CHAPTER 6: System Implementation

System implementation is a set of activities that is used to create this application as an operational system. It has been done by translating the application’s requirements and system design into programming codes then developing and testing the classes in the system. System implementation utilizes system design since its main reference is to accomplish the functions and the application designs. The process involved in system implementation integrates the physical and conceptual resources, output and limitations.

6.1 Development Environment

It is very important to choose the right environmental development to be used, as it will affect the final outcome for the system deployment. In this thesis, proper combinations of hardware and software are ensured as to make the development process of final outcome will be more efficient. Vice versa, quality of a system is affected if a wrong combination of hardware and software was chosen. Listed below is a short discussion of the system requirement regarding to the development of the application.

6.2 System Requirement

The details for system requirement have been thoroughly discussed in the previous chapter; CHAPTER 4: SYSTEM ANALYSIS, under the subtopic Hardware Requirement and Software Requirement. Briefly, the hardware requirement and software requirement for the system will be explained in general as below.
6.2.1 Hardware Requirement

- Personal computer (include other device like keyboard, mouse, etc)

6.2.2 Software Requirement

- Microsoft® Windows XP with Service Pack 2
- Netbeans 6.7.1

6.3 Application Development

Application programming codes are designed using NetBeans IDE 6.7.1 which utilizes Java programming language. With the edition of the Java SE package which can be downloaded from NetBeans webpage, it will much easier to create an application using Java. Below is the overview of NetBeans IDE 6.7.1 workspace.
Every programming language has to be learned from basic. In Java development there are templates that we can use in order to start an application. Using the template, it makes it easier for the language to be programmed.

```
/* To change this template, choose Tools | Templates and open the template in the editor. */

package javaapplication1;

/**
 * @author fais
 */

public class Main {

    /**
     * @param args the command line arguments
     */

    public static void main(String[] args) {
        // TODO code application logic here
    }
}
```

Figure 6.2 Basic Java application template

### 6.4 System Coding

This section shows the fragment of the most important coding in each class. These coding is very important in order the application to run properly.

#### 6.4.1 Jumble.java Class

This class is used to first flipping the image and jumbles the blocks of the selected image. Figure 6.3 below shows coding that is scattered the image 'bi' into four rectangular areas and draws each of these areas in to a different part of the image. It is also to jumble up the image. ‘cell’ is an array which has been populated with values which redirect drawing of one subarea to another sub area.

```
int cellWidth = bi.getWidth(null)/2;
```
int cellHeight = bi.getHeight(null)/2;
for (int x=0; x<2; x++) {
    int sx = x*cellWidth;
    for (int y=0; y<2; y++) {
        int sy = y*cellHeight;
        int cell = cells[x*2+y];
        int dx = (cell / 2) * cellWidth;
        int dy = (cell % 2) * cellHeight;
        g.drawImage(bi,
                    dx, dy, dx+cellWidth, dy+cellHeight,
                    sx, sy, sx+cellWidth, sy+cellHeight,
                    null);
    }
}

Figure 6.3 Image Jumbling Coding

Based on Figure 6.3;

- bi - the specified image to be drawn. This method does nothing if img is null.
- dx (dstx1) - the x coordinate of the first corner of the destination rectangle.
- dy (dsty1) - the y coordinate of the first corner of the destination rectangle.
- dx+cw (dstx2) - the x coordinate of the second corner of the destination rectangle.
- dy+ch (dsty2) - the y coordinate of the second corner of the destination rectangle.
- sx (srcx1) - the x coordinate of the first corner of the source rectangle.
- sy (srcy1) - the y coordinate of the first corner of the source rectangle.
- sx+cw (srcx2) - the x coordinate of the second corner of the source rectangle.
- sy+ch (srcy2) - the y coordinate of the second corner of the source rectangle.
- null - object to be notified as more of the image is scaled and converted.
Figure 6.4 below show the flipping coding that flip the image where the reflective image is generated.

```java
BufferedImage bufferedImage = ba;
Graphics2D g2d = (Graphics2D)bufferedImage.getGraphics();

Graphics gb = bufferedImage.getGraphics();
gb.drawImage(bufferedImage, 0, 0, null);
gb.dispose();

AffineTransform tx = AffineTransform.getScaleInstance(-1, 1);
    tx.translate(-bufferedImage.getWidth(null), 0);
AffineTransformOp op = new AffineTransformOp(tx,
    AffineTransformOp.TYPE_NEAREST_NEIGHBOR);
bufferedImage = op.filter(bufferedImage, null);

    g2d.drawImage(bufferedImage, null, 300, 10);
return bufferedImage;
```

Figure 6.4 Image Flipping Coding

6.4.2 MD5.java Class

This class is a Java implementation of the RSA Data Security, Inc. of MD5 Message Digest Algorithm, as defined in RFC 1321. This class is freely available and can be downloaded from the Internet.
6.4.3 HashHandler.java Class

Figure 6.5 shows the coding used to embedding the hash calculated in the End-Of-File(EOF). The comment (//) states the functions of those coding after it.

```java
//converting string output to byte buf
byte buf[] = output.getBytes();
ByteArrayOutputStream f = new ByteArrayOutputStream();

//writing buf to ByteArrayOutputStream of f
f.write(buf);

//b = the current contents of this output stream, as a byte array.
byte b[] = f.toByteArray();

int c;
while ((c = in.read()) != -1) {
    //copying from in to out until EOF
    out.write(c);
}

//writes the complete contents of this byte array output stream to
//the specified output stream argument,
f.writeTo(out);
```

Figure 6.5 Embedding Hash in Image
6.4.4 FrontPage.java Class

This class is the main menu of the program that connects all the other classes for the application to work. It provides many functions to interact with the user. Figure 6.6 below show the code to acquire an image and display the preview on the application.

```java
try {
    BufferedImage img = ImageIO.read(new File(FileLocation.getText()));
    bi = img;
    ImageIcon iconA = new ImageIcon(img);
    JLabel3.setIcon(iconA);
} catch (IOException e) {
}
```

Figure 6.6 Acquire an Image and Set The Preview

Figure 6.7 show the creation of MD5 object and generation of hash from the password entered by the user.

```java
MD5 hash = new MD5();
String passcompare = hash.calcMD5(password1.getText());
```

Figure 6.7 Hash Generation

Figure 6.8 show the process of creation on HashHandler object and the embedding process into the picture.

```java
HashHandler a = new HashHandler();
a.embedHash(FileChooser.getSelectedFile().getPath(), password1.getText());
```

Figure 6.8 Hash Saving Into The Image