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\_orig = getImage(file\_path);

BufferedImage image = user\_space(image\_orig);

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 image = null;

File file = new File(f);

try

{

image = ImageIO.read(file);

}

catch(Exception ex)

{

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

}

return image;

}

private boolean setImage(BufferedImage image, File file, String ext)

{

try

{

file.delete(); //delete resources used by the File

ImageIO.write(image,ext,file);

return true;

}

catch(Exception e)

{

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 positiong

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)

{

//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);

graphics.dispose(); //release all allocated memory for this image

return new\_img;

}

private byte[] get\_byte\_data(BufferedImage image)

{

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) //i=24 will also work, as only the 4th byte contains real data

{

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;

}

}