**THEORY**

The GUI of the program was developed using java Swing and Java AWT. In order to facilitate ease of use, the project has been done on NetBeans Integrated Development Environment.

**What is NetBeans?**

NetBeans is a Java-based Integrated Development Environment (IDE). The IDE is designed to limit coding errors and facilitate error correction with tools such as the NetBeans FindBugs to locate and fix common Java coding problems and Debugger to manage complex code with field watches, breakpoints and execution monitoring. Although the NetBeans IDE is designed specifically for Java developers, it also supports C/C++,[PHP](https://whatis.techtarget.com/definition/PHP-Hypertext-Preprocessor), [Groovy](https://www.theserverside.com/definition/Groovy), and HTML5 in addition to[Java](https://www.theserverside.com/definition/Java), JavaScript and JavaFX.

Tools and capabilities of the NetBeans IDE include a feature-rich text editor with refactoring tools and code templates, high level and granular views of applications, a drag and drop GUI design, and versioning using out-of-the-box integration with tools such as [Git.](https://whatis.techtarget.com/fileformat/GIT-BioWare-Aurora-Engine-Area-BioWare-Corp) The NetBeans IDE can run on any operating system that supports a compatible JVM including [Linux](https://searchdatacenter.techtarget.com/definition/Linux-operating-system), Windows and OS X.

The underlying NetBeans platform supports creation of new applications and further development of existing applications using modular software components. As an application running on the NetBeans Platform, the NetBeans IDE itself is extensible and can be extended to support new languages.

**What was used?**

The GUI of the program is based on java swing (Jframe), components of which are primarily generated by the NetBeans IDE itself. Swing is a part of JFC (Java Foundation Classes). Building Graphical User Interface in Java requires the use of Swings. Swing Framework contain a large set of components which allow high level of customization and provide rich functionalities, and is used to create window-based applications.  
Java swing components are lightweight, platform independent, provide powerful components like tables, scroll panels, buttons, list, color chooser, etc.

During the writing of this program, various Swing features and classes were used. They are listed and explained below:

**List of java classes used:**

java.awt.Image

java.io.File

javax.swing.ImageIcon

javax.swing.JFileChooser

javax.swing.JOptionPane

javax.swing.filechooser.FileNameExtensionFilter

1. **java.awt.Image**: The java.awt.image package contains classes and interfaces for manipulating images. In Java 2D, the image-processing model has been extended (and simplified) to accommodate image data that is stored and manipulated in memory. The key pieces of this new image-processing model are the BufferedImage class, which represents an image in memory, and the BufferedImageOp interface, which represents an image-processing operation. Every BufferedImage contains a Raster object that hold the pixels of the image and a ColorModel object that can interpret those pixel values as Color objects. A Raster object, in turn, contains a DataBuffer that holds the raw image data and a SampleModel object that knows how to extract pixel values from that raw data.

2. **java.io.File**: The File class is Java’s representation of a file or directory path name. Because file and directory names have different formats on different platforms, a simple string is not adequate to name them. The File class contains several methods for working with the path name, deleting and renaming files, creating new directories, listing the contents of a directory, and determining several common attributes of files and directories.

3. **javax.swing.JfileChooser**: JFileChooser is a part of java Swing package. The java Swing package is part of JavaTM Foundation Classes(JFC) . JFC contains many features that help in building graphical user interface in java . Java Swing provides components such as buttons, panels, dialogs, etc . JFileChooser is a easy and an effective way to prompt the user to choose a file or a directory .

4. **javax.swing.JoptionPane**: Message dialogs provide information to the user. Message dialogs are created with the JOptionPane.showMessageDialog() method.

We call the static showMessageDialog() method of the JOptionPane class to create a message dialog. We provide the dialog’s parent, message text, title and a message type. The JOptionPane class is used to provide standard dialog boxes such as message dialog box, confirm dialog box and input dialog box. These dialog boxes are used to display information or get input from the user. The JOptionPane class inherits JComponent class.

5. **javax.swing.filechooser.FileNameExtensionFilter**: This is used to limit the type of data that will be input into the filechooser. Suppose you need to use take only a text file as an input the this can b achieved by using the fuctions defined in above class.

**Working of netbeans:**

In netbeans, every swing component given its own method. The component is first defined in the design area where all the dimensions and location of the component is decided. Then, the code for this particular component is defined by netbeans itself. Further, it is allotted an action performed method in which we can define what action will be performed by the swing entity. Hence, instead of using action listener, the button is linked to its own method which is executed when the button is clicked in the GUI. Other elements of the GUI can also be called in the component. E.g, a textfield can be called in button in order to execute some function related to the field.

**Some methods used:**

From these classes, a number of methods have been used in order to build the GUI for the steganography project. They are listed below:

1. *jTextFieldname*.getText(): This function is used to get the text stored in the given jtextfield.

2. *jTextFieldname.*setText(output): This function is used to set the given text onto the textfield.

3. *FileNameExtensionFilter filter = new FileNameExtensionFilter(types)*: This is used to define a filter which will limit the types of data that can be input by the user.

4. *file.getAbsolutePath()*: This is used to get the absolute location of the file in memory

5.*ImageIcon imIco = new ImageIcon(getselectedImage):* This is a constructor which constructs an object which can be used to store buffered images.

These are few of the classes and functions which were used while writing the GUI for the given code. The working of the steganography class is further defined in the functions section.

**FUNCTIONALITY**

For Steganography process, we have used the LSB method to encode the text on the image and while decoding the same logic is used to extract the text from the image. The details of LSB method of steganography is described below.

**LSB Steganography**

The simplest form of digital steganography (and probably the most common) is the

Least Significant Bit (LSB) method, where the binary representation of the data that's

to be hidden is written into the LSB of the bytes of the carrier. The overall change to

the image is so minor that it can't be seen by the human eye.

Figure 1 shows the first stage of the process, when the image data is accessed as a

series of bytes. Depending on the image format, a pixel may be represented by one or

more bytes. In my examples, I'll be using 24-bit PNG images, which use a byte each

for the red (R), green (G), and blue (B) channels.



Figure 1. Accessing the Bits of a PNG image.

The next stage is to read in the text file, and access its bits, as shown in Figure 2.



Figure 2. Accessing the Bits of a Text File.

Now its time to insert the bits of the text file into the image. The LSB approach only

modifies the least significant bit of each image byte, as illustrated by Figure 3.

**

Figure 3. Inserting the Text Bits into the Image.

Extracting the text from the image at a later time involves copying the LSBs of the

modified image's bytes, and recombining them into bytes in a text file, as in Figure 4.



Figure 4. Extracting the Text from the Modified Image.

The LSB approach is the basis of all of my steganography classes.

**2. Implementing LSB Steganography**

The Steganography class implements the LSB method for hiding text inside a PNG

image as explained in the diagrams above, but with one addition. The length of the

text in binary form is calculated beforehand, and hidden in the image before the text.

In other words, the steganographic information (the *stego*) has two parts: the size of

the binary message, followed by the message itself.

The stego is spread over the image by modifying each byte's LSB. This means that 1

byte of stego data requires the modification of 8 bytes of the image (i.e. 1 stego data

bit is stored in 1 image byte).

The size information is utilized when the text is extracted from the image, so the

extraction process knows when to stop. The size data is a Java integer, which employs

four bytes, and so needs 32 (4\*8) bytes at the start of the image.

**PROGRAM CODE**

import java.awt.Image;

import java.io.File;

import javax.swing.ImageIcon;

import javax.swing.JFileChooser;

import javax.swing.JOptionPane;

import javax.swing.filechooser.FileNameExtensionFilter;

/\*

<applet code="Login.class" width=900 height=1000>

</applet>

\*/

/\*\*

\*

\* @author Abhijit & Saurav

\*/

public class StegJFrame extends javax.swing.JFrame {

/\*\*

\* Creates new form StegJFrame

\*/

public StegJFrame() {

initComponents();

}

/\*\*

\* This method is called from within the constructor to initialize the form.

\* WARNING: Do NOT modify this code. The content of this method is always

\* regenerated by the Form Editor.

\*/

@SuppressWarnings("unchecked")

// <editor-fold defaultstate="collapsed" desc="Generated Code">

private void initComponents() {

EncodeText = new javax.swing.JTextField();

DecodedText = new javax.swing.JTextField();

ImgDisplay = new javax.swing.JLabel();

EncodeButton = new javax.swing.JButton();

jLabel1 = new javax.swing.JLabel();

jLabel2 = new javax.swing.JLabel();

DecodeButton = new javax.swing.JButton();

ClearEncodeText = new javax.swing.JButton();

EncodeImgLoc = new javax.swing.JTextField();

DecodeImgLoc = new javax.swing.JTextField();

ChooseEncodeLoc = new javax.swing.JButton();

ChooseDecodeLoc = new javax.swing.JButton();

jLabel3 = new javax.swing.JLabel();

jLabel4 = new javax.swing.JLabel();

ClearDecodedText = new javax.swing.JButton();

HelpEncode = new javax.swing.JButton();

HelpDecode = new javax.swing.JButton();

jScrollPane1 = new javax.swing.JScrollPane();

Current = new javax.swing.JTextArea();

Exit = new javax.swing.JButton();

jLabel5 = new javax.swing.JLabel();

jLabel6 = new javax.swing.JLabel();

ClearAll = new javax.swing.JButton();

setDefaultCloseOperation(javax.swing.WindowConstants.EXIT\_ON\_CLOSE);

EncodeText.setToolTipText("");

EncodeText.addActionListener(new java.awt.event.ActionListener() {

public void actionPerformed(java.awt.event.ActionEvent evt) {

EncodeTextActionPerformed(evt);

}

});

DecodedText.setEditable(false);

DecodedText.setText(" ");

DecodedText.setDisabledTextColor(new java.awt.Color(0, 0, 0));

DecodedText.setDoubleBuffered(true);

DecodedText.addActionListener(new java.awt.event.ActionListener() {

public void actionPerformed(java.awt.event.ActionEvent evt) {

DecodedTextActionPerformed(evt);

}

});

EncodeButton.setBackground(new java.awt.Color(220, 20, 60));

EncodeButton.setText("Encode");

EncodeButton.addActionListener(new java.awt.event.ActionListener() {

public void actionPerformed(java.awt.event.ActionEvent evt) {

EncodeButtonActionPerformed(evt);

}

});

jLabel1.setText("Enter the text to be inserted:");

jLabel2.setText("The extracted text is:");

DecodeButton.setBackground(new java.awt.Color(34, 139, 34));

DecodeButton.setText("Decode");

DecodeButton.addActionListener(new java.awt.event.ActionListener() {

public void actionPerformed(java.awt.event.ActionEvent evt) {

DecodeButtonActionPerformed(evt);

}

});

ClearEncodeText.setBackground(new java.awt.Color(0, 255, 255));

ClearEncodeText.setText("Clear text");

ClearEncodeText.addActionListener(new java.awt.event.ActionListener() {

public void actionPerformed(java.awt.event.ActionEvent evt) {

ClearEncodeTextActionPerformed(evt);

}

});

EncodeImgLoc.addActionListener(new java.awt.event.ActionListener() {

public void actionPerformed(java.awt.event.ActionEvent evt) {

EncodeImgLocActionPerformed(evt);

}

});

ChooseEncodeLoc.setBackground(new java.awt.Color(0, 255, 255));

ChooseEncodeLoc.setText("Choose");

ChooseEncodeLoc.addActionListener(new java.awt.event.ActionListener() {

public void actionPerformed(java.awt.event.ActionEvent evt) {

ChooseEncodeLocActionPerformed(evt);

}

});

ChooseDecodeLoc.setBackground(new java.awt.Color(0, 255, 255));

ChooseDecodeLoc.setText("Choose");

ChooseDecodeLoc.addActionListener(new java.awt.event.ActionListener() {

public void actionPerformed(java.awt.event.ActionEvent evt) {

ChooseDecodeLocActionPerformed(evt);

}

});

jLabel3.setFont(new java.awt.Font("Arial", 1, 18)); // NOI18N

jLabel3.setText("ENCODE");

jLabel4.setFont(new java.awt.Font("Arial", 1, 18)); // NOI18N

jLabel4.setText("DECODE");

ClearDecodedText.setBackground(new java.awt.Color(0, 255, 255));

ClearDecodedText.setText("Clear text");

ClearDecodedText.addActionListener(new java.awt.event.ActionListener() {

public void actionPerformed(java.awt.event.ActionEvent evt) {

ClearDecodedTextActionPerformed(evt);

}

});

HelpEncode.setBackground(new java.awt.Color(255, 165, 0));

HelpEncode.setText("Help");

HelpEncode.addActionListener(new java.awt.event.ActionListener() {

public void actionPerformed(java.awt.event.ActionEvent evt) {

HelpEncodeActionPerformed(evt);

}

});

HelpDecode.setBackground(new java.awt.Color(255, 165, 0));

HelpDecode.setText("Help");

HelpDecode.addActionListener(new java.awt.event.ActionListener() {

public void actionPerformed(java.awt.event.ActionEvent evt) {

HelpDecodeActionPerformed(evt);

}

});

Current.setColumns(20);

Current.setRows(5);

Current.setCursor(new java.awt.Cursor(java.awt.Cursor.TEXT\_CURSOR));

Current.setDisabledTextColor(new java.awt.Color(0, 0, 0));

Current.setDoubleBuffered(true);

Current.setEnabled(false);

jScrollPane1.setViewportView(Current);

Exit.setText("Exit");

Exit.addActionListener(new java.awt.event.ActionListener() {

public void actionPerformed(java.awt.event.ActionEvent evt) {

ExitActionPerformed(evt);

}

});

jLabel5.setText("Selected Image:");

jLabel6.setText("Debugging window:");

ClearAll.setText("Clear all");

ClearAll.addActionListener(new java.awt.event.ActionListener() {

public void actionPerformed(java.awt.event.ActionEvent evt) {

ClearAllActionPerformed(evt);

}

});

javax.swing.GroupLayout layout = new javax.swing.GroupLayout(getContentPane());

getContentPane().setLayout(layout);

layout.setHorizontalGroup(

layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

.addGroup(layout.createSequentialGroup()

.addContainerGap()

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

.addGroup(javax.swing.GroupLayout.Alignment.TRAILING, layout.createSequentialGroup()

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

.addComponent(jLabel1)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRAILING)

.addGroup(layout.createSequentialGroup()

.addComponent(EncodeImgLoc, javax.swing.GroupLayout.PREFERRED\_SIZE, 388, javax.swing.GroupLayout.PREFERRED\_SIZE)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)

.addComponent(ChooseEncodeLoc))

.addGroup(layout.createSequentialGroup()

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRAILING)

.addGroup(javax.swing.GroupLayout.Alignment.LEADING, layout.createSequentialGroup()

.addComponent(EncodeButton, javax.swing.GroupLayout.PREFERRED\_SIZE, 73, javax.swing.GroupLayout.PREFERRED\_SIZE)

.addGap(18, 18, 18)

.addComponent(HelpEncode))

.addComponent(EncodeText, javax.swing.GroupLayout.PREFERRED\_SIZE, 352, javax.swing.GroupLayout.PREFERRED\_SIZE))

.addGap(18, 18, 18)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING, false)

.addComponent(ClearEncodeText, javax.swing.GroupLayout.DEFAULT\_SIZE, 100, Short.MAX\_VALUE)

.addComponent(ClearAll, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, Short.MAX\_VALUE)))))

.addGap(31, 31, 31))

.addGroup(layout.createSequentialGroup()

.addComponent(jLabel3)

.addGap(353, 353, 353)))

.addGroup(layout.createSequentialGroup()

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

.addGroup(layout.createSequentialGroup()

.addGap(10, 10, 10)

.addComponent(jLabel6))

.addComponent(jScrollPane1, javax.swing.GroupLayout.PREFERRED\_SIZE, 473, javax.swing.GroupLayout.PREFERRED\_SIZE))

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)))

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

.addComponent(ImgDisplay, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, Short.MAX\_VALUE)

.addGroup(layout.createSequentialGroup()

.addGap(6, 6, 6)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

.addComponent(jLabel4)

.addComponent(jLabel2)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRAILING, false)

.addGroup(javax.swing.GroupLayout.Alignment.LEADING, layout.createSequentialGroup()

.addComponent(DecodeButton)

.addGap(18, 18, 18)

.addComponent(HelpDecode)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED, javax.swing.GroupLayout.DEFAULT\_SIZE, Short.MAX\_VALUE)

.addComponent(Exit, javax.swing.GroupLayout.PREFERRED\_SIZE, 140, javax.swing.GroupLayout.PREFERRED\_SIZE))

.addGroup(javax.swing.GroupLayout.Alignment.LEADING, layout.createSequentialGroup()

.addComponent(DecodedText, javax.swing.GroupLayout.PREFERRED\_SIZE, 352, javax.swing.GroupLayout.PREFERRED\_SIZE)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED, javax.swing.GroupLayout.DEFAULT\_SIZE, Short.MAX\_VALUE)

.addComponent(ClearDecodedText, javax.swing.GroupLayout.PREFERRED\_SIZE, 100, javax.swing.GroupLayout.PREFERRED\_SIZE))

.addGroup(javax.swing.GroupLayout.Alignment.LEADING, layout.createSequentialGroup()

.addComponent(DecodeImgLoc, javax.swing.GroupLayout.PREFERRED\_SIZE, 390, javax.swing.GroupLayout.PREFERRED\_SIZE)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)

.addComponent(ChooseDecodeLoc)))

.addComponent(jLabel5))

.addGap(0, 40, Short.MAX\_VALUE)))

.addContainerGap())

);

layout.setVerticalGroup(

layout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRAILING)

.addGroup(layout.createSequentialGroup()

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

.addGroup(layout.createSequentialGroup()

.addGap(40, 40, 40)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)

.addComponent(EncodeImgLoc, javax.swing.GroupLayout.PREFERRED\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.PREFERRED\_SIZE)

.addComponent(ChooseEncodeLoc)

.addComponent(DecodeImgLoc, javax.swing.GroupLayout.PREFERRED\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.PREFERRED\_SIZE)

.addComponent(ChooseDecodeLoc))

.addGap(22, 22, 22)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)

.addComponent(jLabel1)

.addComponent(jLabel2))

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)

.addComponent(EncodeText, javax.swing.GroupLayout.PREFERRED\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.PREFERRED\_SIZE)

.addComponent(ClearEncodeText)

.addComponent(ClearDecodedText)

.addComponent(DecodedText, javax.swing.GroupLayout.PREFERRED\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.PREFERRED\_SIZE)))

.addGroup(layout.createSequentialGroup()

.addContainerGap()

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)

.addComponent(jLabel4)

.addComponent(jLabel3))))

.addGap(18, 18, 18)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

.addComponent(Exit, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, Short.MAX\_VALUE)

.addGroup(layout.createSequentialGroup()

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)

.addComponent(EncodeButton, javax.swing.GroupLayout.PREFERRED\_SIZE, 51, javax.swing.GroupLayout.PREFERRED\_SIZE)

.addComponent(HelpEncode, javax.swing.GroupLayout.PREFERRED\_SIZE, 52, javax.swing.GroupLayout.PREFERRED\_SIZE))

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)

.addComponent(DecodeButton, javax.swing.GroupLayout.PREFERRED\_SIZE, 52, javax.swing.GroupLayout.PREFERRED\_SIZE)

.addComponent(HelpDecode, javax.swing.GroupLayout.PREFERRED\_SIZE, 52, javax.swing.GroupLayout.PREFERRED\_SIZE)

.addComponent(ClearAll, javax.swing.GroupLayout.PREFERRED\_SIZE, 52, javax.swing.GroupLayout.PREFERRED\_SIZE)))

.addGap(0, 1, Short.MAX\_VALUE)))

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

.addComponent(jLabel6)

.addComponent(jLabel5))

.addGap(4, 4, 4)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

.addComponent(ImgDisplay, javax.swing.GroupLayout.DEFAULT\_SIZE, 321, Short.MAX\_VALUE)

.addComponent(jScrollPane1, javax.swing.GroupLayout.DEFAULT\_SIZE, 321, Short.MAX\_VALUE))

.addContainerGap())

);

pack();

}// </editor-fold>

private void DecodedTextActionPerformed(java.awt.event.ActionEvent evt) {

// TODO add your handling code here:

}

private void EncodeTextActionPerformed(java.awt.event.ActionEvent evt) {

}

private void EncodeButtonActionPerformed(java.awt.event.ActionEvent evt) {

//Getting file location

String FileLoc=EncodeImgLoc.getText();

//Debugging window

if(!FileLoc.isEmpty())

{

Current.append("The chosen file is:"+FileLoc+"\n");

Current.append("Encoding message onto the photo.....\n");

//creating object of class Steganography

Steganography stego = new Steganography();

//Getting message

String Message = EncodeText.getText();

//Encoding

stego.encode(FileLoc, Message);

//Successful encoding

Current.append("The text has been successfully encoded onto the selected photo\n");

}

else

{

Current.append("File not chosen\n");

}

}

private void DecodeButtonActionPerformed(java.awt.event.ActionEvent evt) {

//Getting file location

String filename=DecodeImgLoc.getText();

//Debugging window

if(!filename.isEmpty())

{

Current.append("The chosen file is:"+filename+"\n");

//creating object of class Steganography

Steganography steg = new Steganography();

//Getting message

String output= steg.decode(filename);

//printing message

DecodedText.setText(output);

//Successful decoding

Current.append("The decoded data has been successfully displayed\n");

}

else

{

Current.append("File not chosen\n");

}

}

private void EncodeImgLocActionPerformed(java.awt.event.ActionEvent evt) {

// TODO add your handling code here:

}

private void ChooseEncodeLocActionPerformed(java.awt.event.ActionEvent evt) {

Current.setText("");

//Creating object of type filechooser

JFileChooser fChooser = new JFileChooser();

//Limit type of file name extensions supported.

FileNameExtensionFilter filter = new FileNameExtensionFilter("4 Extensions Supported", "jpg", "png", "jpeg", "gif");

//Applying filter

fChooser.setFileFilter(filter);

int selected = fChooser.showOpenDialog(null);

//check if open button has been clicked.

if (selected == JFileChooser.APPROVE\_OPTION) {

File file = fChooser.getSelectedFile();

//Getting Path of the selected image.

String getselectedImage = file.getAbsolutePath();

//Creating object to store image in type imageicon

ImageIcon imIco = new ImageIcon(getselectedImage);

//make image fit on jlabel.

Image imFit = imIco.getImage();

//modifying object on the basis of the jLabel

Image imgFit = imFit.getScaledInstance(ImgDisplay.getWidth(), ImgDisplay.getHeight(), Image.SCALE\_SMOOTH);

//COnverting image back to its imageicon form in order to print it onto the jLabel

ImgDisplay.setIcon(new ImageIcon(imgFit));

//Getting file location and printing it

String filename=file.getAbsolutePath();

EncodeImgLoc.setText(filename);

}

}

private void ClearEncodeTextActionPerformed(java.awt.event.ActionEvent evt) {

EncodeText.setText("");

Current.append("The text field has been cleared\n");

}

private void ClearDecodedTextActionPerformed(java.awt.event.ActionEvent evt) {

DecodedText.setText("");

Current.append("The text field has been cleared\n");

}

private void ChooseDecodeLocActionPerformed(java.awt.event.ActionEvent evt) {

Current.setText("");

//Creating object of type filechooser

JFileChooser fileChooser = new JFileChooser();

//Limit type of file name extensions supported.

FileNameExtensionFilter filter = new FileNameExtensionFilter("4 Extensions Supported", "jpg", "png", "jpeg", "gif");

//Applying filter

fileChooser.setFileFilter(filter);

int selected = fileChooser.showOpenDialog(null);

//check if button has been clicked.

if (selected == JFileChooser.APPROVE\_OPTION) {

File file = fileChooser.getSelectedFile();

//Getting Path of the selected image

String getselectedImage = file.getAbsolutePath();

//Creating object to store image in type imageicon

ImageIcon imIco = new ImageIcon(getselectedImage);

//make image fit on jlabel.

Image imFit = imIco.getImage();

//modifying object on the basis of the jLabel

Image imgFit = imFit.getScaledInstance(ImgDisplay.getWidth(), ImgDisplay.getHeight(), Image.SCALE\_SMOOTH);

//COnverting image back to its imageicon form in order to print it onto the jLabel

ImgDisplay.setIcon(new ImageIcon(imgFit));

//Getting file location and printing it

String filename=file.getAbsolutePath();

DecodeImgLoc.setText(filename);

}

}

private void HelpEncodeActionPerformed(java.awt.event.ActionEvent evt) {

JOptionPane.showMessageDialog(null, "1. Select the image using choose button\n2. Enter the text in the given field\n3. Click on encode\nYour trext has now been encoded onto that image");

}

private void HelpDecodeActionPerformed(java.awt.event.ActionEvent evt) {

JOptionPane.showMessageDialog(null, "1. Select the image using choose button\n2. Click on Decode\nThe decoded text will now be shown");

}

private void ExitActionPerformed(java.awt.event.ActionEvent evt) {

System.exit(0);

}

private void ClearAllActionPerformed(java.awt.event.ActionEvent evt) {

//Clearing every field

EncodeText.setText("");

DecodedText.setText("");

EncodeImgLoc.setText("");

DecodeImgLoc.setText("");

Current.setText("");

ImgDisplay.setIcon(null);

}

/\*\*

\* @param args the command line arguments

\*/

public static void main(String args[]) {

/\* Set the Nimbus look and feel \*/

//<editor-fold defaultstate="collapsed" desc=" Look and feel setting code (optional) ">

/\* If Nimbus (introduced in Java SE 6) is not available, stay with the default look and feel.

\* For details see http://download.oracle.com/javase/tutorial/uiswing/lookandfeel/plaf.html

\*/

try {

for (javax.swing.UIManager.LookAndFeelInfo info : javax.swing.UIManager.getInstalledLookAndFeels()) {

if ("Nimbus".equals(info.getName())) {

javax.swing.UIManager.setLookAndFeel(info.getClassName());

break;

}

}

} catch (ClassNotFoundException ex) {

java.util.logging.Logger.getLogger(StegJFrame.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);

} catch (InstantiationException ex) {

java.util.logging.Logger.getLogger(StegJFrame.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);

} catch (IllegalAccessException ex) {

java.util.logging.Logger.getLogger(StegJFrame.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);

} catch (javax.swing.UnsupportedLookAndFeelException ex) {

java.util.logging.Logger.getLogger(StegJFrame.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);

}

//</editor-fold>

/\* Create and display the form \*/

java.awt.EventQueue.invokeLater(new Runnable() {

public void run() {

new StegJFrame().setVisible(true);

}

});

}

// Variables declaration - do not modify

private javax.swing.JButton ChooseDecodeLoc;

private javax.swing.JButton ChooseEncodeLoc;

private javax.swing.JButton ClearAll;

private javax.swing.JButton ClearDecodedText;

private javax.swing.JButton ClearEncodeText;

private javax.swing.JTextArea Current;

private javax.swing.JButton DecodeButton;

private javax.swing.JTextField DecodeImgLoc;

private javax.swing.JTextField DecodedText;

private javax.swing.JButton EncodeButton;

private javax.swing.JTextField EncodeImgLoc;

private javax.swing.JTextField EncodeText;

private javax.swing.JButton Exit;

private javax.swing.JButton HelpDecode;

private javax.swing.JButton HelpEncode;

private javax.swing.JLabel ImgDisplay;

private javax.swing.JLabel jLabel1;

private javax.swing.JLabel jLabel2;

private javax.swing.JLabel jLabel3;

private javax.swing.JLabel jLabel4;

private javax.swing.JLabel jLabel5;

private javax.swing.JLabel jLabel6;

private javax.swing.JScrollPane jScrollPane1;

// End of variables declaration

}

import java.awt.Graphics2D;

import java.awt.image.BufferedImage;

import java.awt.image.DataBufferByte;

import java.awt.image.WritableRaster;

import java.io.File;

import javax.imageio.ImageIO;

import javax.swing.JOptionPane;

public class Steganography

{

public Steganography()

{

}

public boolean encode(String file\_path, String message)

{

BufferedImage image = user\_space(getImage(file\_path));

image = add\_text(image,message);

return(setImage(image,new File(file\_path),"png"));

}

public String decode(String file\_path)

{

byte[] decode;

try

{

BufferedImage image = user\_space(getImage(file\_path));

decode = decode\_text(get\_byte\_data(image));

return(new String(decode));

}

catch(Exception e)

{

JOptionPane.showMessageDialog(null, "There is no hidden message in this image!","Error",JOptionPane.ERROR\_MESSAGE);

return "";

}

}

private BufferedImage getImage(String f)//BufferedImage method used to get Image into the buffered file

{

BufferedImage image = null;//temporary varaible initialised

File file = new File(f);//New file declared with file variable

try

{

image = ImageIO.read(file);//image variable loaded with the image file

}

catch(Exception ex)

{

JOptionPane.showMessageDialog(null, "Image could not be read!","Error",JOptionPane.ERROR\_MESSAGE);

}

return image;//imsge returned

}

private boolean setImage(BufferedImage image, File file, String extension)//SetImage method used

{

try

{

ImageIO.write(image,extension,file);//ImageIO file used to write image with its extension and file data

return true;

}

catch(Exception e)//catch statement

{

JOptionPane.showMessageDialog(null, "File could not be saved!","Error",JOptionPane.ERROR\_MESSAGE);

return false;

}

}

private BufferedImage add\_text(BufferedImage image, String text)

{

//convert all items to byte arrays: image, message, message length

byte img[] = get\_byte\_data(image);

byte msg[] = text.getBytes();

byte len[] = bit\_conversion(msg.length);

try

{

encode\_text(img, len, 0); //0 first positioning

encode\_text(img, msg, 32); //4 bytes of space for length: 4bytes\*8bit = 32 bits

}

catch(Exception e)

{

JOptionPane.showMessageDialog(null, "Target File cannot hold message!", "Error",JOptionPane.ERROR\_MESSAGE);

}

return image;

}

private BufferedImage user\_space(BufferedImage image)//class BufferedImage called

{

//create new\_img with the attributes of image

BufferedImage new\_img = new BufferedImage(image.getWidth(), image.getHeight(), BufferedImage.TYPE\_3BYTE\_BGR);

Graphics2D graphics = new\_img.createGraphics();

graphics.drawRenderedImage(image, null);

return new\_img;

}

private byte[] get\_byte\_data(BufferedImage image)//Used to convert image into byte form

{

WritableRaster raster = image.getRaster();

DataBufferByte buffer = (DataBufferByte)raster.getDataBuffer();

return buffer.getData();

}

private byte[] bit\_conversion(int i)

{

//only using 4 bytes

byte byte3 = (byte)((i & 0xFF000000) >>> 24); //0

byte byte2 = (byte)((i & 0x00FF0000) >>> 16); //0

byte byte1 = (byte)((i & 0x0000FF00) >>> 8 ); //0

byte byte0 = (byte)((i & 0x000000FF) );

//{0,0,0,byte0} is equivalent, since all shifts >=8 will be 0

return(new byte[]{byte3,byte2,byte1,byte0});

}

private byte[] encode\_text(byte[] image, byte[] addition, int offset)

{

//check that the data + offset will fit in the image

if(addition.length + offset > image.length)

{

throw new IllegalArgumentException("File not long enough!");

}

//loop through each addition byte

for(int i=0; i<addition.length; ++i)

{

//loop through the 8 bits of each byte

int add = addition[i];

for(int bit=7; bit>=0; --bit, ++offset) //ensure the new offset value carries on through both loops

{

//assign an integer to b, shifted by bit spaces AND 1

//a single bit of the current byte

int b = (add >>> bit) & 1;

//assign the bit by taking: [(previous byte value) AND 0xfe] OR bit to add

//changes the last bit of the byte in the image to be the bit of addition

image[offset] = (byte)((image[offset] & 0xFE) | b );

}

}

return image;

}

private byte[] decode\_text(byte[] image)

{

int length = 0;

int offset = 32;

//loop through 32 bytes of data to determine text length

for(int i=0; i<32; ++i)

{

length = (length << 1) | (image[i] & 1);

}

byte[] result = new byte[length];

//loop through each byte of text

for(int b=0; b<result.length; ++b )

{

//loop through each bit within a byte of text

for(int i=0; i<8; ++i, ++offset)

{

//assign bit: [(new byte value) << 1] OR [(text byte) AND 1]

result[b] = (byte)((result[b] << 1) | (image[offset] & 1));

}

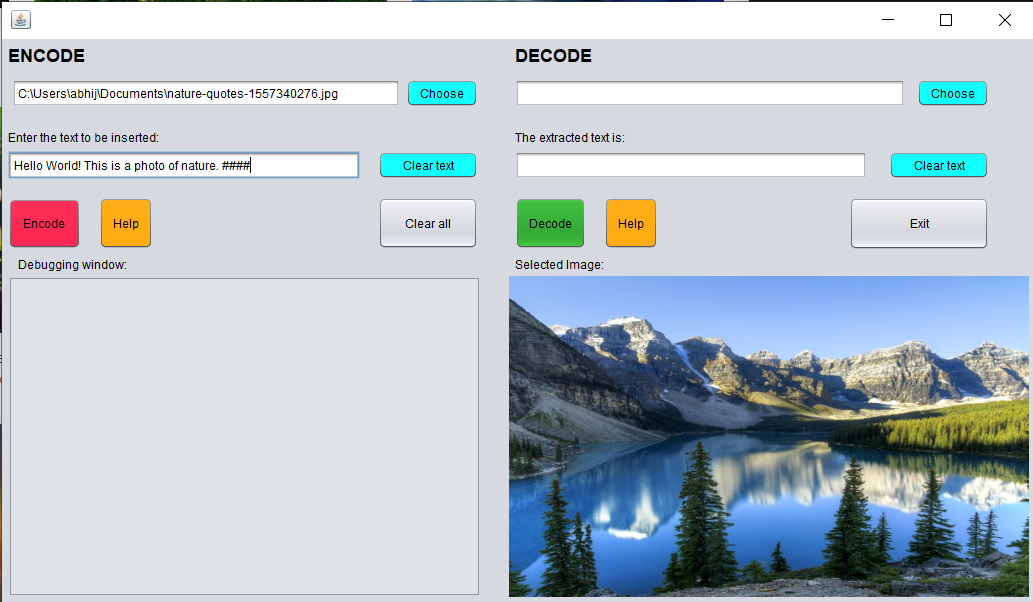
}

return result;

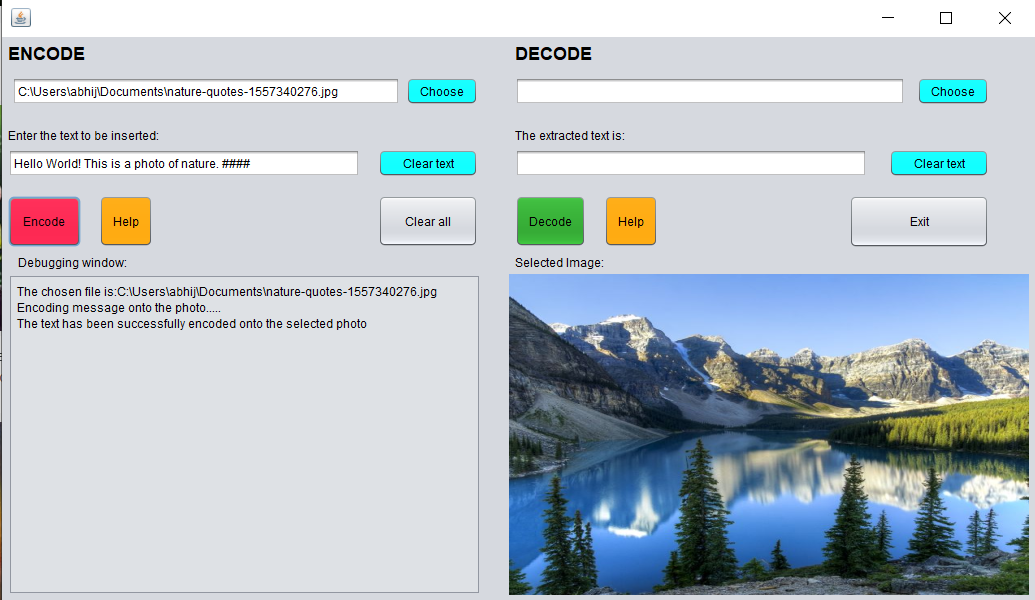
}

}

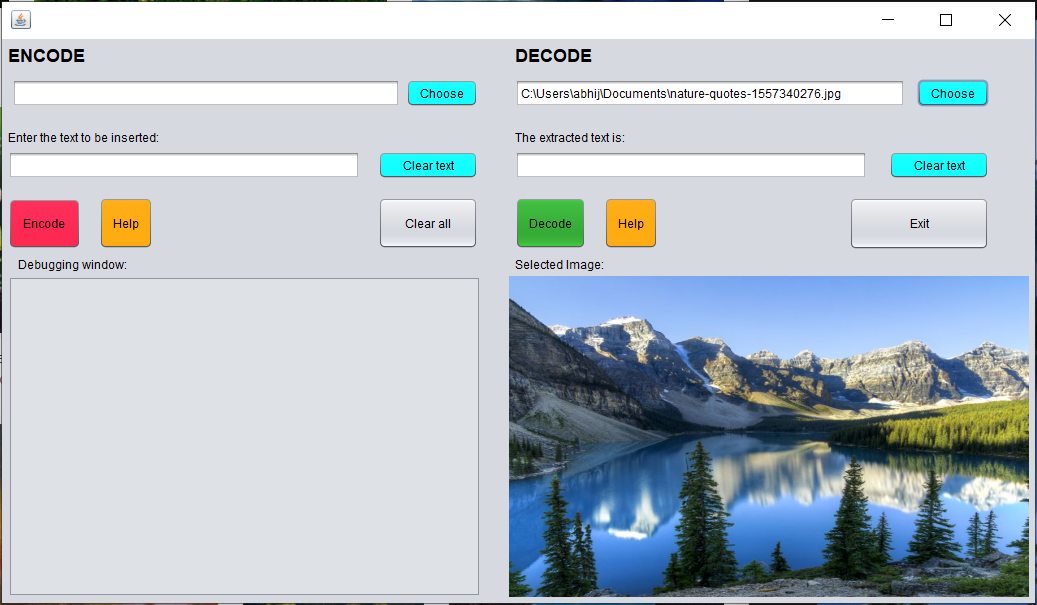
**OUTPUT**



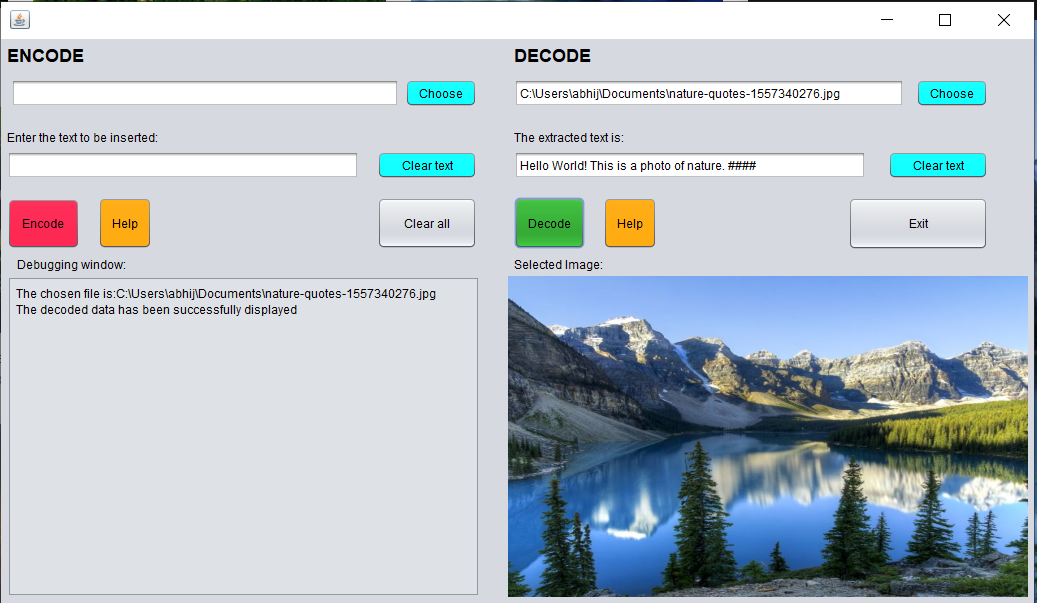
Before encoding the text



After encoding the text on the image file.



Before decoding the text from the image file.



After decoding, the text recovered is displayed on the text area.