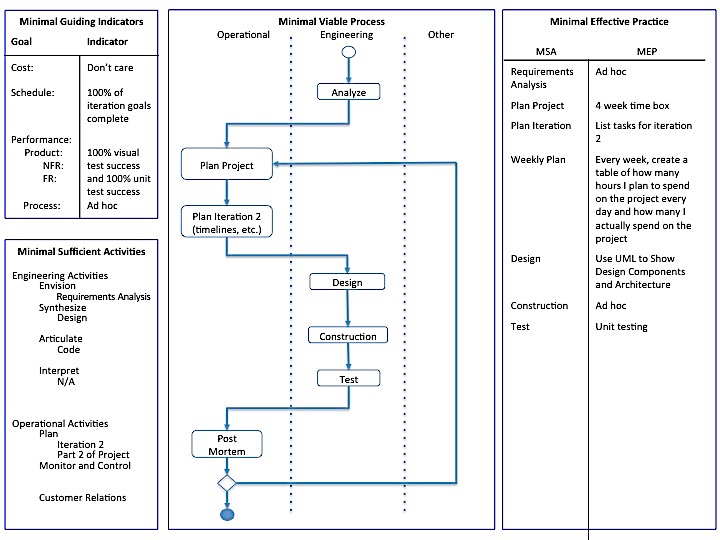
Iteration 4

# Process for Iteration 4



As mentioned in Iteration 3, a weekly plan will be added to this iteration. At the beginning of each week, I will plan how many hours a day I will spend working on the project. Throughout the week, I will also be keeping track of how many hours I actually spend on the project. This Minimal Effective Practice should help with time management. This iteration will be less productive since finals, Christmas, and New Years are all happening.

# Requirements Analysis

The requirements have not changed since Iteration 1. They are as follows:

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 (Sept. 11 – Oct. 9):

* 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 (Oct. 10 – Nov. 6):

* Research visual cryptography schemes with gray scale images

Iteration 3 (Nov. 7 – Dec. 4):

* Implement the visual cryptography scheme with grayscale images
* Begin researching how to modify the current algorithm to handle color images

Iteration 4 (Dec. 5 – Jan. 1):

* Add the ability to encode and decode multicolor images

Iteration 5 (Jan. 2 – Jan. 29):

* Add unit tests for the components of the visual cryptography tool
* Analyze the tool and look for ways to improve efficiency (performance and memory storage)
* Add features to project to help boost robustness (i.e. add in checks to keep the user from breaking the tool easily)

# Plans for Iteration 4

* Add the ability to encode and decode multicolor images

# Weekly Plans

Week 1:

|  |  |  |
| --- | --- | --- |
| Day | Expected Hours | Actual Hours |
| Saturday, December 5th | 0 | 0 |
| Sunday, December 6th | 0 | 0 |
| Monday, December 7th | 0 | 0 |
| Tuesday, December 8th | 4 | 2 |
| Wednesday, December 9th | 0 | 0 |
| Thursday, December 10th | 0 | 0 |
| Friday, December 11th | 0 | 0 |

Week 2:

|  |  |  |
| --- | --- | --- |
| Day | Expected Hours | Actual Hours |
| Saturday, December 12th | 0 | 0 |
| Sunday, December 13th | 0 | 0 |
| Monday, December 14th | 4 | 0 |
| Tuesday, December 15th | 2 | 0 |
| Wednesday, December 16th | 0 | 0 |
| Thursday, December 17th | 0 | 0 |
| Friday, December 18th | 0 | 0 |

Week 3:

|  |  |  |
| --- | --- | --- |
| Day | Expected Hours | Actual Hours |
| Saturday, December 19th | 0 | 0 |
| Sunday, December 20th | 0 | 0 |
| Monday, December 21st | 0 | 0 |
| Tuesday, December 22nd | 0 | 0 |
| Wednesday, December 23rd | 0 | 0 |
| Thursday, December 24th | 0 | 0 |
| Friday, December 25th | 0 | 0 |

Week 4:

|  |  |  |
| --- | --- | --- |
| Day | Expected Hours | Actual Hours |
| Saturday, December 26th | 0 | 0 |
| Sunday, December 27th | 0 | 0 |
| Monday, December 28th | 4 |  |
| Tuesday, December 29th | 4 |  |
| Wednesday, December 30th | 4 |  |
| Thursday, December 31st | 4 |  |
| Friday, January 1st | 4 |  |

# Design

Figure 1: UML Diagram from Iteration 1

Note the design has not changed since Iteration 2.

Before changing the Java files to handle the gray scale images, I exported the PlantUML diagram of the visual cryptography tool. Figure 1 shows the class relations.

RESEARCH ON COLOR IMAGES

# Construction

TBD

# Test

As stated in the Design section of this iteration, my algorithm for encrypting the secret message involved randomly splitting up the dark concentration and adding the darkness to the innocent images. The secret image and the innocent images are shown in Figure 2. The formula for calculating the embedded pixel concentration in my algorithm was 255 – (innocent concentration + secret concentration). Upon performing the visual tests, the embedded files, displayed in Figure 3, showed that pixels that were white in the innocent image became a dark gray.

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

|  |
| --- |
|  |
| Figure 3: Encoded Share A (left), Encoded Share B (right) |

The decryption of the encoded shares resulted in the secret message being revealed, but the message is not as pronounced as I would like it to be. The other two innocent images are still visible within the decrypted image shown in Figure 4.



Figure 4: Result of decoding shares A and B from Figure 3.

Since the test results from the initial algorithm was poor, I updated the calculation of the embedded pixel concentration. The revised formula for calculating the embedded pixel concentration was the average of the gray concentration between the innocent image and the portion of the secret concentration. Figure 5 below shows the results from the revised algorithm using the innocent and secret images shown in Figure 2.

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

The encrypted images clearly show the secret and the innocent image. The formula used to calculate the decrypted pixels was 255 – (encryptedShare1 + encryptedShare2), which did not work well for the grayscale images. The results of the decrypting the shares in Figure 5 are shown in Figure 6. The secret message was difficult to see in the decrypted image. More discussion will occur in the post mortem.



Figure 6: Result of decoding shares A and B from Figure 5.

# Post Mortem

The algorithm needs a better way to encrypt and decrypt the pixels when the pixels are more than just black or white. First the decryption algorithm should be edited to better reveal the secret image. Also, I need to find a better way to make the encrypted pixels blend together. After discussing these issues with my advisor, I have decided to push on to adding color to the encryption and decryption process first. Once that capability has been added, I will work on improving the blending of the encrypted pixels.

# Source Code

MainFrame.java

TBD

EncodeFrame.java

TBD

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 import javax.swing.JOptionPane;  
 14   
 15 /\*\*  
 16 \*  
 17 \* @author allisonholt  
 18 \*/  
 19 public class DecodeFrame extends javax.swing.JFrame {  
 20   
 21 /\*\*  
 22 \* Creates new form DecodeFrame  
 23 \*/  
 24 public DecodeFrame() {  
 25 initComponents();  
 26 }  
 27   
 28 /\*\*  
 29 \* This method is called from within the constructor to initialize the form.  
 30 \* WARNING: Do NOT modify this code. The content of this method is always  
 31 \* regenerated by the Form Editor.  
 32 \*/  
 33 @SuppressWarnings("unchecked")  
 34 // <editor-fold defaultstate="collapsed" desc="Generated Code">//GEN-BEGIN:initComponents  
 35 private void initComponents() {  
 36   
 37 imageChooser = new javax.swing.JFileChooser();  
 38 directoryChooser = new javax.swing.JFileChooser();  
 39 jPanel1 = new javax.swing.JPanel();  
 40 jLabel1 = new javax.swing.JLabel();  
 41 encodedTextField1 = new javax.swing.JTextField();  
 42 browseButton1 = new javax.swing.JButton();  
 43 encodedTextField2 = new javax.swing.JTextField();  
 44 browseButton2 = new javax.swing.JButton();  
 45 jPanel2 = new javax.swing.JPanel();  
 46 jLabel2 = new javax.swing.JLabel();  
 47 jLabel3 = new javax.swing.JLabel();  
 48 stackedTextField = new javax.swing.JTextField();  
 49 jLabel4 = new javax.swing.JLabel();  
 50 storageDirectoryTextField = new javax.swing.JTextField();  
 51 browseButton3 = new javax.swing.JButton();  
 52 jButton2 = new javax.swing.JButton();  
 53 jButton3 = new javax.swing.JButton();  
 54   
 55 imageChooser.setDialogTitle("Choose an Image");  
 56 imageChooser.setFileFilter(new ImageCustomFilter());  
 57   
 58 directoryChooser.setDialogTitle("Choose a Directory");  
 59 directoryChooser.setFileFilter(new DirectoryCustomFilter());  
 60 directoryChooser.setFileSelectionMode(javax.swing.JFileChooser.DIRECTORIES\_ONLY);  
 61   
 62 setDefaultCloseOperation(javax.swing.WindowConstants.EXIT\_ON\_CLOSE);  
 63   
 64 jPanel1.setBorder(javax.swing.BorderFactory.createTitledBorder("Encoded Images"));  
 65   
 66 jLabel1.setText("Please select your two encoded image files:\*");  
 67   
 68 browseButton1.setText("Browse");  
 69 browseButton1.addActionListener(  
 70 new java.awt.event.ActionListener() {  
 71 public void actionPerformed(java.awt.event.ActionEvent evt) {  
 72 imageBrowsePressed(evt);  
 73 }  
 74 });  
 75   
 76 browseButton2.setText("Browse");  
 77 browseButton2.addActionListener(  
 78 new java.awt.event.ActionListener() {  
 79 public void actionPerformed(java.awt.event.ActionEvent evt) {  
 80 imageBrowsePressed(evt);  
 81 }  
 82 });  
 83   
 84 javax.swing.GroupLayout jPanel1Layout = new javax.swing.GroupLayout(jPanel1);  
 85 jPanel1.setLayout(jPanel1Layout);  
 86 jPanel1Layout.setHorizontalGroup(  
 87 jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)  
 88 .addGroup(jPanel1Layout.createSequentialGroup()  
 89 .addContainerGap()  
 90 .addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)  
 91 .addGroup(jPanel1Layout.createSequentialGroup()  
 92 .addComponent(jLabel1)  
 93 .addGap(0, 310, Short.MAX\_VALUE))  
 94 .addGroup(jPanel1Layout.createSequentialGroup()  
 95 .addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRAILING)  
 96 .addComponent(encodedTextField2)  
 97 .addComponent(encodedTextField1))  
 98 .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)  
 99 .addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)  
100 .addComponent(browseButton1)  
101 .addComponent(browseButton2))))  
102 .addContainerGap())  
103 );  
104 jPanel1Layout.setVerticalGroup(  
105 jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)  
106 .addGroup(jPanel1Layout.createSequentialGroup()  
107 .addContainerGap()  
108 .addComponent(jLabel1)  
109 .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)  
110 .addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)  
111 .addComponent(encodedTextField1, javax.swing.GroupLayout.PREFERRED\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.PREFERRED\_SIZE)  
112 .addComponent(browseButton1))  
113 .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)  
114 .addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)  
115 .addComponent(encodedTextField2, javax.swing.GroupLayout.PREFERRED\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.PREFERRED\_SIZE)  
116 .addComponent(browseButton2))  
117 .addContainerGap(javax.swing.GroupLayout.DEFAULT\_SIZE, Short.MAX\_VALUE))  
118 );  
119   
120 jPanel2.setBorder(javax.swing.BorderFactory.createTitledBorder("Optional"));  
121   
122 jLabel2.setText("File Name for Decrypted Secret:");  
123   
124 jLabel3.setText("Name (without extension):");  
125   
126 jLabel4.setText("Directory for Decrypted Image:");  
127   
128 browseButton3.setText("Browse");  
129 browseButton3.addActionListener(  
130 new java.awt.event.ActionListener() {  
131 public void actionPerformed(java.awt.event.ActionEvent evt) {  
132 directoryBrowsePressed(evt);  
133 }  
134 });  
135   
136 javax.swing.GroupLayout jPanel2Layout = new javax.swing.GroupLayout(jPanel2);  
137 jPanel2.setLayout(jPanel2Layout);  
138 jPanel2Layout.setHorizontalGroup(  
139 jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)  
140 .addGroup(jPanel2Layout.createSequentialGroup()  
141 .addContainerGap()  
142 .addGroup(jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)  
143 .addGroup(jPanel2Layout.createSequentialGroup()  
144 .addComponent(jLabel2)  
145 .addGap(0, 0, Short.MAX\_VALUE))  
146 .addGroup(jPanel2Layout.createSequentialGroup()  
147 .addGap(6, 6, 6)  
148 .addGroup(jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)  
149 .addGroup(jPanel2Layout.createSequentialGroup()  
150 .addComponent(jLabel3)  
151 .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)  
152 .addComponent(stackedTextField))  
153 .addGroup(jPanel2Layout.createSequentialGroup()  
154 .addComponent(jLabel4)  
155 .addGap(0, 0, Short.MAX\_VALUE))  
156 .addGroup(jPanel2Layout.createSequentialGroup()  
157 .addComponent(storageDirectoryTextField)  
158 .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)  
159 .addComponent(browseButton3)))))  
160 .addContainerGap())  
161 );  
162 jPanel2Layout.setVerticalGroup(  
163 jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)  
164 .addGroup(jPanel2Layout.createSequentialGroup()  
165 .addContainerGap()  
166 .addComponent(jLabel2)  
167 .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)  
168 .addGroup(jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)  
169 .addComponent(jLabel3)  
170 .addComponent(stackedTextField, javax.swing.GroupLayout.PREFERRED\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.PREFERRED\_SIZE))  
171 .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)  
172 .addGroup(jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRAILING)  
173 .addGroup(jPanel2Layout.createSequentialGroup()  
174 .addComponent(jLabel4)  
175 .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)  
176 .addComponent(storageDirectoryTextField, javax.swing.GroupLayout.PREFERRED\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.PREFERRED\_SIZE))  
177 .addComponent(browseButton3))  
178 .addContainerGap(19, Short.MAX\_VALUE))  
179 );  
180   
181 jButton2.setText("Cancel");  
182 jButton2.addActionListener(  
183 new java.awt.event.ActionListener() {  
184 public void actionPerformed(java.awt.event.ActionEvent evt) {  
185 cancelPressed(evt);  
186 }  
187 });  
188   
189 jButton3.setText("Decode");  
190 jButton3.addActionListener(  
191 new java.awt.event.ActionListener() {  
192 public void actionPerformed(java.awt.event.ActionEvent evt) {  
193 decodePressed(evt);  
194 }  
195 });  
196   
197 javax.swing.GroupLayout layout = new javax.swing.GroupLayout(getContentPane());  
198 getContentPane().setLayout(layout);  
199 layout.setHorizontalGroup(  
200 layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)  
201 .addGroup(layout.createSequentialGroup()  
202 .addContainerGap()  
203 .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)  
204 .addComponent(jPanel2, javax.swing.GroupLayout.Alignment.TRAILING, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, Short.MAX\_VALUE)  
205 .addComponent(jPanel1, javax.swing.GroupLayout.Alignment.TRAILING, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, Short.MAX\_VALUE)  
206 .addGroup(javax.swing.GroupLayout.Alignment.TRAILING, layout.createSequentialGroup()  
207 .addGap(0, 0, Short.MAX\_VALUE)  
208 .addComponent(jButton3)  
209 .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)  
210 .addComponent(jButton2)))  
211 .addContainerGap())  
212 );  
213 layout.setVerticalGroup(  
214 layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)  
215 .addGroup(layout.createSequentialGroup()  
216 .addContainerGap()  
217 .addComponent(jPanel1, javax.swing.GroupLayout.PREFERRED\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.PREFERRED\_SIZE)  
218 .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)  
219 .addComponent(jPanel2, javax.swing.GroupLayout.PREFERRED\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.PREFERRED\_SIZE)  
220 .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)  
221 .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)  
222 .addComponent(jButton2)  
223 .addComponent(jButton3))  
224 .addContainerGap(javax.swing.GroupLayout.DEFAULT\_SIZE, Short.MAX\_VALUE))  
225 );  
226   
227 pack();  
228 }// </editor-fold>//GEN-END:initComponents  
229   
230 private void cancelPressed(java.awt.event.ActionEvent evt) {//GEN-FIRST:event\_cancelPressed  
231 // TODO add your handling code here:  
232 this.setVisible(false);  
233 new MainFrame().setVisible(true);  
234 }//GEN-LAST:event\_cancelPressed  
235   
236 private void imageBrowsePressed(java.awt.event.ActionEvent evt) {//GEN-FIRST:event\_imageBrowsePressed  
237 // TODO add your handling code here:  
238 int returnVal = imageChooser.showOpenDialog(this);  
239 if(returnVal == JFileChooser.APPROVE\_OPTION)  
240 {  
241 File imageFile = imageChooser.getSelectedFile();  
242 if(evt.getSource() == browseButton1)  
243 {  
244 encodedTextField1.setText(imageFile.getAbsolutePath());  
245 shareFiles[0] = imageFile.getAbsolutePath();  
246 }  
247 else if(evt.getSource() == browseButton2)  
248 {  
249 encodedTextField2.setText(imageFile.getAbsolutePath());  
250 shareFiles[1] = imageFile.getAbsolutePath();  
251 }  
252 }  
253 }//GEN-LAST:event\_imageBrowsePressed  
254   
255 private void directoryBrowsePressed(java.awt.event.ActionEvent evt) {//GEN-FIRST:event\_directoryBrowsePressed  
256 // TODO add your handling code here:  
257 int returnVal = directoryChooser.showOpenDialog(this);  
258 if(returnVal == JFileChooser.APPROVE\_OPTION)  
259 {  
260 File dir = directoryChooser.getSelectedFile();  
261 if(evt.getSource() == browseButton3)  
262 {  
263 storageDirectoryTextField.setText(dir.getAbsolutePath());  
264 directoryForStorage = dir.getAbsolutePath();  
265 }  
266 }  
267 }//GEN-LAST:event\_directoryBrowsePressed  
268   
269 private void decodePressed(java.awt.event.ActionEvent evt) {//GEN-FIRST:event\_decodePressed  
270 // TODO add your handling code here:  
271 BufferedImage[] sharesEVCS = new BufferedImage[2];  
272 boolean fileFound = false;  
273   
274 for(int i = 0; i < 2; i++)  
275 {  
276 try  
277 {  
278 sharesEVCS[i] = ImageIO.read(new File(shareFiles[i]));  
279 fileFound = true;  
280 }  
281 catch(IOException e)  
282 {  
283 JOptionPane.showMessageDialog(null,   
284 ("Error reading file share" + (i + 1)),  
285 "ERROR", JOptionPane.ERROR\_MESSAGE);  
286 fileFound = false;  
287 }  
288 }  
289   
290 if(fileFound)  
291 {  
292 ExtendedVCS myEVCS = new ExtendedVCS(sharesEVCS);  
293 myEVCS.decryptImage();  
294   
295 if(storageDirectoryTextField.getText().equals(""))  
296 {  
297 //Get path to users desktop  
298 //BUG!!! Not working.  
299 directoryForStorage = "C:/Users/allisonholt/Desktop";  
300 //makeDir = false;  
301 }  
302   
303 String decodedFileName;  
304 if(stackedTextField.getText().equals(""))  
305 {  
306 //Get path to users desktop  
307 //BUG!!! Not working.  
308 decodedFileName = directoryForStorage + "/secretMsg.png";  
309 //makeDir = false;  
310 }  
311 else  
312 {  
313 decodedFileName = directoryForStorage + "/" + stackedTextField.getText() + ".png";  
314 }  
315   
316 try  
317 {  
318 BufferedImage decryptImage = new BufferedImage(myEVCS.getImgWidth(), myEVCS.getImgHeight(), BufferedImage.TYPE\_INT\_ARGB);  
319 decryptImage.setRGB(0, 0, myEVCS.getImgWidth(), myEVCS.getImgHeight(), myEVCS.getDecryptImgPixels(), 0, myEVCS.getImgWidth());  
320   
321 File tempOutput = new File(decodedFileName);  
322 ImageIO.write(decryptImage, "png", tempOutput);  
323   
324 new MainFrame().setVisible(true);  
325 this.setVisible(false);  
326 JOptionPane.showMessageDialog(null, "Your decrypted image has been created.",  
327 "SUCCESS", JOptionPane.PLAIN\_MESSAGE);  
328 }  
329 catch(IOException e)  
330 {  
331 JOptionPane.showMessageDialog(null, "Error decrypting your secret message",  
332 "ERROR", JOptionPane.ERROR\_MESSAGE);  
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 || file.getAbsolutePath().endsWith(".jpg");  
23 }  
24   
25 @Override  
26 public String getDescription()  
27 {  
28 return "Image files (\*.png, \*.jpeg, \*.jpg)";  
29 }  
30   
31 }  
32

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 private int[][] secretSharesRGB;  
 28   
 29 private int numSharesToDecrypt;  
 30 private BufferedImage[] sharesToDecrypt;  
 31 private int[] secretMsgPixels;  
 32   
 33   
 34 //For encryption purposes  
 35 ExtendedVCS(BufferedImage secretMsgIn, BufferedImage[] innocentSharesIn)  
 36 {  
 37 k = 2;  
 38 n = 2;  
 39 secretMsg = secretMsgIn;  
 40 imgWidth = secretMsg.getWidth();  
 41 imgHeight = secretMsg.getHeight();  
 42 innocentShares = innocentSharesIn;  
 43 }  
 44   
 45 //For decryption purposes  
 46 ExtendedVCS(BufferedImage[] shareImgs)  
 47 {  
 48 numSharesToDecrypt = 2;  
 49 sharesToDecrypt = shareImgs;  
 50 imgWidth = shareImgs[0].getWidth();  
 51 imgHeight = shareImgs[0].getHeight();  
 52 }  
 53   
 54 int getImgWidth()  
 55 {  
 56 return imgWidth;  
 57 }  
 58   
 59 int getImgHeight()  
 60 {  
 61 return imgHeight;  
 62 }  
 63   
 64 int[][] getRGBPixelsForShares()  
 65 {  
 66 return encryptedShareRGB;  
 67 }  
 68   
 69 int[] getDecryptImgPixels()  
 70 {  
 71 return secretMsgPixels;  
 72 }  
 73   
 74 void encryptImage()  
 75 {  
 76 int[] secretRGB = new int[imgWidth \* imgHeight];  
 77 int[][] shareOrigRGB = new int[2][imgWidth \* imgHeight];  
 78 secretMsg.getRGB(0, 0, imgWidth, imgHeight, secretRGB, 0, imgWidth);  
 79 innocentShares[0].getRGB(0, 0, imgWidth, imgHeight, shareOrigRGB[0], 0, imgWidth);  
 80 innocentShares[1].getRGB(0, 0, imgWidth, imgHeight, shareOrigRGB[1], 0, imgWidth);  
 81 createPixelsOfShares(secretRGB, shareOrigRGB);  
 82 }  
 83   
 84 /\*\*  
 85 \*   
 86 \* @param secretImgRGB The RGB values of the secret image  
 87 \* @param shareOriginalRGB The RGB values of the innocent images  
 88 \*/  
 89 void createPixelsOfShares(int[] secretImgRGB, int[][] shareOriginalRGB)  
 90 {  
 91 //Used to store the embedded RGB values  
 92 encryptedShareRGB = new int[2][imgWidth \* imgHeight];  
 93   
 94 //Used to bring the secret image up using a size invarint-ish technique  
 95 secretSharesRGB = new int[2][imgWidth \* imgHeight];  
 96   
 97 for(int i = 0; i < secretImgRGB.length; i++)  
 98 {  
 99 int redVal = (secretImgRGB[i] & 0x00ff0000) >> 16;  
100 int greenVal = (secretImgRGB[i] & 0x0000ff00) >> 8;  
101 int blueVal = (secretImgRGB[i] & 0x000000ff);  
102 Pixel orig = new Pixel(redVal, greenVal, blueVal);  
103   
104 redVal = (shareOriginalRGB[0][i] & 0x00ff0000) >> 16;  
105 greenVal = (shareOriginalRGB[0][i] & 0x0000ff00) >> 8;  
106 blueVal = (shareOriginalRGB[0][i] & 0x000000ff);  
107 Pixel innocent0 = new Pixel(redVal, greenVal, blueVal);  
108   
109 redVal = (shareOriginalRGB[1][i] & 0x00ff0000) >> 16;  
110 greenVal = (shareOriginalRGB[1][i] & 0x0000ff00) >> 8;  
111 blueVal = (shareOriginalRGB[1][i] & 0x000000ff);  
112 Pixel innocent1 = new Pixel(redVal, greenVal, blueVal);  
113   
114 Random randomGen = new Random();  
115 int maxGrayCon = orig.getConcentration();  
116 int grayCon1 = randomGen.nextInt(maxGrayCon + 1);  
117 int grayCon2 = maxGrayCon - grayCon1;  
118   
119 Color secretGray1 = new Color(grayCon1, grayCon1, grayCon1);  
120 secretSharesRGB[0][i] = secretGray1.getRGB();  
121   
122 Color secretGray2 = new Color(grayCon2, grayCon2, grayCon2);  
123 secretSharesRGB[1][i] = secretGray2.getRGB();  
124   
125 int innocent1Con = innocent0.getConcentration();  
126 int embedded1Con = (innocent1Con + grayCon1) / 2;  
127 if(embedded1Con < 0)  
128 embedded1Con = 0;  
129 Color embedded1 = new Color(embedded1Con, embedded1Con, embedded1Con);  
130 encryptedShareRGB[0][i] = embedded1.getRGB();  
131   
132 int innocent2Con = innocent1.getConcentration();  
133 int embedded2Con = (innocent2Con + grayCon2) / 2;  
134 if(embedded2Con < 0)  
135 embedded2Con = 0;  
136 Color embedded2 = new Color(embedded2Con, embedded2Con, embedded2Con);  
137 encryptedShareRGB[1][i] = embedded2.getRGB();  
138 }  
139 }  
140   
141 void decryptImage()  
142 {  
143 //Make a 2d array of pixel arrays  
144 int[][] embeddedPixels = new int[numSharesToDecrypt][imgWidth \* imgHeight];  
145 secretMsgPixels = new int[imgWidth \* imgHeight];  
146   
147 //getRGB pixels of BufferedImages  
148 for(int i = 0; i < numSharesToDecrypt; i++)  
149 {  
150 sharesToDecrypt[i].getRGB(0, 0, imgWidth, imgHeight, embeddedPixels[i], 0, imgWidth);  
151 }  
152   
153 //Logical OR pixel with all three share values  
154 int numOfPixels = embeddedPixels[0].length;  
155 for(int i = 0; i < numOfPixels; i++)  
156 {  
157 int redVal = (embeddedPixels[0][i] & 0x00ff0000) >> 16;  
158 int greenVal = (embeddedPixels[0][i] & 0x0000ff00) >> 8;  
159 int blueVal = (embeddedPixels[0][i] & 0x000000ff);  
160 Pixel embedded1 = new Pixel(redVal, greenVal, blueVal);  
161 int share1Con = embedded1.getConcentration();  
162   
163 redVal = (embeddedPixels[1][i] & 0x00ff0000) >> 16;  
164 greenVal = (embeddedPixels[1][i] & 0x0000ff00) >> 8;  
165 blueVal = (embeddedPixels[1][i] & 0x000000ff);  
166 Pixel embedded2 = new Pixel(redVal, greenVal, blueVal);  
167 int share2Con = embedded2.getConcentration();  
168   
169 int totalCon = 255 - (share1Con + share2Con);  
170 if(totalCon < 0)  
171 totalCon = 0;  
172   
173 Color decryptedColor = new Color(totalCon, totalCon, totalCon);  
174 secretMsgPixels[i] = decryptedColor.getRGB();  
175 }  
176 }  
177   
178 }  
179

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 int getConcentration()  
29 {  
30 return redVal;  
31 }  
32   
33 }  
34