CHAPTER V
IMPLEMENTATION AND TESTING

5.1 Implementation

The implementation will use Java programming language. Class that will be used for Encode and Decode process will be created first to test in advance to make it easier to monitor if there are errors.

5.1.1 GUI

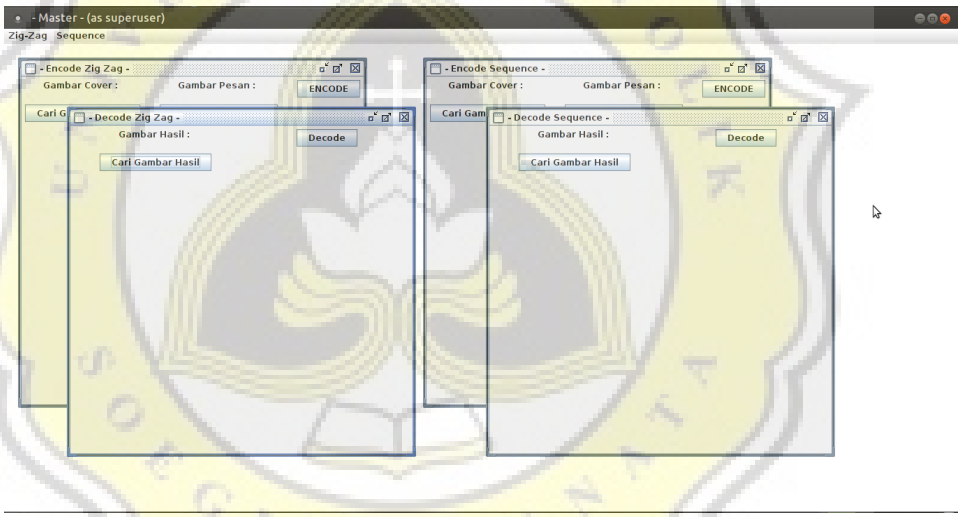

Figure 8. Graphical User Interface

There will be two menus, Zig-Zag and Sequence. From Zig-Zag and Sequence menu, there will be two sub menu, Encode and Decode. For the Zig-Zag menu, process of encode and decode will use zig-zag pattern, for the Sequence menu, process of encode and decode will use sequence pattern. There is no difference of interface between Zig-Zag and Sequence menu, the difference is just the pattern to encode and decode.

- **Encode Zig-zag**

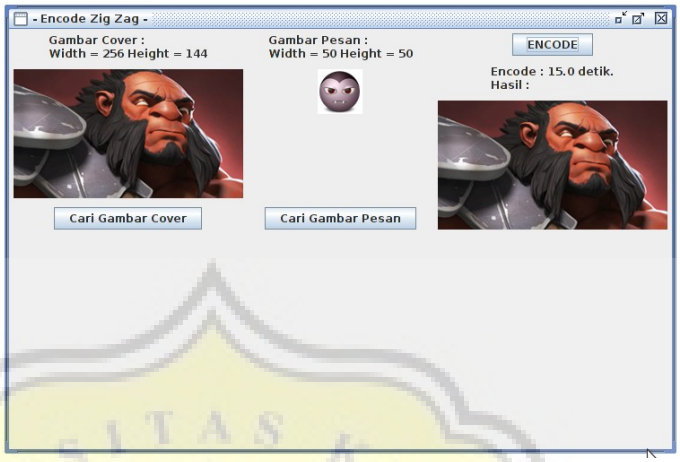

Figure 9. GUI Encode Zig-zag

Encode process using zig-zag pattern can be started if cover image and message image have been chosen, press 'Cari Gambar Cover' to choose cover image, and press 'Cari Gambar Pesan' to choose message image, to start Encoding Process press 'ENCODE'. The message image will be encoded into cover image that produce new image (result image).

- **Encode Sequence**

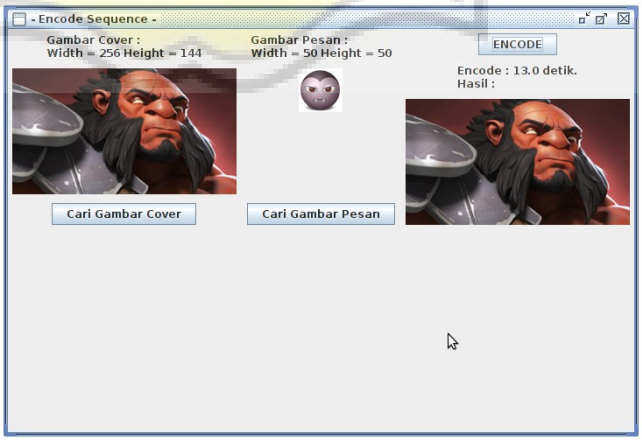

Figure 10. GUI Encode Sequence

Encode using sequence pattern can be started if cover image and message image have been chosen. Button 'Cari Gambar Cover' is used to choose cover image, and button 'Cari Gambar Pesan' is used to choose message image, button 'ENCODE' to start Encoding Process. The message image will be encoded into cover image that produce new image.

- **Decode Zig-zag**

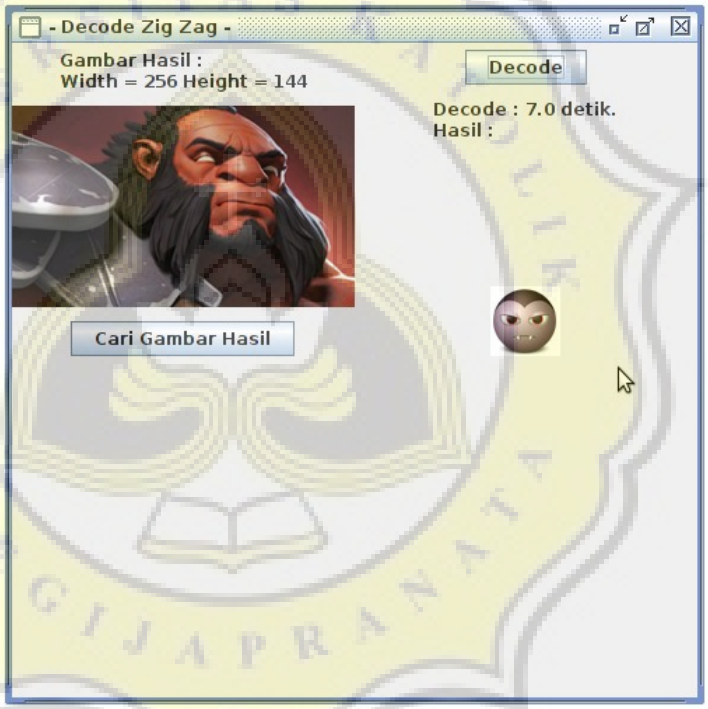

Figure 11. GUI Decode Zig-zag

Decode process using zig-zag pattern can be started if an image is chosen. To choose an image, press 'Cari Gambar Hasil' and then press 'Decode' to begin decoding process that produce result image or in this case message image that encoded in chosen image.

- **Decode Sequence**

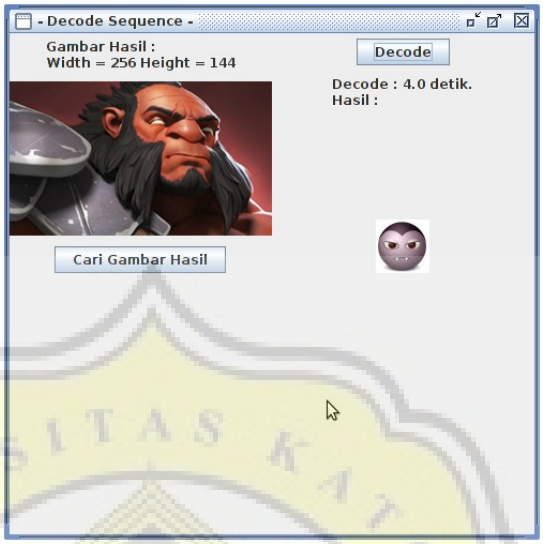

Figure 12. GUI Decode Sequence

Decode process using sequence pattern can be started if an image is chosen. Button 'Cari Gambar Hasil' is used to choose an image and then Button 'Decode' is used to begin decoding process that produce message image that encoded in chosen image.

- **Error**

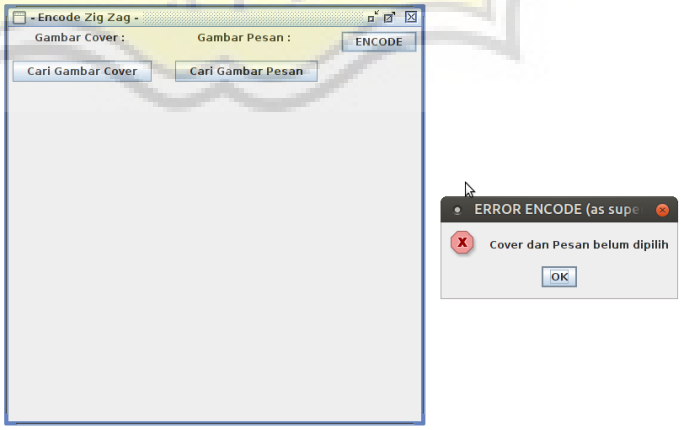

Figure 13. GUI Error No File

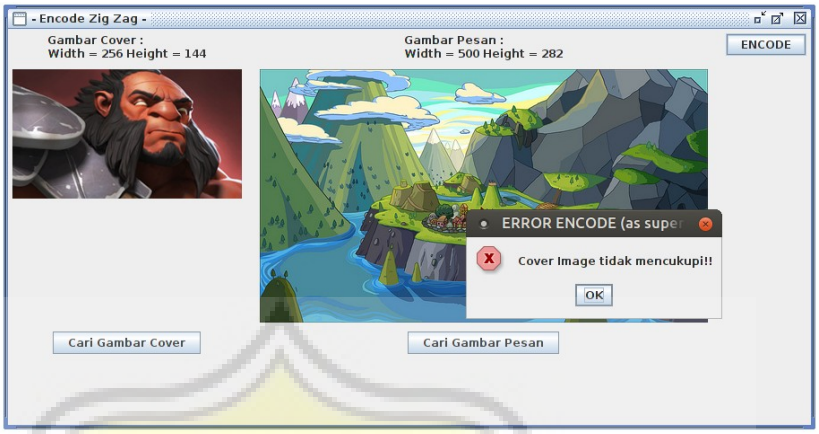

Figure 14. GUI Space is Not Enough

There will be some error handling, if there is no file (*Figure 13*) and Space for encoding the message image is not enough (*Figure 14*). For the error, the program will give dialog box and show the message of the error.

5.1.2 Code

- **Encode Zig-zag**

```
public void isi(int x, int y)
{
        String convertAlpha = "";
        String convertRed = "";
        String convertGreen = "";
        String convertBlue = "";
        if(counter <= (8*widthPesan*heightPesan))
        {
            System.out.print("Encode = " +counter);
            System.out.print(" CounterBiner = " +counterBiner+ "\n");

            if(counter < 8)
            {
                System.out.print("ISI MATRIX >> W : " +y+ " H : " +x+ "\n");

                    Biner binerAlpha = new Biner(pAlpha);
                    Biner binerRed = new Biner(pRed);
                    Biner binerGreen = new Biner(pGreen);
                    Biner binerBlue = new Biner(pBlue);

                    int []hasilBinerAlpha = binerAlpha.getBiner();
                    System.out.print("PESAN >> Biner Alpha : " );
                    for(int i=0 ; i<8 ; i++)
                    {
                        System.out.print(hasilBinerAlpha[i]);
                    }
```

Figure 15. Encode Zig-zag Process

Encode process with zig-zag pattern will require cover image and message image. Encode can be started when there are two images and the cover image's size is more than message image. New buffered image will be needed to generating the result image. The step of encode process is read the cover image with zig-zag matrix and breaking cover image and message image into pixels and then binary value can be established. A pixel contains four color elements, alpha, red, green, blue. Alpha is for transparency of the pixel and then red, green and blue is for determining each pixel color combinations. When binary value has been established, encoding process using LSB can be done. After this step is done, the result image will be generated.

- **Encode Sequence**

```
public EncodeSequence(String pFileCover, String pFilePesan)
{
    try
    {
        File input = new File(pFileCover);
        cover = ImageIO.read(input);
        widthCover = cover.getWidth();
        heightCover = cover.getHeight();

        hasil = new BufferedImage(widthCover,heightCover,BufferedImage.TYPE_INT_ARGB);

        System.out.println("Panjang : " + widthCover + " Lebar : " + heightCover);


        File message = new File(pFilePesan);
        pesan = ImageIO.read(message);
        widthPesan = pesan.getWidth();
        heightPesan = pesan.getHeight();

        System.out.println("Pesan : ");
        System.out.println("Panjang - " + widthPesan + " Lebar - " + heightPesan);

        if((widthPesan*heightPesan*8) > (widthCover*heightCover) || widthCover < 16 || heightCover < 16)
        {
            messages = "Cover Image tidak mencukupi!!";
            System.out.println("Cover Image tidak mencukupi!!");
```

Figure 16. Encode Sequence Process

Encode process using sequence pattern is like encode process using zig-zag pattern. This encode process using sequence pattern to read and substitute the LSB value in cover image.

- **Decode Zig-zag**

```
public void baca(int x, int y)
{
    System.out.println("width = " +y+ " height = " +x);

    if(counter <= (widthPesan * heightPesan * 8))
    {
        //System.out.println("Baca");

        int copyGambarHasil = gambarHasil.getRGB(y,x);

        newAlpha = (copyGambarHasil>>24) & 0xff;
        newRed = (copyGambarHasil>>16) & 0xff;
        newGreen = (copyGambarHasil>>8) & 0xff;
        newBlue = copyGambarHasil & 0xff;

        System.out.println(" A > " +newAlpha+ " R > " +newRed+ " G > " +newGreen+ " B > " +newBlue);

        Biner bNewAlpha = new Biner(newAlpha);
        Biner bNewRed = new Biner(newRed);
        Biner bNewGreen = new Biner(newGreen);
        Biner bNewBlue = new Biner(newBlue);

        int []hasilNewAlpha = bNewAlpha.getBiner();
        System.out.print("Biner Alpha : " );
        for(int i=0 ; i<8 ; i++)
        {
            System.out.print(hasilNewAlpha[i]);
```

Figure 17. Decode Zig-zag Process

Decode process will require an image. New buffered image will be needed to generating the watermark as an image. The step of decode process is read the image with zig-zag matrix and break image into pixels. From the pixels, binary value can be established and then read the LSB of the binary value. From a set of bits read, a pixel can be formed. From some pixels, image can be reshaped with the help of buffered image.

- **Decode Sequence**

```
public DecodeSequence(String pGambar)
{
    try
    {
        File input = new File(pGambar);
        gambarHasil = ImageIO.read(input);
        widthHasil = gambarHasil.getWidth();
        heightHasil = gambarHasil.getHeight();
        System.out.println("Width : " + widthHasil + " Height : " + heightHasil);

        for(int i=0 ; i<2 ; i++)
        {
            if(i==1)
            {
                for(int j=0 ; j<16 ; j++)
                {
                    System.out.println("I = " +i+ " J = " +j);

                    int copyCover = gambarHasil.getRGB(i,j);

                    newAlpha = (copyCover>>24) & 0xff;
                    newRed = (copyCover>>16) & 0xff;
                    newGreen = (copyCover>>8) & 0xff;
                    newBlue = copyCover & 0xff;

                    System.out.println(" A > " +newAlpha+ " R > " +newRed+ " G > " +newGreen+ " B > " +newBlue);
```

Figure 18. Decode Sequence Process

Decode process using sequence pattern is like decode process using zig-zag pattern. This decode process using sequence pattern to read data per pixels and save the LSB value and save into the new buffered image until the message image is generated.

- **Zig-zag Pattern**

```
Counter : 1
0 0 0
0 0 1
Counter : 2
0 0 0
0 1 1
Counter : 3
0 0 1
0 1 1
Counter : 4
0 1 1
0 1 1
Counter : 5
0 1 1
1 1 1
Counter : 6
1 1 1
1 1 1
```

Figure 19. Zig-zag Pattern

Zig-zag pattern is used to read for encode and decode process. From the image above, there is a matrix with 2 rows and 3 columns. Zig-zag pattern read from bottom right matrix coordinate [1,2] at the first step and move to the left [1,1] next step move to the right and up to the coordinate [0,2]. When move into top right of the matrix, the program will move into the left into coordinate [0,1] and the move into left down to the coordinate [1,0] and last into coordinate [0,0]. The main concept of zig-zag pattern is move from bottom right until the top left with zig-zag. It will start from bottom right and then move to the left, then move to the up right, the rules is when it moves to the first row it must move to the left for next step and diagonally down to the left. When it moves to the first column, for the next step it must up and for the next step diagonally up to the right and for the final, it will move to the end of the matrix.

- **Sequence Pattern**

```
Counter : 1
1 0 0
0 0 0
Counter : 2
1 0 0
1 0 0
Counter : 3
1 1 0
1 0 0
Counter : 4
1 1 0
1 1 0
Counter : 5
1 1 1
1 1 0
Counter : 6
1 1 1
1 1 1
```

Figure 20. Sequence Pattern

Sequence pattern is used to read for encode and decode process. From the image above, there is a matrix with 2 rows and 3 columns. Sequence pattern read from first column and first row [0,0] and move into the first column and second row [0,1] and then to the second column and first row [1,0] and so on. The main point is it moves from first column and first row until the end of row then move to the second column and first row until the end of row and so on until the end of column and the end of row.

5.2 Testing

Cover Image :

Figure 21. Cover Image

Size : Width = 1920 px
Height = 1080 px
Storage Capacity : 259.200 pixels

Source :
http://vignette1.wikia.nocookie.net/swordartonline/images/3/38/ISL_Ragnarok_-_Richie's_mountain.png/revision/latest?cb=20140823191311

5.2.1 Zig-zag Encode and Decode

Table 5. Zig-zag Encode and Decode Testing

Message Image	Status	Size (kB)	Encode Process Time (second)	Size Result Image Size (kB)	Decode Process Time (second)	Size Message Image after Decode (kB)
 500 x 282 Source : http://vignette2.wikia.nocookie.net/adventuretimewithfinnandjake/images/8/81/Memories_of_Boom_Boom_Mountain.png/revision/latest?cb=20091211025738	Success	241,4	660	3477,2	230	318,0
 250 x 250 Source : http://3.bp.blogspot.com/-JBDaPgtUjTo/U0zcyY46JyI/AAAAAAAAAcMQ/dGksvQ9orXY/s250-p/Doraemon.png	Success	55,4	327	2793,5	136	52,8

Message Image	Status	Size (kB)	Encode Process Time (second)	Size Result Image Size (kB)	Decode Process Time (second)	Size Message Image after Decode (kB)
 1024 x 1024 Source : http://www.freeiconspng.com/uploads/mustache-by-hurricamo-on-deviantart-10.png	Fail	15,2	-	-	-	-
 375 x 456 Source : http://nagybarbara.ingyenweb.hu/Hello-kitty-2.png	Success	71,7	847	3096	374	55,5
 50 x 50 Source : http://cdn.iconsmash.com/Content/icons/halloween_by_arrioch-d31udde/iconpreviews/128/dracula.png	Success	6,9	54	2602,9	5	5,4

Cover Image has maximum storage capacity of message image until 259.200 pixels. Four of five tested images are successfully encoded into the cover image, but one of them is failed and caused by its size is too big. The message image which failed to encoded has 1.024 width and 1.024 height, it contains 1.048.576 pixels so it can not be encoded because it exceeds the storage capacity of cover image. More pixels to encoded more time to process needed. Time running during the decode process is faster than the encode process. Size of the cover image after encode process and size message image after decode process is changed.

5.2.2 Sequence Encode and Decode

Table 6. Sequence Encode and Decode Testing

Message Image	Status	Size (kB)	Encode Process Time (second)	Size Result Image Size (kB)	Decode Process Time (second)	Size Message Image after Decode (kB)
 500 x 282 Source : http://vignette2.wikia.nocookie.net/adventuretimewithfinnandjake/images/8/81/Memories_of_Boom_Boom_Mountain.png/revision/latest?cb=20091211025738	Success	241,4	600	3549,9	135	318,0
 250 x 250 Source : http://3.bp.blogspot.com/-JBDaPgtUjTo/U0zcyY46JyI/AAAAAAAAcMQ/dGksvQ9orXY/s250-p/Doraemon.png	Success	55,4	313	2802,9	62	52,8

Message Image	Status	Size (kB)	Encode Process Time (second)	Size Result Image Size (kB)	Decode Process Time (second)	Size Message Image after Decode (kB)
 1024 x 1024 Source : http://www.freeiconspng.com/uploads/mustache-by-hurricamo-on-deviantart-10.png	Fail	15,2	-	-	-	-
 375 x 456 Source : http://nagybarbara.ingyenweb.hu/Hello-kitty-2.png	Success	71,7	859	3047,1	410	55,5
 50 x 50 Source : http://cdn.iconsmash.com/Content/icons/halloween_by_arrioch-d31udde/iconpreviews/128/dracula.png	Success	6,9	43	2610,3	3	5,4

Cover Image has maximum storage capacity of message image until 259.200 pixels. Same as encode process using zig-zag pattern, four of five tested image is successfully encoded into the cover image, but one of them is failed and caused by its size is too big. It contains 1.048.576 pixels and exceeds the storage capacity of cover image. More pixels to encoded more time to process needed. Time running during the decode process is faster than the encode process. Size of the cover image after encode process and size message image after decode process is changed.

5.2.3 Explanation

From the two testing tables above, five message images were tested to be encoded into one cover image. Four of them are successfully encoded into cover image, but one of them is failed because its size is too big for cover image. The cover image only can save 259.200 pixels, but the failed message image needs 1.048.576 pixels, it exceeds the storage capacity of cover image.

File size of message images from original message image and from decoding process is different, but from testing which successfully have been done to message images, decoding process using zig-zag pattern or sequence pattern have no file size difference.

Zig-zag pattern needs more time to do encode and decode process. The status in the tables is the information of the encoding process, success if encode process successfully and fail if encode process is failed. If there is an error caused by size of message image too big, the program will alert the user and show what caused the error.

Time running and file size using both pattern is different. More time is needed in the encode and decode process using zig-zag pattern to encode and decode image. The file size of encode process using zig zag pattern changes only slightly when compared to sequence pattern, but in the decode process, file size of message image which generated from the process has the same size.