/\*

\* To change this license header, choose License Headers in Project Properties.

\* To change this template file, choose Tools | Templates

\* and open the template in the editor.

\*/

package steganography;

import javax.swing.JLabel;

import java.awt.Image;

import java.awt.Dimension;

import javax.swing.ImageIcon;

import java.awt.\*;

import java.awt.color.ColorSpace;

import java.awt.image.\*;

import java.io.ByteArrayInputStream;

import java.io.File;

import java.io.FileOutputStream;

import java.io.IOException;

import java.io.UnsupportedEncodingException;

import java.util.\*;

import java.util.logging.Level;

import java.util.logging.Logger;

import javax.imageio.ImageIO;

import javax.swing.JOptionPane;

import steganography.ValueComparator;

public class Steganography extends JLabel {

public int height, width;

int[] data;

public Steganography() {

}

String path;

Image image, gray, binary, encodedImage;

int pixels[];

ImageIcon ic;

int binArray[];

ArrayList<String> arrayList = new ArrayList<>();

ArrayList<Integer> countArray = new ArrayList<>();

ArrayList<String> backupList = new ArrayList<>();

public String H1 = "";

public String H2 = "";

BufferedImage bufferedImage = null;

int backupHeight = 0;

int backupWidth = 0;

public void loadImage(String imagePath) throws Exception {

path = imagePath;

//setOpaque(true);

ic = new ImageIcon(imagePath);

System.out.println("imagePath : " + imagePath);

image = ic.getImage();

height = ic.getIconHeight();

width = ic.getIconWidth();

backupHeight = height;

backupWidth = width;

System.out.println("Height " + height + " " + width);

//setOpaque(false);

pixels = new int[width \* height];

binArray = new int[height \* width];

PixelGrabber pg = new PixelGrabber(image, 0, 0, width, height, pixels, 0, width);

pg.grabPixels();

//pg.gr

if (image != null) {

System.out.println(" image is there " + pixels[9]);

}

}

public BufferedImage convertTogray() {

System.out.println("Convert to gray ");

BufferedImage bi = null;

try {

for (int i = 0; i < width \* height; i++) {

int p = pixels[i];

int r = 0xff & (p >> 16);

int g = 0xff & (p >> 8);

int b = 0xff & (p);

int q = (int) (0.56 \* r + .33 \* g + b \* 0.11);

pixels[i] = (0xff000000 | q << 16 | q << 8 | q);

}

System.out.println("pixel[0] " + pixels[0]);

// bufferedImage = createImage(new MemoryImageSource(backupWidth, backupHeight, pixels, 0, width));

bi = convertToBufferedImage(gray);

} catch (Exception e) {

// JOptionPane.showMessageDialog(this, e.getMessage(), "Convert 2 Gray", JOptionPane.ERROR\_MESSAGE);

}

return bi;

} // END of function convert2gray

public int convertToBinary() {

int[] imageArray = new int[width \* height];

for (int i = 0; i < width \* height; i++) {

int p = pixels[i];

int r = 0xff & (p >> 16);

int g = 0xff & (p >> 8);

int b = 0xff & (p);

int q = (int) (0.56 \* r + .33 \* g + b \* 0.11);

//temp

// imageArray[i] = q;

if (q > 200) {

q = 255;

imageArray[i] = 255;

binArray[i] = 1;

} else {

q = 0;

imageArray[i] = 0;

binArray[i] = 0;

}

pixels[i] = (0xff000000 | q << 16 | q << 8 | q);

}

System.out.println("pixel in binary " + pixels[0]);

binary = createImage(new MemoryImageSource(width, height, pixels, 0, 0));

// File f = new File("C:\\Users\\Aditi\\Desktop\\output\_1.jpg");

// try {

// ImageIO.write(convertToBufferedImage(binary), "jpg", f);

// } catch (Exception e) {

// }

String patternString[] = new String[width \* height / 3];

int k = 0;

for (int m = 0; m < height / 3; m++) {

for (int n = 0; n < width / 3; n++) {

patternString[k] = "";

for (int i = 0; i < 3; i++) {

for (int j = 0; j < 3; j++) {

patternString[k] = patternString[k].concat(Integer.toString(binArray[(width \* (3 \* m + i)) + (3 \* n + j)]));

}

}

System.out.println("block pattern =" + patternString[k]);

arrayList.add(patternString[k]);

backupList.add(patternString[k]);

k++;

}

}

for (String string : arrayList) {

countArray.add(Integer.parseInt(string, 2));

}

arrayList.clear();

for (Integer integer : countArray) {

arrayList.add(String.format("%9s", Integer.toBinaryString(integer)).replace(' ', '0'));

}

Map<String, Integer> map = new HashMap<>();

Set<String> set = new HashSet(arrayList);

ArrayList<String> al = new ArrayList<>(set);

for (int l = 0; l < al.size(); l++) {

int count = 0;

for (int o = 0; o < arrayList.size(); o++) {

if (al.get(l).contentEquals(arrayList.get(o))) {

count++;

}

map.put(al.get(l), count);

}

}

ValueComparator bvc = new ValueComparator(map);

TreeMap<String, Integer> treeMap = new TreeMap<>(bvc);

treeMap.putAll(map);

int count = 0;

String headers[] = new String[2];

int[] countOfHeaders = new int[2];

for (Map.Entry<String, Integer> entry : treeMap.entrySet()) {

String key = entry.getKey();

Integer value = entry.getValue();

System.out.println(key+" "+value);

}

for (Map.Entry<String, Integer> entry : treeMap.entrySet()) {

if (count == 2) {

break;

}

String key = entry.getKey();

Integer value = entry.getValue();

System.out.println(key + " " + value);

headers[count] = key;

countOfHeaders[count] = value;

count++;

}

H1 = headers[0];

H2 = headers[1];

System.out.println("H1 = " + H1 + " H2= " + H2);

return 1;

}

String stringToBinary(String s) {

byte[] bytes = s.getBytes();

StringBuilder binary = new StringBuilder();

for (byte b : bytes) {

int val = b;

for (int i = 0; i < 8; i++) {

binary.append((val & 128) == 0 ? 0 : 1);

val <<= 1;

}

}

return binary.toString();

}

public String encrypt(String stringForEncoding, String keyForEncoding) {

String string = stringToBinary(stringForEncoding);

String key = stringToBinary(keyForEncoding);

int[] sourceString = new int[string.length()];

int[] keyString = new int[key.length()];

char[] cs = string.toCharArray();

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

sourceString[i] = Integer.parseInt(String.valueOf(cs[i]));

}

char[] keyArray = key.toCharArray();

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

keyString[i] = Integer.parseInt(String.valueOf(keyArray[i]));

}

int counter = 0;

int[] result = new int[sourceString.length];

for (int i = 0; i < keyArray.length; i++, counter++) {

result[i] = sourceString[i] ^ keyArray[i];

}

for (int i = 0; i < counter; i++) {

result[i] = result[i] - 48;

}

for (int i = counter; i < sourceString.length; i++) {

result[i] = sourceString[i];

}

StringBuilder builder = new StringBuilder();

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

builder.append(result[i]);

}

return builder.toString();

}

public String decrypt(String source, String key) {

char[] sourceString = source.toCharArray();

char[] keyArray = key.toCharArray();

int[] sourceStringInt = new int[sourceString.length];

int[] keyArrayInt = new int[keyArray.length];

int[] result = new int[sourceString.length];

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

sourceStringInt[i] = Integer.parseInt(String.valueOf(sourceString[i]));

}

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

keyArrayInt[i] = Integer.parseInt(String.valueOf(keyArray[i]));

}

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

result[i] = sourceStringInt[i] + 48;

}

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

result[i] = result[i] ^ keyArray[i];

}

for (int i = keyArray.length; i < sourceString.length; i++) {

result[i] = sourceStringInt[i];

}

StringBuilder builder = new StringBuilder();

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

builder.append(result[i]);

}

return builder.toString();

}

ArrayList<String> newList = new ArrayList<>();

public BufferedImage encode(String encrypted, String H1, String H2) {

char[] array = encrypted.toCharArray();

int[] intArray = new int[backupList.size() \* 9];

int counter = 0;

System.out.println("Encrypted msg is " + encrypted + " H1 = " + H1 + " H2 = " + H2);

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

if (backupList.get(i).contentEquals(H1) && array[i] == '1') {

System.out.println("embedded without changing bit in H1 " + H1 + " array" + array[i]);

newList.add(backupList.get(i));

} else if (backupList.get(i).contentEquals(H1) && array[i] == '0') {

System.out.println("Corner bit changed in H1 " + H1 + " array" + array[i]);

String temp = backupList.get(i);

char[] tempArray = temp.toCharArray();

if (tempArray[0] == '0') {

tempArray[0] = '1';

} else {

tempArray[0] = '0';

}

newList.add(String.copyValueOf(tempArray));

System.out.println(String.copyValueOf(tempArray));

} else if (backupList.get(i).contentEquals(H2) && array[i] == '0') {

System.out.println("embedded without changing bit in H2" + H2 + " array" + array[i]);

newList.add(backupList.get(i));

} else if (backupList.get(i).contentEquals(H2) && array[i] == '1') {

System.out.println("Corner bit changed in H2 " + H2 + " array" + array[i]);

String temp = backupList.get(i);

char[] tempArray = temp.toCharArray();

if (tempArray[0] == '0') {

tempArray[0] = '1';

} else {

tempArray[0] = '0';

}

newList.add(String.copyValueOf(tempArray));

} else {

newList.add(backupList.get(i));

}

counter++;

}

for (int i = counter; i < backupList.size(); i++) {

newList.add(backupList.get(i));

}

System.out.println(newList.size() + " new ");

for (int i = 0; i < backupList.size(); i++) {

System.out.println("backup " + backupList.get(i) + " new " + newList.get(i));

}

int j = 0;

for (int i = 0; i < newList.size(); i++) {

char[] temp = newList.get(i).toCharArray();

for (int k = 0; k < temp.length; k++) {

if (temp[k] == '1') {

intArray[j++] = 255;

} else {

intArray[j++] = 255;

}

}

}

System.out.println("newList.size()\*9 " + newList.size() \* 9);

System.out.println("j = " + j);

System.out.println("Height " + height + " " + width);

Image image = createImage(new MemoryImageSource(width, height, pixels, 0, width));

BufferedImage img = convertToBufferedImage(image);

return img;

}

public static BufferedImage convertToBufferedImage(Image image) {

BufferedImage newImage = new BufferedImage(

image.getWidth(null), image.getHeight(null),

BufferedImage.TYPE\_BYTE\_GRAY);

Graphics2D g = newImage.createGraphics();

g.drawImage(image, 0, 0, null);

g.dispose();

return newImage;

}

public String decode(String key) {

return null;

}

public static String toText(String info) {

String[] array = new String[info.length() / 8];

int j = 0;

for (int i = 0; i < info.length(); i = i + 8) {

array[j] = info.substring(i, i + 8);

j++;

}

int[] intArray = new int[array.length];

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

intArray[i] = Integer.parseInt(array[i], 2);

}

StringBuilder builder = new StringBuilder();

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

builder.append((char) intArray[i]);

}

return builder.toString();

}

public BufferedImage encodeMessage(String msg,String filePath) {

try {

bufferedImage = ImageIO.read(new File(filePath));

} catch (IOException ex) {

Logger.getLogger(Steganography.class.getName()).log(Level.SEVERE, null, ex);

}

int w = bufferedImage.getWidth();

int h = bufferedImage.getHeight();

System.out.println("message=" + msg + " length=" + msg.length());

if (msg.length() > 255) {

System.out.println("MESSAGE IS LARGE THAN 255 CHARACTERS");

} else if (msg.length() \* 11 > w \* h) {

System.out.println("Image is too small");

} else {

byte[] msgbytes = msg.getBytes();

int msglendecode = (bufferedImage.getRGB(0, 0) >> 8) << 8;

msglendecode |= msg.length();

bufferedImage.setRGB(0, 0, msglendecode);//hidig msg length at first position

for (int i = 1, msgpos = 0, row = 0, j = 0; row < h; row++) {

for (int col = 0; col < w && j < msgbytes.length; col++, i++) {

if (i % 11 == 0) {

int rgb = bufferedImage.getRGB(col, row);

int a = ((rgb >> 24) & 0xff);

int r = (((rgb >> 16) & 0xff) >> 3) << 3;

r = r | (msgbytes[msgpos] >> 5);

int g = (((rgb >> 8) & 0xff) >> 3) << 3;

g = g | ((msgbytes[msgpos] >> 2) & 7);

int b = ((rgb & 0xff) >> 2) << 2;

b = b | (msgbytes[msgpos] & 0x3);

rgb = 0;

rgb = (rgb | (a << 24));

rgb = (rgb | (r << 16));

rgb = (rgb | (g << 8));

rgb = (rgb | b);

bufferedImage.setRGB(col, row, rgb);

msgpos++;

j++;

}

}//for 2

}//for 1

}//else

return bufferedImage;

}

public String decryptMessage(BufferedImage bimg) {

System.out.println("in decode");

width = bimg.getWidth();

height = bimg.getHeight();

int msglength = (bimg.getRGB(0, 0) & 0xff);

StringBuilder builder = new StringBuilder();

System.out.println("Message Length=" + msglength);

for (int row = 0, j = 0, i = 1; row < height; row++) {

for (int col = 0; col < width && j < msglength; col++, i++) {

if (i % 11 == 0) {

int result = bimg.getRGB(col, row);

int charatpos = (((result >> 16) & 0x7) << 5);

charatpos |= (((result >> 8) & 0x7) << 2);

charatpos |= ((result & 0x3));

// System.out.print((char) charatpos);

builder.append((char) charatpos);

j++;

}

}

}

System.out.println(builder.toString());

System.out.println("decoding done");

return builder.toString();

}//function

public static void main(String[] args) {

Steganography steganography = new Steganography();

try {

steganography.loadImage("C:\\Users\\Aditi\\Documents\\Baboon.jpeg");

} catch (Exception ex) {

Logger.getLogger(Steganography.class.getName()).log(Level.SEVERE, null, ex);

}

BufferedImage bufferedImage = steganography.convertTogray();

int status = steganography.convertToBinary();

String encrypted = steganography.encrypt("message", "key");

bufferedImage = steganography.encode(encrypted, steganography.H1, steganography.H2);

}

}