LAMPIRAN

Proses Encoding

```
public class Steganography_File
{
public static final int OFFSET = 64;
public static final String VERSION= "";
private String message;
private String message2;
private String outputFile, dataFile, inputFile;
private DataInputStream in, in2, data;
private DataOutputStream out, out2;
private int i, j, size;
private short messageSize, temp;
private int dataFileSize, tempInt, vectorSize;
byte by,byt, byb;
public Steganography_File()
        {
                i=0;
                j=0;
                vectorSize= 0;
```

```
        }
        public String getMessage()
        {
                return message;
        }
public boolean encodeMessage(String msg, String inFile, String outFile)
        {
                message= msg;
                inputFile= inFile;
                outputFile= outFile;
                try
                {
// Open the data file
in=newDataInputStream(newFileInputStream(inputFile));
out=new DataOutputStream(new FileOutputStream(outputFile));
// Save OFFSET bytes
for(i=0; i<= OFFSET; i++)
out.writeByte(in.readByte());
messageSize= (short) message.length();
// Embed the size of the message
```

```
// 8 bit size of the message, will be stored as 4 pairs of 2 bits
// Shift the bits in pair of 2 to least significant position
for(i=6; i>=0; i-=2)
        {
temp= messageSize;
temp>>=i;
by= (byte) temp;
by&= 0x03;
        // Write these bytes to output file
byt&= 0xFC;
byt|= by;
out.writeByte(byt);
}
for(i=0; i<messageSize; i++)  // Embed the message
{
byt= (byte) message.charAt(i);
byt&= 0x7F;
for(j=6; j>=0; j-=2)
{by= byt;
by>>= j;
```

```
by&= 0x03;
// Write these bytes to the output file
byb&= 0xFC;
byb|= by;
out.writeByte(byb);
}
        }
        // Write the remaining bytes
while(true)
{
byt= in.readByte();
out.writeByte(byt);
}
        }
catch(EOFException e)
{
        }
catch(Exception e)
{
message= "Oops!!\nError: "+ e.toString();
```

```
return false;
}
finally{
        try
        {
        in.close();
        out.close();
                }
                catch(Exception ex)
                {
                message= "Oops!!\nError: "+ ex.toString();
                        return false;
                        }
                }
        message= "Penggabungan Berhasil...";
        return true;
        }
        public String decodeMessage(String inFile)
        {
                inputFile= inFile;
```

```
                boolean flag= true;
                char mesg[];
                String messg = ".";
                try
                {
            System.out.println(in);
in= new DataInputStream(new FileInputStream(inputFile));
// Get the size of the message
messageSize= 0;
        for(i=0; i<=OFFSET; i++)     // Skip OFFSET bytes
                in.readByte();
            * Retrieve the data file
                messageSize= 0;
                // Retrieve the size of the file name
                        for(i=6; i>=0; i-=2)
                        {
                        by= in.readByte();
                        by&= 0x03;
                        temp= (short) by;
                        temp<<=i;
```

```
            messageSize|= temp;
                        }
   System.out.println("msize:"+messageSize);
                        /*
    if(messageSize<=0){
message= "This file does not contain a message.";
return("#FAILED#");
                        }*/
// Retrieve the message
mesg= new char[messageSize];
for(i=0; i<messageSize; i++)
                        {
                                by= 0;
                                for(j=6; j>=0; j-=2)
                                {
                                        byt= in.readByte();
                                        byt&= 0x03;
                                        byt<<= j;
                                        by|= byt;
                                }by&= 0x7F;
```

```
                    mesg[i]= (char) by;
                }
messg= new String(mesg);
        }
        catch(Exception e)
        {
                message= "Oops!!\n Error: "+ e;
                return("#FAILED#");
        }
        finally
        {
                try
                {
                        in.close();
                }
                catch(Exception ex)
                {
                        message= "Oops!!\nError: "+ ex;
                        return("#FAILED#");
                }
```

```
message= "Message size: "+ messageSize+ " B";
        System.out.println("pesan:"+messg);
                return messg;
        }
}
public boolean encodeFile(String inFile, String datFile, String outFile)
{
        inputFile   = inFile;
        dataFile    = datFile;
        outputFile  = outFile;
        try
        {
in= new DataInputStream(new FileInputStream(inputFile));
data=new DataInputStream(new FileInputStream(dataFile));
out=new DataOutputStream(new FileOutputStream(outputFile));
        // Write OFFSET bytes from input file to output file
                for(i=0; i<= OFFSET; i++)
                out.writeByte(in.readByte());
                File file= new File(dataFile);
                message= file.getName();
```

```
                    messageSize= (short) message.length();
                    dataFileSize= (int) file.length();
                    // Embed the size of the file name
// 8 bit size of the file name, will be stored as 4 pairs of 2 bits
// Shift the bits in pair of 2 to least significant position
                    for(i=6; i>=0; i-=2)
                    {
                        temp= messageSize;
                        temp>>=i;
        by= (byte) temp;
        by&= 0x03;
// Write these bytes to output file
        byt&= 0xFC;
        byt|= by;
        out.writeByte(byt);
                    }
                    // Embed the message in the file
            for(i=0; i<messageSize; i++)
                    {
                        by= (byte) message.charAt(i);
```

```
                                        by&=0x7F;
                                        for(j=6; j>=0; j-=2)
                                        {
                                                byt= by;
                                                byt>>= j;
                                                byt&= 0x03;
        // Write these bytes to output file
                        byb&= 0xFC;
                        byb|= byt;
                        out.writeByte(byb);
                                        }
                                }
        // Embed the size of the data file
// dataFileSize being 32 bit long type, will be stored in 16 pairs of 2 bits
                                for(i=30; i>=0; i-=2)
                                {
                        tempInt= dataFileSize;
                                        tempInt>>=i;
                                        by= (byte) tempInt;
                                        by&= 0x03;
```

```
                // Write these bytes to output file
                        byt&= 0xFC;
                        byt|= by;
                        out.writeByte(byt);
                }
                // Embed the file
                while(true)
                {
                        byt= data.readByte();
                        for(j=6; j>=0; j-=2)
                        {
                                by= byt;
                                by>>= j;
                                by&= 0x03;
// Write these bytes to output file
                                byb&= 0xFC;
                                byb|= by;
                                out.writeByte(byb);
                        }
                }
```

```
            }
            catch(EOFException e) {}
            catch(Exception ex)
            {
message= "Oops!!\nError: "+ ex.toString();
            return false;
            }
//write original carrier file name information
try
{
   File file= new File(inputFile);
   message= file.getName();
   messageSize= (short) message.length();
   // Embed the size of the file name
         // 8 bit size of the file name, will be stored as 4 pairs of 2 bits
         // Shift the bits in pair of 2 to least significant position
   for(i=6; i>=0; i-=2)
   {
         temp= messageSize;
         temp>>=i;
```

```
        by= (byte) temp;
        by&= 0x03;
        // Write these bytes to output file
        byt&= 0xFC;
        byt|= by;
        out.writeByte(byt);
}
// Embed the message in the file
for(i=0; i<messageSize; i++)
{
        by= (byte) message.charAt(i);
        by&=0x7F;
        for(j=6; j>=0; j-=2)
        {
                byt= by;
                byt>>= j;
                byt&= 0x03;
                // Write these bytes to output file
                byb&= 0xFC;
                byb|= byt;
```

```
                out.writeByte(byb);
            }
        }
                }
                catch(EOFException e) {}
                catch(Exception ex)
                {
                message= "Oops!!\nError: "+ ex.toString();
                        return false;
                }
                try
                {
// Write the remaining bytes from the input file
                        while(true)
                        {
                                byt= in.readByte();
                                out.writeByte(byt);
                        }
                }
                catch(EOFException e) {}
```

```
            catch(Exception e)
            {
message= "Oops!!\nError: "+ e.toString();
                        return false;
            }
            finally
            {
                        try
                        {
                                    in.close();
                                    data.close();
                                    out.close();
                        }
                        catch(Exception ex)
                        {
message= "Oops!!\nError: "+ ex.toString();
                                    return false;
                        }
            }
```

```
                message= "File Sisipan Berhasil Digabungkan "+ new
File(outputFile).getName();

                return true;

        }

        public String decodeFile(String inFile)

        {

                char fileName[];

                inputFile= inFile;

                try

                {

in= new DataInputStream(new FileInputStream(inputFile));

                        // Skip past Offset bytes

                        for(i=0; i<= OFFSET; i++)

                        in.readByte();

                * Retrieve the data file

                */

                        messageSize= 0;

                        // Retrieve the size of the file name

                        for(i=6; i>=0; i-=2)

                        {
```

```
                by= in.readByte();
                by&= 0x03;
                temp= (short) by;
                temp<<=i;
                messageSize|= temp;
}
// Retrieve the message
fileName= new char[messageSize];
for(i=0; i<messageSize; i++)
{
                by= 0;
                for(j=6; j>=0; j-=2)
                {
                                byt= in.readByte();
                                byt&= 0x03;
                                byt<<= j;
                                by|= byt;
                }
                by&= 0x7F;
                fileName[i]= (char) by;
```

```
                    }
                    message= new String(fileName);
                    // Retrieve the data file size
                    dataFileSize= 0;
                    for(i=30; i>=0; i-=2)
                    {
                              by= in.readByte();
                              tempInt= (int) by;
                              tempInt&= 0x00000003;
                              tempInt<<=i;
                              dataFileSize|= tempInt;
                    }
          // Retrieve the data file
          // Create a new file with the name just retrieved
out= new DataOutputStream(new FileOutputStream(message));
                    for(i=0; i<dataFileSize; i++)
                              {by= 0;
                                        for(j=6; j>=0; j-=2)
                                        {
                                        byt= in.readByte();
```

```
                                        byt&= 0x03;
                                        byt<<= j;
                                        by|= byt;
                                }
                                out.writeByte(by);
                        }
/*
 * Retrieve the carrier file
 */
                        messageSize= 0;
                        // Retrieve the size of the file name
                        for(i=6; i>=0; i-=2)
                        {
                                by= in.readByte();
                                by&= 0x03;
                                temp= (short) by;
                                temp<<=i;
                                messageSize|= temp;
                        }
                        // Retrieve the message
```

```
                    fileName= new char[messageSize];
                    for(i=0; i<messageSize; i++)
                    {
                            by= 0;
                            for(j=6; j>=0; j-=2)
                            {
                                    byt= in.readByte();
                                    byt&= 0x03;
                                    byt<<= j;
                                    by|= byt;
                            }
                            by&= 0x7F;
                            fileName[i]= (char) by;
                    }
message2= new String(fileName);
// Retrieve the data file
// Create a new file with the name just retrieved
out2=newDataOutputStream(newFileOutputStream(message2));
 in2= new DataInputStream(new FileInputStream(inputFile));
     // Write OFFSET bytes from input file to output file
```

```
                for(i=0; i<=OFFSET; i++)
                {
                        by= in2.readByte();
                        out2.writeByte(by);
                }
// Write the remaining bytes from the input file
while(true)
{
   byt= in.readByte();
   out2.writeByte(byt);
}
        }
        catch(EOFException e) {}
        catch(Exception ex)
        {
        message= "Oops!!\nError: "+ ex.toString();
                return "#FAILED#";
        }
        finally
        {
```

```
                    try
                    {
                            in.close();
            in2.close();
                            out.close();
            out2.close();
            File file = new File(inFile);
            if(file.delete()){
               //delete success
            }
                    }
                    catch(Exception ex)
                    {
message= "Oops!!\nError: "+ ex.toString();
            return "#FAILED#";
                    }
            }
message= "Data sisipan berhasil dipisahkan "+ message;
            return message;
}}
```

Proses Decode

```
private boolean validateInput()
        {       if(txtInFile.getText().length()<=0)
                {
JOptionPane.showMessageDialog(this, "pilih Carrier", "membutuhkan file Carrier", JOptionPane.WARNING_MESSAGE);
        return false;
                }
                return true;
        }
        private boolean updateEmbedability()
        {
                if(inFileExists)
                {
                lblSizeIn.setText(""+ inFile.length()/1024+ " Kb");
return true;
                }
                return false;
        }
        public void actionPerformed(ActionEvent e)
```

```
{
        Object obj= e.getSource();
        // Input Browse button was pressed
        if(obj== btnBrowseIn)
        {
if(fileChooser.showOpenDialog(this)== fileChooser.APPROVE_OPTION)
                {
                        inFile= fileChooser.getSelectedFile();
                        txtInFile.setText(inFile.getPath());
                        if(!inFile.exists())
                        {
JOptionPane.showMessageDialog(this, "Input file "'+ inFile.getName()+'" does not exist!!", "File does not exist!", JOptionPane.WARNING_MESSAGE);
        inFileExists= false;
        }
                else
                        {
                        inFileExists= true;
                        updateEmbedability();
                        }
```

```
                    }
                }
                // Decode button was pressed
                if(obj== btnExtraction)
                {
                    action= false;
// Decode
                    if(validateInput())
                    {
            operationStarted("Proses Memisahkan.,Tunggu sebentar...");
backend= new InformasiJava("pisah", txtInFile.getText(), txtDataFile.getText(),
txtOutFile.getText(), this);
                    backend.start();
                    }
                    else return;
                }
                if(obj== btnExit)
            {
               view.setVisible(true);
               dispose();
```

```
        }
        }
        private void operationStarted(String status)
        {
                btnBrowseIn.setEnabled(false);
                btnExtraction.setEnabled(false);
                btnExit.setEnabled(false);
                lblStatus.setText(status);
        }
        public void operationComplete(String status)
        {
                btnBrowseIn.setEnabled(true);
                btnExtraction.setEnabled(true);
                btnExit.setEnabled(true);
                lblStatus.setText(status);
        }
}
        private void updateEmbedability()
        {
                message= txtMessage.getText();
```

```
                int messageSize= message.length();

lblStatus.setText("Minimum input file size required: "+ (messageSize*4+ 20+
OFFSET) + " B ("+ ((messageSize*4+ 20+ OFFSET)/1024+ 1)+ " Kb)");

        if(txtInFile.getText().length()>0 && messageSize>0)

                {

                        if(inputFileSize- OFFSET>messageSize*4)

                        isEmbeddable= true;

                        else

                        isEmbeddable= false;

                }

        }

        public void actionPerformed(ActionEvent e)

        {

                Object obj= e.getSource();

                // Input Browse button was pressed

                if(obj== btnBrowseIn)

                {

                if(fileChooser.showOpenDialog(this)==
fileChooser.APPROVE_OPTION)

                        {

                String path= fileChooser.getSelectedFile().getPath();
```

```
                                        txtInFile.setText(path);
                                        inputFileSize= new File(path).length();
                                lblSizeIn.setText(""+ inputFileSize+ " B");
                                        updateEmbedability();
                                }
                        }
                        // Output Browse button was pressed
                        if(obj== btnBrowseOut)
                        {
                if(fileChooser.showSaveDialog(this)==
fileChooser.APPROVE_OPTION)
                        txtOutFile.setText(fileChooser.getSelectedFile().getPath());
                        }
                        // Encode button was pressed
                        if(obj== btnEmbed)
                        {
                                action= true;
                                if(validateInput())
                                {
                        operationStarted("Encoding... please wait...");
```

```
backend= new InformasiJava(txtMessage.getText(), txtInFile.getText(), null,
txtOutFile.getText(), this);
                                backend.start();
                                }
                                else return;
                        }
                        // Decode button was pressed
                        if(obj== btnRetrieve)
                        {
                                action= false;
                                if(validateInput())
                                {
        operationStarted("Decoding... please wait...");
backend= new InformasiJava(txtMessage.getText(), txtInFile.getText(), null,
txtOutFile.getText(), this);
                        backend.start();}else return;}
```