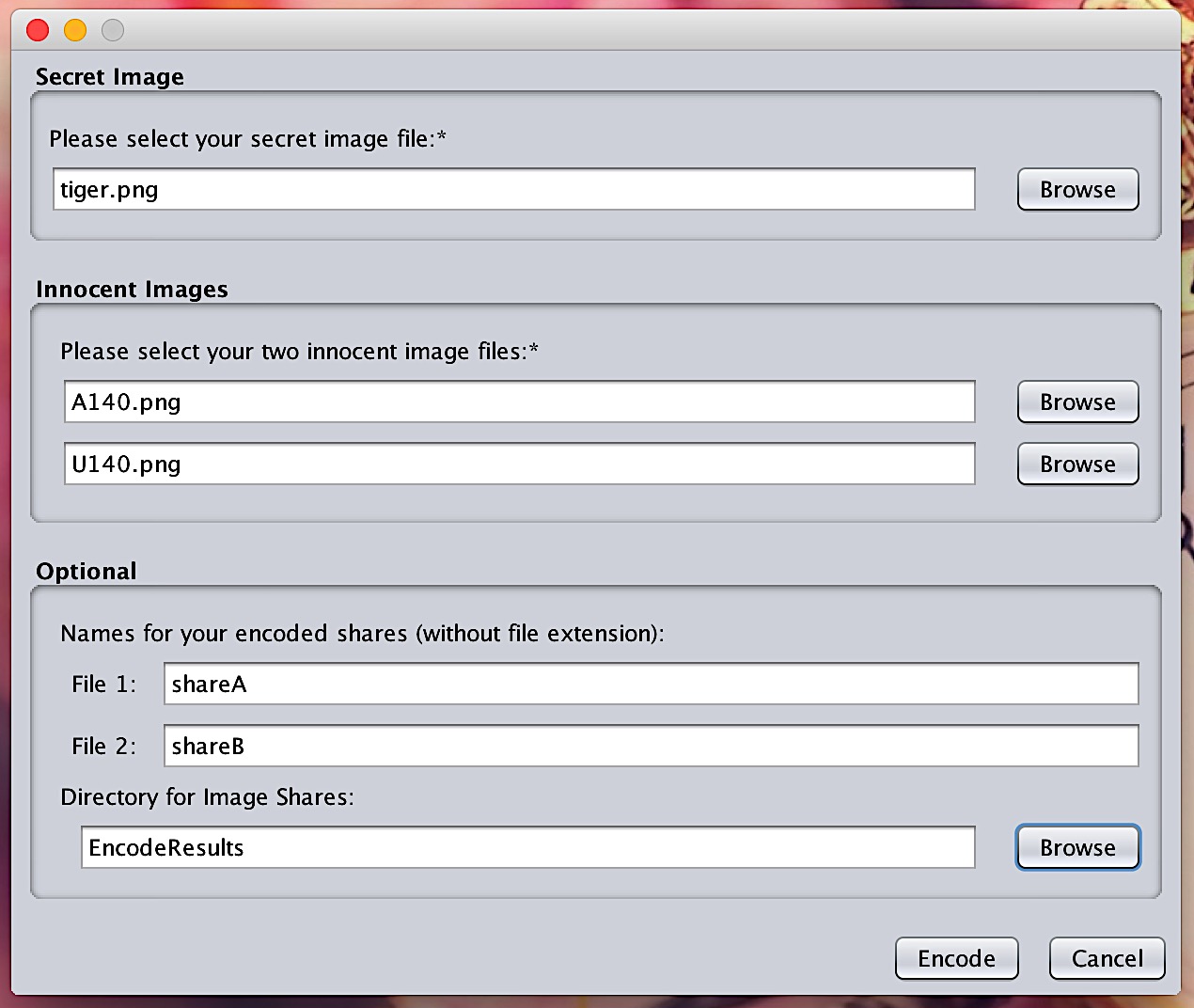


Figure 6: GUI before decoding the images in Figure 5.

The result of the decoding Share A and Share B can be viewed in Figure 7.

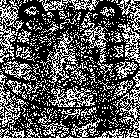


Figure 7: Result of decoding shares A and B from Figure 5.

# Post Mortem

During Iteration 1, I created a graphical user interface to allow for better interaction with the visual cryptography tool. The user interface lets you visual find the image files you wish to use for encoding or decoding without worrying about typing out the absolute path to the file. Once the basics for the GUI were completed, I added some of the classes I made over summer into the project. Currently, the project can encode and decode images that contain only black or white pixels. Thus, the first two tasks listed under the Iteration 1 plan were met.

I did not have a significant amount of time for testing during this iteration. Hence, I only performed one encoding test and one decoding test. The results can be seen in the Testing section.

There is an issue with the matrices I am using to encode the black and white images. Share A looks fine, and you can tell the image is the letter A. Share B, on the other hand, is not as clear as it should be. A person should be able to look at Share B and only see the letter U. With this file, you can make out the letter A and see a portion of the tiger’s face. This issue will be investigated during iteration 2 along with adding functionality for the project to handle gray scale images.

As for the decoding test, you can see a tiger face in the decoded image (Figure 7). However, the decoding process adds more noise to the image making it difficult to see all the aspects of the secret message. The algorithm I am using should permit a user to print decoded images onto transparencies and physically stack the images on top of one another to reveal the secret image. More research needs to be done to see if the background noise can be reduced or if the requirements of no technology needed for the decoding needs to be altered.

During Iteration 2, I plan on adding the ability to encode and decode gray scale images and make the GUI more robust. Based on the little amount of time I had to test, the task of applying more robust error checking may be delayed until Iteration 4 since it is not a task necessary for the encryption or decryption of images. The delay will only occur if the gray scale implementation and testing takes longer than expected.

# Source Code

MainFrame.java

1 package Masters\_Proj;  
 2   
 3 /\*  
 4 \* To change this license header, choose License Headers in Project Properties.  
 5 \* To change this template file, choose Tools | Templates  
 6 \* and open the template in the editor.  
 7 \*/  
 8   
 9 /\*\*  
 10 \*  
 11 \* @author allisonholt  
 12 \*/  
 13 public class MainFrame extends javax.swing.JFrame {  
 14   
 15 /\*\*  
 16 \* Creates new form StartFrame  
 17 \*/  
 18 public MainFrame() {  
 19 initComponents();  
 20 }  
 21   
 22 /\*\*  
 23 \* This method is called from within the constructor to initialize the form.  
 24 \* WARNING: Do NOT modify this code. The content of this method is always  
 25 \* regenerated by the Form Editor.  
 26 \*/  
 27 @SuppressWarnings("unchecked")  
 28 // <editor-fold defaultstate="collapsed" desc="Generated Code">//GEN-BEGIN:initComponents  
 29 private void initComponents() {  
 30   
 31 welcomeBanner = new javax.swing.JLabel();  
 32 jScrollPane1 = new javax.swing.JScrollPane();  
 33 descriptionArea = new javax.swing.JTextArea();  
 34 jScrollPane2 = new javax.swing.JScrollPane();  
 35 directionsArea = new javax.swing.JTextArea();  
 36 encodeButton = new javax.swing.JButton();  
 37 decodeButton = new javax.swing.JButton();  
 38   
 39 setDefaultCloseOperation(javax.swing.WindowConstants.EXIT\_ON\_CLOSE);  
 40 setTitle("Holt Visual Cryptography");  
 41   
 42 welcomeBanner.setHorizontalAlignment(javax.swing.SwingConstants.CENTER);  
 43 welcomeBanner.setText("Welcome to the Holt Visual Cryptography Tool!");  
 44   
 45 descriptionArea.setEditable(false);  
 46 descriptionArea.setColumns(20);  
 47 descriptionArea.setLineWrap(true);  
 48 descriptionArea.setRows(5);  
 49 descriptionArea.setText("The Holt Cryptography Tool allows you to encrypt or decrypt a secret image using extended visual cryptography. The secret image gets embedded into two innocent images that must be superimposed in order to reveal the secret information.");  
 50 descriptionArea.setWrapStyleWord(true);  
 51 jScrollPane1.setViewportView(descriptionArea);  
 52   
 53 directionsArea.setColumns(20);  
 54 directionsArea.setLineWrap(true);  
 55 directionsArea.setRows(5);  
 56 directionsArea.setText("If you wish to encrypt a secret image, then select the encode button. If you wish to decrypt a secret message, then select the decode button.");  
 57 directionsArea.setWrapStyleWord(true);  
 58 jScrollPane2.setViewportView(directionsArea);  
 59   
 60 encodeButton.setText("Encode");  
 61 encodeButton.addActionListener(  
 62 new java.awt.event.ActionListener() {  
 63 public void actionPerformed(java.awt.event.ActionEvent evt) {  
 64 encodePressed(evt);  
 65 }  
 66 });  
 67   
 68 decodeButton.setText("Decode");  
 69 decodeButton.setHorizontalAlignment(javax.swing.SwingConstants.RIGHT);  
 70 decodeButton.addActionListener(  
 71 new java.awt.event.ActionListener() {  
 72 public void actionPerformed(java.awt.event.ActionEvent evt) {  
 73 decodePressed(evt);  
 74 }  
 75 });  
 76   
 77 javax.swing.GroupLayout layout = new javax.swing.GroupLayout(getContentPane());  
 78 getContentPane().setLayout(layout);  
 79 layout.setHorizontalGroup(  
 80 layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)  
 81 .addComponent(welcomeBanner, javax.swing.GroupLayout.DEFAULT\_SIZE, 600, Short.MAX\_VALUE)  
 82 .addGroup(javax.swing.GroupLayout.Alignment.TRAILING, layout.createSequentialGroup()  
 83 .addContainerGap()  
 84 .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRAILING)  
 85 .addComponent(jScrollPane2)  
 86 .addComponent(jScrollPane1))  
 87 .addContainerGap())  
 88 .addGroup(layout.createSequentialGroup()  
 89 .addGap(66, 66, 66)  
 90 .addComponent(encodeButton)  
 91 .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED, javax.swing.GroupLayout.DEFAULT\_SIZE, Short.MAX\_VALUE)  
 92 .addComponent(decodeButton)  
 93 .addGap(66, 66, 66))  
 94 );  
 95 layout.setVerticalGroup(  
 96 layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)  
 97 .addGroup(layout.createSequentialGroup()  
 98 .addGap(24, 24, 24)  
 99 .addComponent(welcomeBanner, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, Short.MAX\_VALUE)  
100 .addGap(18, 18, 18)  
101 .addComponent(jScrollPane1, javax.swing.GroupLayout.PREFERRED\_SIZE, 64, javax.swing.GroupLayout.PREFERRED\_SIZE)  
102 .addGap(18, 18, 18)  
103 .addComponent(jScrollPane2, javax.swing.GroupLayout.PREFERRED\_SIZE, 47, javax.swing.GroupLayout.PREFERRED\_SIZE)  
104 .addGap(18, 18, 18)  
105 .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)  
106 .addComponent(encodeButton)  
107 .addComponent(decodeButton))  
108 .addContainerGap(javax.swing.GroupLayout.DEFAULT\_SIZE, Short.MAX\_VALUE))  
109 );  
110   
111 pack();  
112 }// </editor-fold>//GEN-END:initComponents  
113   
114 private void encodePressed(java.awt.event.ActionEvent evt) {//GEN-FIRST:event\_encodePressed  
115 // TODO add your handling code here:  
116 new EncodeFrame().setVisible(true);  
117 this.setVisible(false);  
118   
119 }//GEN-LAST:event\_encodePressed  
120   
121 private void decodePressed(java.awt.event.ActionEvent evt) {//GEN-FIRST:event\_decodePressed  
122 // TODO add your handling code here:  
123 new DecodeFrame().setVisible(true);  
124 this.setVisible(false);  
125 }//GEN-LAST:event\_decodePressed  
126   
127 /\*\*  
128 \* @param args the command line arguments  
129 \*/  
130 public static void main(String args[]) {  
131 /\* Set the Nimbus look and feel \*/  
132 //<editor-fold defaultstate="collapsed" desc=" Look and feel setting code (optional) ">  
133 /\* If Nimbus (introduced in Java SE 6) is not available, stay with the default look and feel.  
134 \* For details see http://download.oracle.com/javase/tutorial/uiswing/lookandfeel/plaf.html   
135 \*/  
136 try {  
137 for (javax.swing.UIManager.LookAndFeelInfo info : javax.swing.UIManager.getInstalledLookAndFeels()) {  
138 if ("Nimbus".equals(info.getName())) {  
139 javax.swing.UIManager.setLookAndFeel(info.getClassName());  
140 break;  
141 }  
142 }  
143 }   
144 catch (ClassNotFoundException ex) {  
145 java.util.logging.Logger.getLogger(MainFrame.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);  
146 }   
147 catch (InstantiationException ex) {  
148 java.util.logging.Logger.getLogger(MainFrame.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);  
149 }   
150 catch (IllegalAccessException ex) {  
151 java.util.logging.Logger.getLogger(MainFrame.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);  
152 }   
153 catch (javax.swing.UnsupportedLookAndFeelException ex) {  
154 java.util.logging.Logger.getLogger(MainFrame.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);  
155 }  
156 //</editor-fold>  
157 //</editor-fold>  
158   
159 /\* Create and display the form \*/  
160 java.awt.EventQueue.invokeLater(  
161 new Runnable() {  
162 public void run() {  
163 new MainFrame().setVisible(true);  
164 }  
165 });  
166 }  
167   
168 // Variables declaration - do not modify//GEN-BEGIN:variables  
169 private javax.swing.JButton decodeButton;  
170 private javax.swing.JTextArea descriptionArea;  
171 private javax.swing.JTextArea directionsArea;  
172 private javax.swing.JButton encodeButton;  
173 private javax.swing.JScrollPane jScrollPane1;  
174 private javax.swing.JScrollPane jScrollPane2;  
175 private javax.swing.JLabel welcomeBanner;  
176 // End of variables declaration//GEN-END:variables  
177 }  
178

EncodeFrame.java

1 package Masters\_Proj;  
 2   
 3 import java.awt.image.BufferedImage;  
 4 import java.io.File;  
 5 import java.io.IOException;  
 6 import javax.imageio.ImageIO;  
 7 import javax.swing.JFileChooser;  
 8   
 9 /\*  
 10 \* To change this license header, choose License Headers in Project Properties.  
 11 \* To change this template file, choose Tools | Templates  
 12 \* and open the template in the editor.  
 13 \*/  
 14   
 15 /\*\*  
 16 \*  
 17 \* @author allisonholt  
 18 \*/  
 19 public class EncodeFrame extends javax.swing.JFrame {  
 20   
 21 /\*\*  
 22 \* Creates new form EncodeFrame  
 23 \*/  
 24 public EncodeFrame() {  
 25 initComponents();  
 26 }  
 27   
 28 public EncodeFrame(EncodeFrame prevState)  
 29 {  
 30 this.secretTextField.setText(prevState.secretTextField.getText());  
 31 }  
 32   
 33 /\*\*  
 34 \* This method is called from within the constructor to initialize the form.  
 35 \* WARNING: Do NOT modify this code. The content of this method is always  
 36 \* regenerated by the Form Editor.  
 37 \*/  
 38 @SuppressWarnings("unchecked")  
 39 // <editor-fold defaultstate="collapsed" desc="Generated Code">//GEN-BEGIN:initComponents  
 40 private void initComponents() {  
 41   
 42 imageChooser = new javax.swing.JFileChooser();  
 43 directoryChooser = new javax.swing.JFileChooser();  
 44 cancelButton = new javax.swing.JButton();  
 45 encodeButton = new javax.swing.JButton();  
 46 panel1 = new javax.swing.JPanel();  
 47 secretTextField = new javax.swing.JTextField();  
 48 jLabel1 = new javax.swing.JLabel();  
 49 browseButton1 = new javax.swing.JButton();  
 50 jPanel1 = new javax.swing.JPanel();  
 51 jLabel2 = new javax.swing.JLabel();  
 52 innocentTextField1 = new javax.swing.JTextField();  
 53 browseButton2 = new javax.swing.JButton();  
 54 innocentTextField2 = new javax.swing.JTextField();  
 55 browseButton3 = new javax.swing.JButton();  
 56 optionalPanel = new javax.swing.JPanel();  
 57 jLabel3 = new javax.swing.JLabel();  
 58 jLabel4 = new javax.swing.JLabel();  
 59 jLabel5 = new javax.swing.JLabel();  
 60 filename1 = new javax.swing.JTextField();  
 61 filename2 = new javax.swing.JTextField();  
 62 jLabel6 = new javax.swing.JLabel();  
 63 storageDirectoryTextField = new javax.swing.JTextField();  
 64 browseButton4 = new javax.swing.JButton();  
 65   
 66 imageChooser.setDialogTitle("Choose an Image");  
 67 imageChooser.setFileFilter(new ImageCustomFilter());  
 68   
 69 directoryChooser.setDialogTitle("Choose a Directory");  
 70 directoryChooser.setFileFilter(new DirectoryCustomFilter());  
 71 directoryChooser.setFileSelectionMode(javax.swing.JFileChooser.DIRECTORIES\_ONLY);  
 72   
 73 setDefaultCloseOperation(javax.swing.WindowConstants.EXIT\_ON\_CLOSE);  
 74 setResizable(false);  
 75   
 76 cancelButton.setText("Cancel");  
 77 cancelButton.addActionListener(  
 78 new java.awt.event.ActionListener() {  
 79 public void actionPerformed(java.awt.event.ActionEvent evt) {  
 80 cancelPressed(evt);  
 81 }  
 82 });  
 83   
 84 encodeButton.setText("Encode");  
 85 encodeButton.addActionListener(  
 86 new java.awt.event.ActionListener() {  
 87 public void actionPerformed(java.awt.event.ActionEvent evt) {  
 88 encodePressed(evt);  
 89 }  
 90 });  
 91   
 92 panel1.setBorder(javax.swing.BorderFactory.createTitledBorder("Secret Image"));  
 93 panel1.setToolTipText("Secret Image");  
 94   
 95 jLabel1.setText("Please select your secret image file:\*");  
 96   
 97 browseButton1.setText("Browse");  
 98 browseButton1.addActionListener(  
 99 new java.awt.event.ActionListener() {  
100 public void actionPerformed(java.awt.event.ActionEvent evt) {  
101 imageBrowsePressed(evt);  
102 }  
103 });  
104   
105 javax.swing.GroupLayout panel1Layout = new javax.swing.GroupLayout(panel1);  
106 panel1.setLayout(panel1Layout);  
107 panel1Layout.setHorizontalGroup(  
108 panel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)  
109 .addGroup(panel1Layout.createSequentialGroup()  
110 .addComponent(jLabel1)  
111 .addGap(0, 0, Short.MAX\_VALUE))  
112 .addGroup(panel1Layout.createSequentialGroup()  
113 .addComponent(secretTextField)  
114 .addGap(18, 18, 18)  
115 .addComponent(browseButton1))  
116 );  
117 panel1Layout.setVerticalGroup(  
118 panel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)  
119 .addGroup(panel1Layout.createSequentialGroup()  
120 .addContainerGap()  
121 .addComponent(jLabel1)  
122 .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)  
123 .addGroup(panel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)  
124 .addComponent(secretTextField, javax.swing.GroupLayout.PREFERRED\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.PREFERRED\_SIZE)  
125 .addComponent(browseButton1))  
126 .addContainerGap(javax.swing.GroupLayout.DEFAULT\_SIZE, Short.MAX\_VALUE))  
127 );  
128   
129 jPanel1.setBorder(javax.swing.BorderFactory.createTitledBorder("Innocent Images"));  
130   
131 jLabel2.setText("Please select your two innocent image files:\*");  
132   
133 browseButton2.setText("Browse");  
134 browseButton2.addActionListener(  
135 new java.awt.event.ActionListener() {  
136 public void actionPerformed(java.awt.event.ActionEvent evt) {  
137 imageBrowsePressed(evt);  
138 }  
139 });  
140   
141 browseButton3.setText("Browse");  
142 browseButton3.addActionListener(  
143 new java.awt.event.ActionListener() {  
144 public void actionPerformed(java.awt.event.ActionEvent evt) {  
145 imageBrowsePressed(evt);  
146 }  
147 });  
148   
149 javax.swing.GroupLayout jPanel1Layout = new javax.swing.GroupLayout(jPanel1);  
150 jPanel1.setLayout(jPanel1Layout);  
151 jPanel1Layout.setHorizontalGroup(  
152 jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)  
153 .addGroup(jPanel1Layout.createSequentialGroup()  
154 .addContainerGap()  
155 .addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)  
156 .addGroup(jPanel1Layout.createSequentialGroup()  
157 .addComponent(jLabel2)  
158 .addContainerGap(javax.swing.GroupLayout.DEFAULT\_SIZE, Short.MAX\_VALUE))  
159 .addGroup(javax.swing.GroupLayout.Alignment.TRAILING, jPanel1Layout.createSequentialGroup()  
160 .addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRAILING)  
161 .addComponent(innocentTextField2, javax.swing.GroupLayout.Alignment.LEADING)  
162 .addComponent(innocentTextField1))  
163 .addGap(18, 18, 18)  
164 .addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)  
165 .addComponent(browseButton2)  
166 .addComponent(browseButton3)))))  
167 );  
168 jPanel1Layout.setVerticalGroup(  
169 jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)  
170 .addGroup(jPanel1Layout.createSequentialGroup()  
171 .addContainerGap()  
172 .addComponent(jLabel2)  
173 .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)  
174 .addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)  
175 .addComponent(innocentTextField1, javax.swing.GroupLayout.PREFERRED\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.PREFERRED\_SIZE)  
176 .addComponent(browseButton2))  
177 .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)  
178 .addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)  
179 .addComponent(innocentTextField2, javax.swing.GroupLayout.PREFERRED\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.PREFERRED\_SIZE)  
180 .addComponent(browseButton3))  
181 .addContainerGap(10, Short.MAX\_VALUE))  
182 );  
183   
184 optionalPanel.setBorder(javax.swing.BorderFactory.createTitledBorder("Optional"));  
185   
186 jLabel3.setText("Names for your encoded shares (without file extension):");  
187   
188 jLabel4.setText("File 1:");  
189   
190 jLabel5.setText("File 2:");  
191   
192 jLabel6.setText("Directory for Image Shares:");  
193   
194 browseButton4.setText("Browse");  
195 browseButton4.addActionListener(  
196 new java.awt.event.ActionListener() {  
197 public void actionPerformed(java.awt.event.ActionEvent evt) {  
198 dirBrowsePressed(evt);  
199 }  
200 });  
201   
202 javax.swing.GroupLayout optionalPanelLayout = new javax.swing.GroupLayout(optionalPanel);  
203 optionalPanel.setLayout(optionalPanelLayout);  
204 optionalPanelLayout.setHorizontalGroup(  
205 optionalPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)  
206 .addGroup(optionalPanelLayout.createSequentialGroup()  
207 .addContainerGap()  
208 .addGroup(optionalPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)  
209 .addGroup(optionalPanelLayout.createSequentialGroup()  
210 .addGroup(optionalPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)  
211 .addComponent(jLabel3)  
212 .addComponent(jLabel6))  
213 .addContainerGap())  
214 .addGroup(optionalPanelLayout.createSequentialGroup()  
215 .addGap(6, 6, 6)  
216 .addGroup(optionalPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)  
217 .addGroup(optionalPanelLayout.createSequentialGroup()  
218 .addComponent(jLabel5)  
219 .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)  
220 .addComponent(filename2))  
221 .addGroup(optionalPanelLayout.createSequentialGroup()  
222 .addComponent(jLabel4)  
223 .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)  
224 .addComponent(filename1))  
225 .addGroup(optionalPanelLayout.createSequentialGroup()  
226 .addGap(0, 3, Short.MAX\_VALUE)  
227 .addComponent(storageDirectoryTextField, javax.swing.GroupLayout.PREFERRED\_SIZE, 480, javax.swing.GroupLayout.PREFERRED\_SIZE)  
228 .addGap(18, 18, 18)  
229 .addComponent(browseButton4))))))  
230 );  
231 optionalPanelLayout.setVerticalGroup(  
232 optionalPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)  
233 .addGroup(optionalPanelLayout.createSequentialGroup()  
234 .addContainerGap()  
235 .addComponent(jLabel3)  
236 .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)  
237 .addGroup(optionalPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)  
238 .addComponent(jLabel4)  
239 .addComponent(filename1, javax.swing.GroupLayout.PREFERRED\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.PREFERRED\_SIZE))  
240 .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)  
241 .addGroup(optionalPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)  
242 .addComponent(jLabel5)  
243 .addComponent(filename2, javax.swing.GroupLayout.PREFERRED\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.PREFERRED\_SIZE))  
244 .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)  
245 .addComponent(jLabel6)  
246 .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)  
247 .addGroup(optionalPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)  
248 .addComponent(storageDirectoryTextField, javax.swing.GroupLayout.PREFERRED\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.PREFERRED\_SIZE)  
249 .addComponent(browseButton4))  
250 .addGap(0, 6, Short.MAX\_VALUE))  
251 );  
252   
253 javax.swing.GroupLayout layout = new javax.swing.GroupLayout(getContentPane());  
254 getContentPane().setLayout(layout);  
255 layout.setHorizontalGroup(  
256 layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)  
257 .addGroup(layout.createSequentialGroup()  
258 .addContainerGap()  
259 .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)  
260 .addGroup(javax.swing.GroupLayout.Alignment.TRAILING, layout.createSequentialGroup()  
261 .addGap(0, 0, Short.MAX\_VALUE)  
262 .addComponent(encodeButton)  
263 .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)  
264 .addComponent(cancelButton))  
265 .addComponent(panel1, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, Short.MAX\_VALUE)  
266 .addComponent(jPanel1, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, Short.MAX\_VALUE)  
267 .addComponent(optionalPanel, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, Short.MAX\_VALUE))  
268 .addContainerGap())  
269 );  
270 layout.setVerticalGroup(  
271 layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)  
272 .addGroup(javax.swing.GroupLayout.Alignment.TRAILING, layout.createSequentialGroup()  
273 .addContainerGap()  
274 .addComponent(panel1, javax.swing.GroupLayout.PREFERRED\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.PREFERRED\_SIZE)  
275 .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)  
276 .addComponent(jPanel1, javax.swing.GroupLayout.PREFERRED\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.PREFERRED\_SIZE)  
277 .addGap(12, 12, 12)  
278 .addComponent(optionalPanel, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, Short.MAX\_VALUE)  
279 .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)  
280 .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)  
281 .addComponent(cancelButton)  
282 .addComponent(encodeButton))  
283 .addContainerGap())  
284 );  
285   
286 pack();  
287 }// </editor-fold>//GEN-END:initComponents  
288   
289 private void cancelPressed(java.awt.event.ActionEvent evt) {//GEN-FIRST:event\_cancelPressed  
290 // TODO add your handling code here:  
291 this.setVisible(false);  
292 new MainFrame().setVisible(true);  
293 }//GEN-LAST:event\_cancelPressed  
294   
295 private void dirBrowsePressed(java.awt.event.ActionEvent evt) {//GEN-FIRST:event\_dirBrowsePressed  
296 // TODO add your handling code here:  
297 //Add code to handle file broswer  
298 //new ImageFileChooser().setVisible(true);  
299   
300 int returnVal = directoryChooser.showOpenDialog(this);  
301 if(returnVal == JFileChooser.APPROVE\_OPTION)  
302 {  
303 File dir = directoryChooser.getSelectedFile();  
304 if(evt.getSource() == browseButton4)  
305 {  
306 storageDirectoryTextField.setText(dir.getName());  
307 directoryForStorage = dir.getAbsolutePath();  
308 }  
309 }  
310   
311 }//GEN-LAST:event\_dirBrowsePressed  
312   
313 private void imageBrowsePressed(java.awt.event.ActionEvent evt) {//GEN-FIRST:event\_imageBrowsePressed  
314 // TODO add your handling code here:  
315   
316 int returnVal = imageChooser.showOpenDialog(this);  
317 if(returnVal == JFileChooser.APPROVE\_OPTION)  
318 {  
319 File imageFile = imageChooser.getSelectedFile();  
320 if(evt.getSource() == browseButton1)  
321 {  
322 secretTextField.setText(imageFile.getName());  
323 secretFile = imageFile.getAbsolutePath();  
324 }  
325 else if(evt.getSource() == browseButton2)  
326 {  
327 innocentTextField1.setText(imageFile.getName());  
328 innocentFiles[0] = imageFile.getAbsolutePath();  
329 }  
330 else if(evt.getSource() == browseButton3)  
331 {  
332 innocentTextField2.setText(imageFile.getName());  
333 innocentFiles[1] = imageFile.getAbsolutePath();  
334 }  
335 }  
336 }//GEN-LAST:event\_imageBrowsePressed  
337   
338 private void encodePressed(java.awt.event.ActionEvent evt) {//GEN-FIRST:event\_encodePressed  
339 //Code to encode secret message  
340 BufferedImage secretImage = null;  
341 boolean fileFound;  
342 try  
343 {  
344 secretImage = ImageIO.read(new File(secretFile));  
345 fileFound = true;  
346 }  
347 catch (IOException e)  
348 {  
349 //Set up and alert window  
350 fileFound = false;  
351 }  
352   
353 BufferedImage[] innocentShares = new BufferedImage[0];  
354 if(fileFound)  
355 {  
356 innocentShares = new BufferedImage[2];  
357 for(int i = 0; i < 2; i++)  
358 {  
359 try  
360 {  
361 innocentShares[i] = ImageIO.read(new File(innocentFiles[i]));  
362 fileFound = true;  
363 }  
364 catch(IOException e)  
365 {  
366 //Set up alert window  
367 fileFound = false;  
368 }  
369 }  
370 }  
371   
372 if(fileFound)  
373 {  
374 ExtendedVCS myEVCS = new ExtendedVCS(secretImage, innocentShares);  
375 myEVCS.encryptImage();  
376   
377 int[][] newInnocentRGB = myEVCS.getRGBPixelsForShares();  
378   
379 //boolean makeDir = true;  
380 if(storageDirectoryTextField.getText().equals(""))  
381 {  
382 //Get path to users desktop  
383 //BUG!!! Not working.  
384 directoryForStorage = "C:/Users/allisonholt/Desktop";  
385 //makeDir = false;  
386 }  
387   
388 //if(makeDir)  
389 //{  
390 //File directory = new File(directoryForStorage);  
391 //}  
392   
393 String[] shareFiles = new String[2];  
394   
395 if(filename1.getText().equals(""))  
396 {  
397 shareFiles[0] = directoryForStorage + "/share1.png";  
398 }  
399 else  
400 {  
401 shareFiles[0] = directoryForStorage + "/" + filename1.getText() +".png";  
402 }  
403   
404 if(filename2.getText().equals(""))  
405 {  
406 shareFiles[1] = directoryForStorage + "/share2.png";  
407 }  
408 else  
409 {  
410 shareFiles[1] = directoryForStorage + "/" + filename2.getText() +".png";  
411 }  
412   
413 for(int i = 0; i < 2; i++)  
414 {  
415 try  
416 {  
417 BufferedImage tempShare = new BufferedImage(myEVCS.getImgWidth(), myEVCS.getImgHeight(), BufferedImage.TYPE\_INT\_ARGB);  
418 tempShare.setRGB(0, 0, myEVCS.getImgWidth(), myEVCS.getImgHeight(), newInnocentRGB[i], 0, myEVCS.getImgWidth());  
419 File tempOutput = new File(shareFiles[i]);  
420 ImageIO.write(tempShare, "png", tempOutput);  
421 }  
422 catch (IOException e)  
423 {  
424 //Print alert here  
425 }  
426 }  
427   
428 }  
429 }//GEN-LAST:event\_encodePressed  
430   
431 /\*\*  
432 \* @param args the command line arguments  
433 \*/  
434 public static void main(String args[]) {  
435 /\* Set the Nimbus look and feel \*/  
436 //<editor-fold defaultstate="collapsed" desc=" Look and feel setting code (optional) ">  
437 /\* If Nimbus (introduced in Java SE 6) is not available, stay with the default look and feel.  
438 \* For details see http://download.oracle.com/javase/tutorial/uiswing/lookandfeel/plaf.html   
439 \*/  
440 try {  
441 for (javax.swing.UIManager.LookAndFeelInfo info : javax.swing.UIManager.getInstalledLookAndFeels()) {  
442 if ("Nimbus".equals(info.getName())) {  
443 javax.swing.UIManager.setLookAndFeel(info.getClassName());  
444 break;  
445 }  
446 }  
447 }   
448 catch (ClassNotFoundException ex) {  
449 java.util.logging.Logger.getLogger(EncodeFrame.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);  
450 }   
451 catch (InstantiationException ex) {  
452 java.util.logging.Logger.getLogger(EncodeFrame.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);  
453 }   
454 catch (IllegalAccessException ex) {  
455 java.util.logging.Logger.getLogger(EncodeFrame.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);  
456 }   
457 catch (javax.swing.UnsupportedLookAndFeelException ex) {  
458 java.util.logging.Logger.getLogger(EncodeFrame.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);  
459 }  
460 //</editor-fold>  
461   
462 /\* Create and display the form \*/  
463 java.awt.EventQueue.invokeLater(  
464 new Runnable() {  
465 public void run() {  
466 new EncodeFrame().setVisible(true);  
467 }  
468 });  
469 }  
470   
471 //Variables for encoding  
472 private String secretFile = "";  
473 private String[] innocentFiles = new String[2];  
474 private String directoryForStorage = "";  
475   
476 // Variables declaration - do not modify//GEN-BEGIN:variables  
477 private javax.swing.JButton browseButton1;  
478 private javax.swing.JButton browseButton2;  
479 private javax.swing.JButton browseButton3;  
480 private javax.swing.JButton browseButton4;  
481 private javax.swing.JButton cancelButton;  
482 private javax.swing.JFileChooser directoryChooser;  
483 private javax.swing.JButton encodeButton;  
484 private javax.swing.JTextField filename1;  
485 private javax.swing.JTextField filename2;  
486 private javax.swing.JFileChooser imageChooser;  
487 private javax.swing.JTextField innocentTextField1;  
488 private javax.swing.JTextField innocentTextField2;  
489 private javax.swing.JLabel jLabel1;  
490 private javax.swing.JLabel jLabel2;  
491 private javax.swing.JLabel jLabel3;  
492 private javax.swing.JLabel jLabel4;  
493 private javax.swing.JLabel jLabel5;  
494 private javax.swing.JLabel jLabel6;  
495 private javax.swing.JPanel jPanel1;  
496 private javax.swing.JPanel optionalPanel;  
497 private javax.swing.JPanel panel1;  
498 private javax.swing.JTextField secretTextField;  
499 private javax.swing.JTextField storageDirectoryTextField;  
500 // End of variables declaration//GEN-END:variables  
501 }  
502

DecodeFrame.java

1 /\*  
 2 \* To change this license header, choose License Headers in Project Properties.  
 3 \* To change this template file, choose Tools | Templates  
 4 \* and open the template in the editor.  
 5 \*/  
 6 package Masters\_Proj;  
 7   
 8 import java.awt.image.BufferedImage;  
 9 import java.io.File;  
 10 import java.io.IOException;  
 11 import javax.imageio.ImageIO;  
 12 import javax.swing.JFileChooser;  
 13   
 14 /\*\*  
 15 \*  
 16 \* @author allisonholt  
 17 \*/  
 18 public class DecodeFrame extends javax.swing.JFrame {  
 19   
 20 /\*\*  
 21 \* Creates new form DecodeFrame  
 22 \*/  
 23 public DecodeFrame() {  
 24 initComponents();  
 25 }  
 26   
 27 /\*\*  
 28 \* This method is called from within the constructor to initialize the form.  
 29 \* WARNING: Do NOT modify this code. The content of this method is always  
 30 \* regenerated by the Form Editor.  
 31 \*/  
 32 @SuppressWarnings("unchecked")  
 33 // <editor-fold defaultstate="collapsed" desc="Generated Code">//GEN-BEGIN:initComponents  
 34 private void initComponents() {  
 35   
 36 imageChooser = new javax.swing.JFileChooser();  
 37 directoryChooser = new javax.swing.JFileChooser();  
 38 jPanel1 = new javax.swing.JPanel();  
 39 jLabel1 = new javax.swing.JLabel();  
 40 encodedTextField1 = new javax.swing.JTextField();  
 41 browseButton1 = new javax.swing.JButton();  
 42 encodedTextField2 = new javax.swing.JTextField();  
 43 browseButton2 = new javax.swing.JButton();  
 44 jPanel2 = new javax.swing.JPanel();  
 45 jLabel2 = new javax.swing.JLabel();  
 46 jLabel3 = new javax.swing.JLabel();  
 47 stackedTextField = new javax.swing.JTextField();  
 48 jLabel4 = new javax.swing.JLabel();  
 49 storageDirectoryTextField = new javax.swing.JTextField();  
 50 browseButton3 = new javax.swing.JButton();  
 51 jButton2 = new javax.swing.JButton();  
 52 jButton3 = new javax.swing.JButton();  
 53   
 54 imageChooser.setDialogTitle("Choose an Image");  
 55 imageChooser.setFileFilter(new ImageCustomFilter());  
 56   
 57 directoryChooser.setDialogTitle("Choose a Directory");  
 58 directoryChooser.setFileFilter(new DirectoryCustomFilter());  
 59 directoryChooser.setFileSelectionMode(javax.swing.JFileChooser.DIRECTORIES\_ONLY);  
 60   
 61 setDefaultCloseOperation(javax.swing.WindowConstants.EXIT\_ON\_CLOSE);  
 62   
 63 jPanel1.setBorder(javax.swing.BorderFactory.createTitledBorder("Encoded Images"));  
 64   
 65 jLabel1.setText("Please select your two encoded image files:\*");  
 66   
 67 browseButton1.setText("Browse");  
 68 browseButton1.addActionListener(  
 69 new java.awt.event.ActionListener() {  
 70 public void actionPerformed(java.awt.event.ActionEvent evt) {  
 71 imageBrowsePressed(evt);  
 72 }  
 73 });  
 74   
 75 browseButton2.setText("Browse");  
 76 browseButton2.addActionListener(  
 77 new java.awt.event.ActionListener() {  
 78 public void actionPerformed(java.awt.event.ActionEvent evt) {  
 79 imageBrowsePressed(evt);  
 80 }  
 81 });  
 82   
 83 javax.swing.GroupLayout jPanel1Layout = new javax.swing.GroupLayout(jPanel1);  
 84 jPanel1.setLayout(jPanel1Layout);  
 85 jPanel1Layout.setHorizontalGroup(  
 86 jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)  
 87 .addGroup(jPanel1Layout.createSequentialGroup()  
 88 .addContainerGap()  
 89 .addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)  
 90 .addGroup(jPanel1Layout.createSequentialGroup()  
 91 .addComponent(jLabel1)  
 92 .addGap(0, 310, Short.MAX\_VALUE))  
 93 .addGroup(jPanel1Layout.createSequentialGroup()  
 94 .addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRAILING)  
 95 .addComponent(encodedTextField2)  
 96 .addComponent(encodedTextField1))  
 97 .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)  
 98 .addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)  
 99 .addComponent(browseButton1)  
100 .addComponent(browseButton2))))  
101 .addContainerGap())  
102 );  
103 jPanel1Layout.setVerticalGroup(  
104 jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)  
105 .addGroup(jPanel1Layout.createSequentialGroup()  
106 .addContainerGap()  
107 .addComponent(jLabel1)  
108 .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)  
109 .addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)  
110 .addComponent(encodedTextField1, javax.swing.GroupLayout.PREFERRED\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.PREFERRED\_SIZE)  
111 .addComponent(browseButton1))  
112 .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)  
113 .addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)  
114 .addComponent(encodedTextField2, javax.swing.GroupLayout.PREFERRED\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.PREFERRED\_SIZE)  
115 .addComponent(browseButton2))  
116 .addContainerGap(javax.swing.GroupLayout.DEFAULT\_SIZE, Short.MAX\_VALUE))  
117 );  
118   
119 jPanel2.setBorder(javax.swing.BorderFactory.createTitledBorder("Optional"));  
120   
121 jLabel2.setText("File Name for Decrypted Secret:");  
122   
123 jLabel3.setText("Name (without extension):");  
124   
125 jLabel4.setText("Directory for Decrypted Image:");  
126   
127 browseButton3.setText("Browse");  
128 browseButton3.addActionListener(  
129 new java.awt.event.ActionListener() {  
130 public void actionPerformed(java.awt.event.ActionEvent evt) {  
131 directoryBrowsePressed(evt);  
132 }  
133 });  
134   
135 javax.swing.GroupLayout jPanel2Layout = new javax.swing.GroupLayout(jPanel2);  
136 jPanel2.setLayout(jPanel2Layout);  
137 jPanel2Layout.setHorizontalGroup(  
138 jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)  
139 .addGroup(jPanel2Layout.createSequentialGroup()  
140 .addContainerGap()  
141 .addGroup(jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)  
142 .addGroup(jPanel2Layout.createSequentialGroup()  
143 .addComponent(jLabel2)  
144 .addGap(0, 0, Short.MAX\_VALUE))  
145 .addGroup(jPanel2Layout.createSequentialGroup()  
146 .addGap(6, 6, 6)  
147 .addGroup(jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)  
148 .addGroup(jPanel2Layout.createSequentialGroup()  
149 .addComponent(jLabel3)  
150 .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)  
151 .addComponent(stackedTextField))  
152 .addGroup(jPanel2Layout.createSequentialGroup()  
153 .addComponent(jLabel4)  
154 .addGap(0, 0, Short.MAX\_VALUE))  
155 .addGroup(jPanel2Layout.createSequentialGroup()  
156 .addComponent(storageDirectoryTextField)  
157 .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)  
158 .addComponent(browseButton3)))))  
159 .addContainerGap())  
160 );  
161 jPanel2Layout.setVerticalGroup(  
162 jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)  
163 .addGroup(jPanel2Layout.createSequentialGroup()  
164 .addContainerGap()  
165 .addComponent(jLabel2)  
166 .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)  
167 .addGroup(jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)  
168 .addComponent(jLabel3)  
169 .addComponent(stackedTextField, javax.swing.GroupLayout.PREFERRED\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.PREFERRED\_SIZE))  
170 .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)  
171 .addGroup(jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRAILING)  
172 .addGroup(jPanel2Layout.createSequentialGroup()  
173 .addComponent(jLabel4)  
174 .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)  
175 .addComponent(storageDirectoryTextField, javax.swing.GroupLayout.PREFERRED\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.PREFERRED\_SIZE))  
176 .addComponent(browseButton3))  
177 .addContainerGap(19, Short.MAX\_VALUE))  
178 );  
179   
180 jButton2.setText("Cancel");  
181 jButton2.addActionListener(  
182 new java.awt.event.ActionListener() {  
183 public void actionPerformed(java.awt.event.ActionEvent evt) {  
184 cancelPressed(evt);  
185 }  
186 });  
187   
188 jButton3.setText("Decode");  
189 jButton3.addActionListener(  
190 new java.awt.event.ActionListener() {  
191 public void actionPerformed(java.awt.event.ActionEvent evt) {  
192 decodePressed(evt);  
193 }  
194 });  
195   
196 javax.swing.GroupLayout layout = new javax.swing.GroupLayout(getContentPane());  
197 getContentPane().setLayout(layout);  
198 layout.setHorizontalGroup(  
199 layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)  
200 .addGroup(layout.createSequentialGroup()  
201 .addContainerGap()  
202 .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)  
203 .addComponent(jPanel2, javax.swing.GroupLayout.Alignment.TRAILING, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, Short.MAX\_VALUE)  
204 .addComponent(jPanel1, javax.swing.GroupLayout.Alignment.TRAILING, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, Short.MAX\_VALUE)  
205 .addGroup(javax.swing.GroupLayout.Alignment.TRAILING, layout.createSequentialGroup()  
206 .addGap(0, 0, Short.MAX\_VALUE)  
207 .addComponent(jButton3)  
208 .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)  
209 .addComponent(jButton2)))  
210 .addContainerGap())  
211 );  
212 layout.setVerticalGroup(  
213 layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)  
214 .addGroup(layout.createSequentialGroup()  
215 .addContainerGap()  
216 .addComponent(jPanel1, javax.swing.GroupLayout.PREFERRED\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.PREFERRED\_SIZE)  
217 .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)  
218 .addComponent(jPanel2, javax.swing.GroupLayout.PREFERRED\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.PREFERRED\_SIZE)  
219 .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)  
220 .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)  
221 .addComponent(jButton2)  
222 .addComponent(jButton3))  
223 .addContainerGap(javax.swing.GroupLayout.DEFAULT\_SIZE, Short.MAX\_VALUE))  
224 );  
225   
226 pack();  
227 }// </editor-fold>//GEN-END:initComponents  
228   
229 private void cancelPressed(java.awt.event.ActionEvent evt) {//GEN-FIRST:event\_cancelPressed  
230 // TODO add your handling code here:  
231 this.setVisible(false);  
232 new MainFrame().setVisible(true);  
233 }//GEN-LAST:event\_cancelPressed  
234   
235 private void imageBrowsePressed(java.awt.event.ActionEvent evt) {//GEN-FIRST:event\_imageBrowsePressed  
236 // TODO add your handling code here:  
237 int returnVal = imageChooser.showOpenDialog(this);  
238 if(returnVal == JFileChooser.APPROVE\_OPTION)  
239 {  
240 File imageFile = imageChooser.getSelectedFile();  
241 if(evt.getSource() == browseButton1)  
242 {  
243 encodedTextField1.setText(imageFile.getAbsolutePath());  
244 shareFiles[0] = imageFile.getAbsolutePath();  
245 }  
246 else if(evt.getSource() == browseButton2)  
247 {  
248 encodedTextField2.setText(imageFile.getAbsolutePath());  
249 shareFiles[1] = imageFile.getAbsolutePath();  
250 }  
251 }  
252 }//GEN-LAST:event\_imageBrowsePressed  
253   
254 private void directoryBrowsePressed(java.awt.event.ActionEvent evt) {//GEN-FIRST:event\_directoryBrowsePressed  
255 // TODO add your handling code here:  
256 int returnVal = directoryChooser.showOpenDialog(this);  
257 if(returnVal == JFileChooser.APPROVE\_OPTION)  
258 {  
259 File dir = directoryChooser.getSelectedFile();  
260 if(evt.getSource() == browseButton3)  
261 {  
262 storageDirectoryTextField.setText(dir.getAbsolutePath());  
263 directoryForStorage = dir.getAbsolutePath();  
264 }  
265 }  
266 }//GEN-LAST:event\_directoryBrowsePressed  
267   
268 private void decodePressed(java.awt.event.ActionEvent evt) {//GEN-FIRST:event\_decodePressed  
269 // TODO add your handling code here:  
270 BufferedImage[] sharesEVCS = new BufferedImage[2];  
271 boolean fileFound = false;  
272   
273 for(int i = 0; i < 2; i++)  
274 {  
275 try  
276 {  
277 sharesEVCS[i] = ImageIO.read(new File(shareFiles[i]));  
278 fileFound = true;  
279 }  
280 catch(IOException e)  
281 {  
282 //Add alert  
283 fileFound = false;  
284 }  
285 }  
286   
287 if(fileFound)  
288 {  
289 ExtendedVCS myEVCS = new ExtendedVCS(sharesEVCS);  
290 myEVCS.decryptImage();  
291   
292 String decodedFileName;  
293   
294 if(stackedTextField.getText().equals(""))  
295 {  
296 //Get path to users desktop  
297 //BUG!!! Not working.  
298 decodedFileName = directoryForStorage + "/secretMsg.png";  
299 //makeDir = false;  
300 }  
301 else  
302 {  
303 decodedFileName = directoryForStorage + "/" + stackedTextField.getText() + ".png";  
304 }  
305   
306 //boolean makeDir = true;  
307 if(storageDirectoryTextField.getText().equals(""))  
308 {  
309 //Get path to users desktop  
310 //BUG!!! Not working.  
311 directoryForStorage = "C:/Users/allisonholt/Desktop";  
312 //makeDir = false;  
313 }  
314   
315 //if(makeDir)  
316 //{  
317 //File directory = new File(directoryForStorage);  
318 //}  
319   
320 String[] shareFiles = new String[2];  
321   
322 try  
323 {  
324 BufferedImage decryptImage = new BufferedImage(myEVCS.getImgWidth(), myEVCS.getImgHeight(), BufferedImage.TYPE\_INT\_ARGB);  
325 decryptImage.setRGB(0, 0, myEVCS.getImgWidth(), myEVCS.getImgHeight(), myEVCS.getDecryptImgPixels(), 0, myEVCS.getImgWidth());  
326   
327 File tempOutput = new File(decodedFileName);  
328 ImageIO.write(decryptImage, "png", tempOutput);  
329 }  
330 catch(IOException e)  
331 {  
332 //Add alert  
333 }  
334   
335 }  
336 }//GEN-LAST:event\_decodePressed  
337   
338 /\*\*  
339 \* @param args the command line arguments  
340 \*/  
341 public static void main(String args[]) {  
342 /\* Set the Nimbus look and feel \*/  
343 //<editor-fold defaultstate="collapsed" desc=" Look and feel setting code (optional) ">  
344 /\* If Nimbus (introduced in Java SE 6) is not available, stay with the default look and feel.  
345 \* For details see http://download.oracle.com/javase/tutorial/uiswing/lookandfeel/plaf.html   
346 \*/  
347 try {  
348 for (javax.swing.UIManager.LookAndFeelInfo info : javax.swing.UIManager.getInstalledLookAndFeels()) {  
349 if ("Nimbus".equals(info.getName())) {  
350 javax.swing.UIManager.setLookAndFeel(info.getClassName());  
351 break;  
352 }  
353 }  
354 }   
355 catch (ClassNotFoundException ex) {  
356 java.util.logging.Logger.getLogger(DecodeFrame.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);  
357 }   
358 catch (InstantiationException ex) {  
359 java.util.logging.Logger.getLogger(DecodeFrame.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);  
360 }   
361 catch (IllegalAccessException ex) {  
362 java.util.logging.Logger.getLogger(DecodeFrame.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);  
363 }   
364 catch (javax.swing.UnsupportedLookAndFeelException ex) {  
365 java.util.logging.Logger.getLogger(DecodeFrame.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);  
366 }  
367 //</editor-fold>  
368   
369 /\* Create and display the form \*/  
370 java.awt.EventQueue.invokeLater(  
371 new Runnable() {  
372 public void run() {  
373 new DecodeFrame().setVisible(true);  
374 }  
375 });  
376 }  
377 //Variables for decoding  
378 private String[] shareFiles = new String[2];  
379 private String directoryForStorage = "";  
380   
381 // Variables declaration - do not modify//GEN-BEGIN:variables  
382 private javax.swing.JButton browseButton1;  
383 private javax.swing.JButton browseButton2;  
384 private javax.swing.JButton browseButton3;  
385 private javax.swing.JFileChooser directoryChooser;  
386 private javax.swing.JTextField encodedTextField1;  
387 private javax.swing.JTextField encodedTextField2;  
388 private javax.swing.JFileChooser imageChooser;  
389 private javax.swing.JButton jButton2;  
390 private javax.swing.JButton jButton3;  
391 private javax.swing.JLabel jLabel1;  
392 private javax.swing.JLabel jLabel2;  
393 private javax.swing.JLabel jLabel3;  
394 private javax.swing.JLabel jLabel4;  
395 private javax.swing.JPanel jPanel1;  
396 private javax.swing.JPanel jPanel2;  
397 private javax.swing.JTextField stackedTextField;  
398 private javax.swing.JTextField storageDirectoryTextField;  
399 // End of variables declaration//GEN-END:variables  
400 }  
401

ImageCustomFilter.java

1 /\*  
 2 \* To change this license header, choose License Headers in Project Properties.  
 3 \* To change this template file, choose Tools | Templates  
 4 \* and open the template in the editor.  
 5 \*/  
 6 package Masters\_Proj;  
 7   
 8 import java.io.File;  
 9   
10 /\*\*  
11 \*  
12 \* @author allisonholt  
13 \*/  
14 public class ImageCustomFilter extends javax.swing.filechooser.FileFilter {  
15   
16 @Override  
17 public boolean accept(File file)  
18 {  
19 //allow only image file  
20 return file.isDirectory() || file.getAbsolutePath().endsWith(".png")  
21 || file.getAbsolutePath().endsWith(".jpeg");  
22 }  
23   
24 @Override  
25 public String getDescription()  
26 {  
27 return "Image files (\*.png, \*.jpeg)";  
28 }  
29   
30 }  
31

DirectoryCustomFilter.java

1 /\*  
 2 \* To change this license header, choose License Headers in Project Properties.  
 3 \* To change this template file, choose Tools | Templates  
 4 \* and open the template in the editor.  
 5 \*/  
 6 package Masters\_Proj;  
 7   
 8 import java.io.File;  
 9   
10 /\*\*  
11 \*  
12 \* @author allisonholt  
13 \*/  
14 public class DirectoryCustomFilter extends javax.swing.filechooser.FileFilter{  
15   
16 @Override  
17 public boolean accept(File file)  
18 {  
19 //allow only image file  
20 return file.isDirectory();  
21 }  
22   
23 @Override  
24 public String getDescription()  
25 {  
26 return "File Directory";  
27 }  
28 }  
29

ExtendedVCS.java

1 /\*  
 2 \* To change this license header, choose License Headers in Project Properties.  
 3 \* To change this template file, choose Tools | Templates  
 4 \* and open the template in the editor.  
 5 \*/  
 6 package Masters\_Proj;  
 7 import java.awt.Color;  
 8 import java.awt.image.BufferedImage;  
 9 import java.util.Arrays;  
 10 import java.util.Random;  
 11   
 12 /\*\*  
 13 \*  
 14 \* @author allisonholt  
 15 \*/  
 16 public class ExtendedVCS   
 17 {  
 18 private int k;  
 19 private int n;  
 20 private int imgWidth;  
 21 private int imgHeight;  
 22 private int numColumns;  
 23 private BufferedImage secretMsg;  
 24 private BufferedImage[] innocentShares;  
 25 //private int[2][] shareOrigRGBPixels;  
 26 private int[][] encryptedShareRGB;  
 27   
 28 private int numSharesToDecrypt;  
 29 private BufferedImage[] sharesToDecrypt;  
 30 private int[] secretMsgPixels;  
 31   
 32 //Matrices  
 33 int[][] wwSw = new int[][]{  
 34 {1, 0, 0, 1},  
 35 {1, 0, 0, 0} };  
 36 int[][] wwSb = new int[][]{  
 37 {1, 0, 0, 1},  
 38 {0, 1, 1, 0} };  
 39 int[][] wbSw = new int [][]{  
 40 {1, 0, 0, 1},  
 41 {1, 0, 1, 1} };  
 42 int[][] wbSb = new int [][]{  
 43 {1, 0, 0, 1},  
 44 {0, 1, 1, 1} };  
 45 int[][] bwSw = new int [][]{  
 46 {1, 0, 1, 1},  
 47 {1, 0, 1, 0} };  
 48 int[][] bwSb = new int [][]{  
 49 {1, 0, 1, 1},  
 50 {0, 1, 1, 0} };  
 51 int[][] bbSw = new int [][]{  
 52 {1, 0, 1, 1},  
 53 {1, 0, 1, 1} };  
 54 int[][] bbSb = new int [][]{  
 55 {1, 0, 1, 1},  
 56 {0, 1, 1, 1} };  
 57   
 58   
 59 //For encryption purposes  
 60 ExtendedVCS(BufferedImage secretMsgIn, BufferedImage[] innocentSharesIn)  
 61 {  
 62 k = 2;  
 63 n = 2;  
 64 secretMsg = secretMsgIn;  
 65 imgWidth = secretMsg.getWidth();  
 66 imgHeight = secretMsg.getHeight();  
 67 innocentShares = innocentSharesIn;  
 68 }  
 69   
 70 //For decryption purposes  
 71 ExtendedVCS(BufferedImage[] shareImgs)  
 72 {  
 73 numSharesToDecrypt = 2;  
 74 sharesToDecrypt = shareImgs;  
 75 imgWidth = shareImgs[0].getWidth();  
 76 imgHeight = shareImgs[0].getHeight();  
 77 }  
 78   
 79 int getImgWidth()  
 80 {  
 81 return imgWidth;  
 82 }  
 83   
 84 int getImgHeight()  
 85 {  
 86 return imgHeight;  
 87 }  
 88   
 89 int[][] getRGBPixelsForShares()  
 90 {  
 91 return encryptedShareRGB;  
 92 }  
 93   
 94 int[] getDecryptImgPixels()  
 95 {  
 96 return secretMsgPixels;  
 97 }  
 98   
 99 void encryptImage()  
100 {  
101 int[] secretRGB = new int[imgWidth \* imgHeight];  
102 int[][] shareOrigRGB = new int[2][imgWidth \* imgHeight];  
103 secretMsg.getRGB(0, 0, imgWidth, imgHeight, secretRGB, 0, imgWidth);  
104 innocentShares[0].getRGB(0, 0, imgWidth, imgHeight, shareOrigRGB[0], 0, imgWidth);  
105 innocentShares[1].getRGB(0, 0, imgWidth, imgHeight, shareOrigRGB[1], 0, imgWidth);  
106 createPixelsOfShares(secretRGB, shareOrigRGB);  
107 }  
108   
109 void createPixelsOfShares(int[] secretImgRGB, int[][] shareOriginalRGB)  
110 {  
111 encryptedShareRGB = new int[2][imgWidth \* imgHeight];  
112   
113 for(int i = 0; i < secretImgRGB.length; i++)  
114 {  
115 int redVal = (secretImgRGB[i] & 0x00ff0000) >> 16;  
116 int greenVal = (secretImgRGB[i] & 0x0000ff00) >> 8;  
117 int blueVal = (secretImgRGB[i] & 0x000000ff);  
118 Pixel orig = new Pixel(redVal, greenVal, blueVal);  
119   
120 redVal = (shareOriginalRGB[0][i] & 0x00ff0000) >> 16;  
121 greenVal = (shareOriginalRGB[0][i] & 0x0000ff00) >> 8;  
122 blueVal = (shareOriginalRGB[0][i] & 0x000000ff);  
123 Pixel innocent0 = new Pixel(redVal, greenVal, blueVal);  
124   
125 redVal = (shareOriginalRGB[1][i] & 0x00ff0000) >> 16;  
126 greenVal = (shareOriginalRGB[1][i] & 0x0000ff00) >> 8;  
127 blueVal = (shareOriginalRGB[1][i] & 0x000000ff);  
128 Pixel innocent1 = new Pixel(redVal, greenVal, blueVal);  
129   
130 Random randomGen = new Random();  
131 int randomColumn = randomGen.nextInt(4);  
132   
133 //If pixel is white  
134 if(innocent0.isMoreWhiteThanBlack())  
135 {   
136 if(innocent1.isMoreWhiteThanBlack())  
137 {  
138 if(orig.isMoreWhiteThanBlack())  
139 {  
140 //Want to use matrix wwSw  
141 if(wwSw[0][randomColumn] == 0)  
142 encryptedShareRGB[0][i] = Color.WHITE.getRGB();  
143 else  
144 encryptedShareRGB[0][i] = Color.BLACK.getRGB();  
145   
146 if(wwSw[1][randomColumn] == 0)  
147 encryptedShareRGB[1][i] = Color.WHITE.getRGB();  
148 else  
149 encryptedShareRGB[1][i] = Color.BLACK.getRGB();  
150 }  
151 else  
152 {  
153 //Want to use matrix wwSb  
154 if(wwSb[0][randomColumn] == 0)  
155 encryptedShareRGB[0][i] = Color.WHITE.getRGB();  
156 else  
157 encryptedShareRGB[0][i] = Color.BLACK.getRGB();  
158   
159 if(wwSb[1][randomColumn] == 0)  
160 encryptedShareRGB[1][i] = Color.WHITE.getRGB();  
161 else  
162 encryptedShareRGB[1][i] = Color.BLACK.getRGB();  
163 }  
164 }  
165 else  
166 {  
167 if(orig.isMoreWhiteThanBlack())  
168 {  
169 //Want to use matrix wbSw  
170 if(wbSw[0][randomColumn] == 0)  
171 encryptedShareRGB[0][i] = Color.WHITE.getRGB();  
172 else  
173 encryptedShareRGB[0][i] = Color.BLACK.getRGB();  
174   
175 if(wbSw[1][randomColumn] == 0)  
176 encryptedShareRGB[1][i] = Color.WHITE.getRGB();  
177 else  
178 encryptedShareRGB[1][i] = Color.BLACK.getRGB();  
179 }  
180 else  
181 {  
182 //Want to use matrix wbSb  
183 if(wbSb[0][randomColumn] == 0)  
184 encryptedShareRGB[0][i] = Color.WHITE.getRGB();  
185 else  
186 encryptedShareRGB[0][i] = Color.BLACK.getRGB();  
187   
188 if(wbSb[1][randomColumn] == 0)  
189 encryptedShareRGB[1][i] = Color.WHITE.getRGB();  
190 else  
191 encryptedShareRGB[1][i] = Color.BLACK.getRGB();  
192 }  
193 }  
194 }  
195 else  
196 {  
197 if(innocent1.isMoreWhiteThanBlack())  
198 {  
199 if(orig.isMoreWhiteThanBlack())  
200 {  
201 //Want to use matrix bwSw  
202 if(bwSw[0][randomColumn] == 0)  
203 encryptedShareRGB[0][i] = Color.WHITE.getRGB();  
204 else  
205 encryptedShareRGB[0][i] = Color.BLACK.getRGB();  
206   
207 if(bwSw[1][randomColumn] == 0)  
208 encryptedShareRGB[1][i] = Color.WHITE.getRGB();  
209 else  
210 encryptedShareRGB[1][i] = Color.BLACK.getRGB();  
211 }  
212 else  
213 {  
214 //Want to use matrix bwSb  
215 if(bwSb[0][randomColumn] == 0)  
216 encryptedShareRGB[0][i] = Color.WHITE.getRGB();  
217 else  
218 encryptedShareRGB[0][i] = Color.BLACK.getRGB();  
219   
220 if(bwSb[1][randomColumn] == 0)  
221 encryptedShareRGB[1][i] = Color.WHITE.getRGB();  
222 else  
223 encryptedShareRGB[1][i] = Color.BLACK.getRGB();  
224 }  
225 }  
226 else  
227 {  
228 if(orig.isMoreWhiteThanBlack())  
229 {  
230 //Want to use matrix bbSw  
231 if(bbSw[0][randomColumn] == 0)  
232 encryptedShareRGB[0][i] = Color.WHITE.getRGB();  
233 else  
234 encryptedShareRGB[0][i] = Color.BLACK.getRGB();  
235   
236 if(bbSw[1][randomColumn] == 0)  
237 encryptedShareRGB[1][i] = Color.WHITE.getRGB();  
238 else  
239 encryptedShareRGB[1][i] = Color.BLACK.getRGB();  
240 }  
241 else  
242 {  
243 //Want to use matrix bbSb  
244 if(bbSb[0][randomColumn] == 0)  
245 encryptedShareRGB[0][i] = Color.WHITE.getRGB();  
246 else  
247 encryptedShareRGB[0][i] = Color.BLACK.getRGB();  
248   
249 if(bbSb[1][randomColumn] == 0)  
250 encryptedShareRGB[1][i] = Color.WHITE.getRGB();  
251 else  
252 encryptedShareRGB[1][i] = Color.BLACK.getRGB();  
253 }  
254 }  
255 }  
256 }  
257 }  
258   
259 void decryptImage()  
260 {  
261 //Make a 2d array of pixel arrays  
262 int[][] pixelsToCompare = new int[numSharesToDecrypt][imgWidth \* imgHeight];  
263 secretMsgPixels = new int[imgWidth \* imgHeight];  
264   
265 //getRGB pixels of BufferedImages  
266 for(int i = 0; i < numSharesToDecrypt; i++)  
267 {  
268 sharesToDecrypt[i].getRGB(0, 0, imgWidth, imgHeight, pixelsToCompare[i], 0, imgWidth);  
269 }  
270   
271 //Logical OR pixel with all three share values  
272 int numOfPixels = pixelsToCompare[0].length;  
273 for(int i = 0; i < numOfPixels; i++)  
274 {  
275 int pixelColor = 0;  
276 for(int j = 0; j < numSharesToDecrypt; j++)  
277 {  
278 if(pixelsToCompare[j][i] == Color.WHITE.getRGB())  
279 {  
280 pixelColor = pixelColor | 0;  
281 }  
282 else  
283 {  
284 pixelColor = pixelColor | 1;  
285 }  
286 }  
287   
288 //Store the result in an array after converting to WHITE and BLACK  
289 if(pixelColor == 1)  
290 {  
291 secretMsgPixels[i] = Color.BLACK.getRGB();  
292 }  
293 else  
294 {  
295 secretMsgPixels[i] = Color.WHITE.getRGB();  
296 }  
297 }  
298 }  
299   
300 }  
301

Pixel.java

1 /\*  
 2 \* To change this license header, choose License Headers in Project Properties.  
 3 \* To change this template file, choose Tools | Templates  
 4 \* and open the template in the editor.  
 5 \*/  
 6 package Masters\_Proj;  
 7 import java.awt.Color;  
 8   
 9 /\*\*  
10 \*  
11 \* @author allisonholt  
12 \*/  
13 public class Pixel   
14 {  
15   
16 private int redVal;  
17 private int greenVal;  
18 private int blueVal;  
19   
20 public Pixel(int redIn, int greenIn, int blueIn)  
21 {  
22 redVal = redIn;  
23 greenVal = greenIn;  
24 blueVal = blueIn;  
25 }  
26   
27 //Used to determine if pixel is closer to white than black  
28 public boolean isMoreWhiteThanBlack()  
29 {  
30 int sum = redVal + greenVal + blueVal;  
31 int avg = sum / 3;  
32 return (avg >= 128);  
33 }  
34   
35 }  
36