

Code Listing

<Fingerprint Recognition>

Supervisor: Nigel Whyte

Student: Dayu Chen

Student ID: C00131022

Submit date: 2010-4-11

Content

[1. DataStructrue part 1](#_Toc290215413)

[1.1 InputImage.java 1](#_Toc290215414)

[1.2 TemplateImage.java 1](#_Toc290215415)

[1.3 DBOriginalImage.java 2](#_Toc290215416)

[1.5 Pixel.java 3](#_Toc290215417)

[1.6 Point.java 5](#_Toc290215418)

[1.7 Minutia.java 6](#_Toc290215419)

[1.8 Featrue.java 7](#_Toc290215420)

[1.9 FingerprintImage.java 8](#_Toc290215421)

[2. Algorithm part 10](#_Toc290215422)

[2.1 Normalization.java 10](#_Toc290215423)

[2.2 OrientationEstimation.java 11](#_Toc290215424)

[2.3 EdgeDetection.java 14](#_Toc290215425)

[2.4 OGorman.java 16](#_Toc290215426)

[2.5 RidgeDetection.java 19](#_Toc290215427)

[2.6 Gaussian.java 21](#_Toc290215428)

[2.7 FixRidge.java 22](#_Toc290215429)

[2.8 Thinning.java 25](#_Toc290215430)

[2.9 MinutiaeExtraction.java 30](#_Toc290215431)

[2.10 RemoveFalseMinutiae.java 31](#_Toc290215432)

[2.11 PointDirection.java 39](#_Toc290215433)

[2.12 FindCore.java 44](#_Toc290215434)

[2.13 GenerateFeatrue.java 47](#_Toc290215435)

[2.14 Match.java 49](#_Toc290215436)

[3. Database access part 55](#_Toc290215437)

[3.1 DAO.java 55](#_Toc290215438)

[3.2 FeatrueDAO.java 59](#_Toc290215439)

[3.3 FingerprintImageDAO.java 63](#_Toc290215440)

[3.4 MatchDB.java 65](#_Toc290215441)

[4. GUI Frame 68](#_Toc290215442)

[4.1 FRSMainApp.java 68](#_Toc290215443)

[4.2 ResultFrame.java 95](#_Toc290215444)

[4.2 ResultWindow.java 99](#_Toc290215445)

[4.4 DBOriginalImagePanel.java 104](#_Toc290215446)

[4.5 DBProcessedImagePanel.java 105](#_Toc290215447)

[4.6 InputImagePanel.java 106](#_Toc290215448)

[4.7 TemplateImagePanel.java 108](#_Toc290215449)

[4.8 ResultImagePanel.java 110](#_Toc290215450)

# DataStructrue part

## 1.1 InputImage.java

package demo;

import java.awt.image.BufferedImage;

import java.util.ArrayList;

/\*\*InputImage structrue is for drawing image in input image panel

 \*/

public class InputImage {

 public static BufferedImage original\_image = null;

 public static BufferedImage normalized\_image = null;

 public static BufferedImage enhanced\_image = null;

 public static BufferedImage binarization\_image = null;

 public static BufferedImage thinning\_image = null;

 public static ArrayList<Minutia> minutiae = new ArrayList();

 public static Point core = null;

 public static ArrayList<Minutia> same = new ArrayList();

}

## 1.2 TemplateImage.java

package demo;

import java.awt.image.BufferedImage;

import java.util.ArrayList;

/\*\* TemplateImage structrue is for drawing image in template image panel

 \*/

public class TemplateImage {

 public static BufferedImage original\_image = null;

 public static BufferedImage normalized\_image = null;

 public static BufferedImage enhanced\_image = null;

 public static BufferedImage binarization\_image = null;

 public static BufferedImage thinning\_image = null;

 public static ArrayList<Minutia> minutiae = new ArrayList();

 public static Point core = null;

 public static ArrayList<Minutia> same = new ArrayList();

}

## 1.3 DBOriginalImage.java

package demo;

import java.awt.image.BufferedImage;

/\*\*this is for drawing image in DataBase function panel

 \*/

public class DBOriginalImage {

 public static BufferedImage original\_image = null;

}

1.4 DBProcessedImage.java

package demo;

import java.awt.image.BufferedImage;

import java.util.ArrayList;

/\*\*This class is for drawing the processed image in database function panel

 \*/

public class DBProcessedImage {

 public static BufferedImage original\_image = null;

 public static BufferedImage normalized\_image = null;

 public static BufferedImage enhanced\_image = null;

 public static BufferedImage binarization\_image = null;

 public static BufferedImage thinning\_image = null;

 public static ArrayList<Minutia> minutiae = new ArrayList();

 public static Point core = null;

}

## 1.5 Pixel.java

package demo;

/\*\*Pixel Structrue express pixel in image

 \*/

public class Pixel {

 private int x,y;

 private int rgb;

 private double gx,gy;//gradientx and gradient y

 private double angle = 0;

 private int mask = 0;//mark binarization

 private int type = 0;//mark type 1 is ending 2 is bifurcation

 private int check = 0;// mark this pixel has not be checked

 public Pixel(int i, int j){

 this.x = i;

 this.y = j;

 }

 public void setRGB(int r){

 this.rgb = r;

 }

 public void setGX(double g){

 this.gx = g;

 }

 public void setGY(double g){

 this.gy = g;

 }

 public void setAngle(double a){

 this.angle = a;

 }

 public void setMask(int i){

 this.mask = i;

 }

 public void setType(int i){

 this.type = i;

 }

 public void setCheck(int i){

 this.check = i;

 }

 public int getCheck(){

 return check;

 }

 public int getType(){

 return type;

 }

 public int getX(){

 return x;

 }

 public int getY(){

 return y;

 }

 public double getGX(){

 return gx;

 }

 public double getGY(){

 return gy;

 }

 public int getRGB(){

 return rgb;

 }

 public double getAngle(){

 return angle;

 }

 public int getMask(){

 return mask;

 }

}

## 1.6 Point.java

package demo;

/\*\*This structrue is for express pixel in point direction algorithm and core point

 \* detection algorithm, in order todistinguish the block orientation and point orientation

 \*/

public class Point {

 int x, y;

 int d;

 int rgb;

 public Point(int x,int y,int r){

 this.x=x;

 this.y = y;

 this.rgb =r;

 }

 public void setD(int d){

 this.d= d;

 }

 public void setRGB(int r){

 this.rgb = r;

 }

 public int getX(){

 return x;

 }

 public int getY(){

 return y;

 }

 public int getRGB(){

 return rgb;

 }

 public int getDirection(){

 return d;

 }

}

## 1.7 Minutia.java

package demo;

/\*\*This class is express minutia structrue

 \*/

public class Minutia {

 private double angle;

 private int x,y;

 private int type;

 private boolean mark = true; // mark minutia is not a false minutia

 public Minutia(int i, int j, int t){

 this.x = i;

 this.y = j;

 this.type = t;

 }

 public void setAngle(double theta){

 this.angle = theta;

 }

 public void setMark(){

 this.mark = false;

 }

 public double getAngle(){

 return angle;

 }

 public int getX(){

 return x;

 }

 public int getY(){

 return y;

 }

 public int getType(){

 return type;

 }

 public boolean getMark(){

 return mark;

 }

}

## 1.8 Featrue.java

package demo;

/\*\*This is the featrue vector structrue, it is used in match algorithm

 \*/

public class Featrue {

 private Minutia m;

 private int direction;

 private double angle,diffAngle;

 private int type;

 private boolean isMatch = false;

 private String image = null;

 private int id = 0; // id in database

 public Featrue(Minutia mi,int d,double a,int t,double diffA){

 this.m=mi;

 this.direction=d;

 this.angle=a;

 this.type =t;

 this.diffAngle = diffA;

 }

 public void setMatch(boolean b){

 this.isMatch = b;

 }

 public void setImage(String i){

 this.image = i;

 }

 public void setID(int i){

 this.id = i;

 }

 public int getID(){

 return id;

 }

 public String getImage(){

 return image;

 }

 public double getDifferentAngle(){

 return diffAngle;

 }

 public boolean getMatch(){

 return isMatch;

 }

 public Minutia getMinutia(){

 return m;

 }

 public int getDirection(){

 return direction;

 }

 public double getAngle(){

 return angle;

 }

 public int getType(){

 return type;

 }

}

## 1.9 FingerprintImage.java

package demo;

/\*\*FingerprintImage structure is for recording the fingerprint image into database

 \*/

public class FingerprintImage {

 private String image\_id;

 private String image\_directory;

 private String description;

 private int core\_x,core\_y;

 private double core\_angle;

 public FingerprintImage(String id, String d,String de, int x, int y, double angle){

 this.image\_id = id;

 this.image\_directory =d;

 this.description = de;

 this.core\_x = x;

 this.core\_y = y;

 this.core\_angle = angle;

 }

 public String getID(){

 return image\_id;

 }

 public String getDirectory(){

 return image\_directory;

 }

 public String getDescription(){

 return description;

 }

 public int getCoreX(){

 return core\_x;

 }

 public int getCoreY(){

 return core\_y;

 }

 public double getCoreAngle(){

 return core\_angle;

 }

}

# Algorithm part

## 2.1 Normalization.java

package demo;

import java.awt.image.BufferedImage;

/\*\*This class is for normalization algorithm, it include calculate the mean gray

 \* and variance, and normalizaed the input image into the accept gray and variance

 \*/

public class Normalization {

 private BufferedImage input = null;

 private static final int M0 = 100; // accept gray level

 private static final int Var0 = 150;// accept variance

 public Normalization(BufferedImage bi){

 this.input = bi;

 }

 private int getAverageGray(){

 int m = 0;

 int sum = 0;

 for(int i=0; i<input.getWidth();i++){

 for(int j=0;j<input.getHeight();j++){

 int rgb = 0;

 rgb = input.getRGB(i, j);

 rgb = (rgb & 0x000000ff);

 sum = sum +rgb;

 }

 }

 m=sum/(input.getWidth()\*input.getHeight());

 return m;

 }

 private int getAverageVariance(){

 int var = 0;

 int sum = 0;

 int gray = getAverageGray();

 for(int i=0; i<input.getWidth();i++){

 for(int j=0;j<input.getHeight();j++){

 int rgb = 0;

 rgb = input.getRGB(i, j);

 rgb = (rgb & 0x000000ff);

 sum = sum +(rgb-gray)\*(rgb-gray);

 }

 }

 var = sum/(input.getWidth()\*input.getHeight());

 return var;

 }

 public BufferedImage getNormalizedImage(){

 int gray = getAverageGray();

 int variance = getAverageVariance();

 for(int i=0; i<input.getWidth();i++){

 for(int j=0;j<input.getHeight();j++){

 int rgb = 0;

 rgb = input.getRGB(i, j);

 rgb = (rgb & 0x000000ff);

 if(rgb>gray){

 rgb = M0+(int)(Math.sqrt((Var0\*((rgb-gray)\*(rgb-gray)))/variance));

 }else {

 rgb = M0-(int)(Math.sqrt((Var0\*((rgb-gray)\*(rgb-gray)))/variance));

 }

 input.setRGB(i, j, (rgb\*256\*256)+(rgb\*256)+rgb);

 }

 }

 return input;

 }

}

## 2.2 OrientationEstimation.java

package demo;

import java.awt.image.BufferedImage;

/\*\*This class is for orientation estimation. It includes function to comput

 \* gradient x and gradien y for each pixel using Soble edge detector and then

 \* devide image into several blocks (size is 11\*11) to comput the orientation

 \*/

public class OrientationEstimation {

 private BufferedImage input;

 private Pixel[][] p ;

 public OrientationEstimation(BufferedImage bi){

 this.input = bi;

 initPixelArray();

 }

/\* this class also will return a Pixel matrix for the next algorithm, this

 function is initialize a pixel matrix

 \*/

 private void initPixelArray(){

 p = new Pixel[input.getWidth()][input.getHeight()];

 for(int i = 0; i<input.getWidth();i++){

 for(int j = 0;j<input.getHeight();j++){

 Pixel pi = new Pixel(i,j);

 pi.setRGB(input.getRGB(i, j)&0x000000ff);

 p[i][j] = pi;

 }

 }

 for(int i = 1; i<input.getWidth()-1;i++){

 for(int j = 1;j<input.getHeight()-1;j++){

 double gx = calculateXGraient(i,j);

 double gy = calculateYGraient(i,j);

 p[i][j].setGX(gx);

 p[i][j].setGY(gy);

 }

 }

 }

 // calculate the x grandient

 private double calculateXGraient(int i, int j){

 double gx = 0;

 gx = p[i+1][j-1].getRGB() + 2\*p[i+1][j].getRGB() + p[i+1][j+1].getRGB()-p[i-1][j-1].getRGB() - 2\*p[i-1][j].getRGB() - p[i-1][j+1].getRGB();

 return gx;

 }

 // calculate the y grandient

 private double calculateYGraient(int i, int j){

 double gy = 0;

 gy = p[i-1][j+1].getRGB() + 2\*p[i][j+1].getRGB() + p[i+1][j+1].getRGB()-p[i-1][j-1].getRGB() - 2\*p[i][j-1].getRGB() - p[i+1][j-1].getRGB();

 return gy;

 }

 // find out the 11X11 block which centre is in pixel(i, j)

 private Pixel[][] getNeighbourMask(int i, int j){

 Pixel[][] neighbour = new Pixel[11][11];

 int x = 0;

 for(int m = -5;m<6;m++){

 int y = 0;

 for(int n = -5; n<6; n++){

 neighbour[x][y] = p[i+m][j+n];

 y = y +1;

 }

 x = x+1;

 }

 return neighbour;

 }

 // calculate the angle for this block

 private double getBlockAngle(Pixel[][] pixel){

 double angle = 0;

 double sum\_1 = 0;

 double sum\_2 = 0;

 for(int i = 0;i<11;i++){

 for(int j = 0; j<11;j++){

 sum\_1 = sum\_1 + 2\*pixel[i][j].getGX()\*pixel[i][j].getGY();

 sum\_2 = sum\_2 + (Math.pow(pixel[i][j].getGX(), 2)-Math.pow(pixel[i][j].getGY(), 2));

 }

 }

 //range of arctan is [-PI/4,PI/4], need to convert it into [0, PI]

 if(sum\_2 ==0){

 angle = 0;

 }else{

 angle = Math.atan(sum\_1/sum\_2)\*0.5;

 }

 if(sum\_2>0){

 angle = angle + Math.PI/2;

 }

 if(sum\_2<0&&sum\_1>0){

 angle = angle + Math.PI;

 }

 return angle;

 }

 private void setAngle(int i , int j, double angle){

 for(int m = -5; m <6; m++){

 for(int n = -5; n<6; n++){

 p[i+m][j+n].setAngle(angle);

 }

 }

 }

 public Pixel[][] getOrientationField(){

 int y = 6;

 while(y+5<input.getHeight()-1){

 int x = 6;

 while(x+5<input.getWidth()-1){

 Pixel[][] n = getNeighbourMask(x,y);

 double angle = getBlockAngle(n);

 setAngle(x,y,angle);

 x = x+11;

 }

 y = y+11;

 }

 return p;

 }

}

## 2.3 EdgeDetection.java

package demo;

import java.awt.image.BufferedImage;

/\*\*This Class is for image enhancment.

\*/

public class EdgeDetection {

 private Pixel[][] p;

 private BufferedImage input;

 EdgeDetection(BufferedImage bi,Pixel[][] pixel){

 this.input = bi;

 this.p = pixel;

 }

 private int getGray(int i, int j){

 int gray=0;

 double angle = p[i][j].getAngle();

 OGorman o = new OGorman();

 o.getOuputFilter(angle);

 int sum = 0;

 int dl = 0;

 for(int m = -3;m<4;m++){

 for(int n = -3;n<4;n++){

 int h = o.getFilter(m+3, n+3);

 int g = p[i+m][j+n].getRGB();

 sum = sum + h \* g;

 dl = dl+h;

 }

 }

 if(dl==0){

 gray = p[i][j].getRGB();

 }else{

 gray = sum/dl;

 }

 return gray;

 }

 public BufferedImage getEnhancementImage(){

BufferedImage output = new BufferedImage(input.getWidth(),input.getHeight(),BufferedImage.TYPE\_BYTE\_GRAY);

 for(int i = 0;i<output.getWidth();i++){

 for(int j = 0;j<output.getHeight();j++){

 output.setRGB(i, j, 0\*256\*256+0\*256+0);

 }

 }

 for(int i=3; i<output.getWidth()-6;i++){

 for(int j=3;j<output.getHeight()-3;j++){

 int new\_RGB = getGray(i, j);

 output.setRGB(i, j, (new\_RGB\*256\*256)+(new\_RGB\*256)+new\_RGB);

 }

 }

//update the pixel matrix for the next algorithm

 for(int i = 0; i<output.getWidth();i++){

 for(int j = 0; j<output.getHeight();j++){

 int rgb = output.getRGB(i, j)&0x000000ff;

 p[i][j].setRGB(rgb);

 }

 }

 return output;

 }

 public Pixel[][] getPixelMatrix(){

 return p;

 }

}

## 2.4 OGorman.java

package demo;

/\*\*This class is Generate a OGorman filter base on the input angle. This filter

 \* is for EdgeDetection.java

 \*/

public class OGorman {

 private int[][] o\_gorman\_filter;

 private int[][] output\_filter;

// initialize a OGorman filter which is in 0 degree direction

 public OGorman(){

 o\_gorman\_filter = new int[7][7];

 o\_gorman\_filter[0][0] = -10;

 o\_gorman\_filter[1][0] = -20;

 o\_gorman\_filter[2][0] = -30;

 o\_gorman\_filter[3][0] = -30;

 o\_gorman\_filter[4][0] = -30;

 o\_gorman\_filter[5][0] = -20;

 o\_gorman\_filter[6][0] = -10;

 o\_gorman\_filter[0][1] = 2;

 o\_gorman\_filter[1][1] = 4;

 o\_gorman\_filter[2][1] = 6;

 o\_gorman\_filter[3][1] = 6;

 o\_gorman\_filter[4][1] = 6;

 o\_gorman\_filter[5][1] = 4;

 o\_gorman\_filter[6][1] = 2;

 o\_gorman\_filter[0][2] = 4;

 o\_gorman\_filter[1][2] = 8;

 o\_gorman\_filter[2][2] = 12;

 o\_gorman\_filter[3][2] = 12;

 o\_gorman\_filter[4][2] = 12;

 o\_gorman\_filter[5][2] = 8;

 o\_gorman\_filter[6][2] = 4;

 o\_gorman\_filter[0][3] = 8;

 o\_gorman\_filter[1][3] = 16;

 o\_gorman\_filter[2][3] = 24;

 o\_gorman\_filter[3][3] = 24;

 o\_gorman\_filter[4][3] = 24;

 o\_gorman\_filter[5][3] = 16;

 o\_gorman\_filter[6][3] = 8;

 o\_gorman\_filter[0][4] = 4;

 o\_gorman\_filter[1][4] = 8;

 o\_gorman\_filter[2][4] = 12;

 o\_gorman\_filter[3][4] = 12;

 o\_gorman\_filter[4][4] = 12;

 o\_gorman\_filter[5][4] = 8;

 o\_gorman\_filter[6][4] = 4;

 o\_gorman\_filter[0][5] = 2;

 o\_gorman\_filter[1][5] = 4;

 o\_gorman\_filter[2][5] = 6;

 o\_gorman\_filter[3][5] = 6;

 o\_gorman\_filter[4][5] = 6;

 o\_gorman\_filter[5][5] = 4;

 o\_gorman\_filter[6][5] = 2;

 o\_gorman\_filter[0][6] = -10;

 o\_gorman\_filter[1][6] = -20;

 o\_gorman\_filter[2][6] = -30;

 o\_gorman\_filter[3][6] = -30;

 o\_gorman\_filter[4][6] = -30;

 o\_gorman\_filter[5][6] = -20;

 o\_gorman\_filter[6][6] = -10;

 }

 public void getOuputFilter(double angle){

 output\_filter = new int[7][7];

 for(int i = -3; i<4; i++){

 for(int j = -3; j<4; j++){

 output\_filter[i+3][j+3] = getRotateValue(angle,i,j);

 }

 }

 }

 private int getRotateValue(double angle, int i , int j){

 int n = 0;

 double new\_i = Math.cos(angle)\*i+Math.sin(angle)\*j;

 double new\_j = Math.cos(angle)\*j-Math.sin(angle)\*i;

 int it = (int)new\_i;

 int jt = (int)new\_j;

/\* in java Math.cos(PI/2) is not 0, so need to do some if..else to make sure

 \* the matrix is not outof boundary

 \*/

 if((new\_i==it)&&(new\_j==jt)){

 n = o\_gorman\_filter[it+3][jt+3];

 }else{

 if((it<=-3)||(it>=3)||(jt<=-3)||(jt>=3)){

 if((it == 3||it==-3)&&jt!=new\_j){

 n = o\_gorman\_filter[it+3][jt+3];

 }else if(it!=new\_i&&(jt==3||jt==-3)){

 n = o\_gorman\_filter[it+3][jt+3];

 }else{

 n = 0;

 }

 }else{

 int x0 = (int) new\_i;

 int y0 = (int) new\_j;

 int x1, y1;

 if(x0<0){

 x1 = x0-1;

 }else{

 x1 = x0+1;

 }

 if(y0<0){

 y1 = y0-1;

 }else{

 y1 = y0+1;

 }

 n = getValue(x0,y0,x1,y1,new\_i,new\_j);

 }

 }

 return n;

 }

// linear interpolation

 private int getValue(int x0, int y0, int x1, int y1,double x, double y){

 int v = 0;

 int a = o\_gorman\_filter[x0+3][y1+3];

 int b = o\_gorman\_filter[x1+3][y1+3];

 int d = o\_gorman\_filter[x0+3][y0+3];

 int c = o\_gorman\_filter[x1+3][y0+3];

 double e = a+(b-a)\*(x-x0);

 double f = d+(c-d)\*(x-x0);

 v = (int) (f + (e - f) \* (y - y0));

 return v;

 }

 public int getFilter(int i, int j){

 return output\_filter[i][j];

}

}

## 2.5 RidgeDetection.java

package demo;

import java.awt.image.BufferedImage;

/\*\*This class if for ridge detection

 \*/

public class RidgeDetection {

 private BufferedImage input;

 private Pixel[][] p;

 public RidgeDetection(BufferedImage bi, Pixel[][] pixel){

 this.input = bi;

 this.p = pixel;

 }

/\*compute the average gray using a 9\*9 matirx which the centre pixel is (i, j),

 \* using this average value as the threshold

 \*/

 private int getAverageGray(int i, int j){

 int averageGray = 0;

 int sum = 0;

 for(int m = -4;m<5;m++){

 for(int n = -4;n<5;n++){

 sum = sum+p[i+m][j+n].getRGB();

 }

 }

 averageGray = sum/81;

 return averageGray;

 }

 public BufferedImage getRidge(){

 BufferedImage output = new BufferedImage(input.getWidth(),input.getHeight(),BufferedImage.TYPE\_BYTE\_GRAY);

 for(int m = 0;m<output.getWidth();m++){

 for(int n = 0;n<output.getHeight();n++){

 output.setRGB(m, n, 255\*256\*256+255\*256+255);

 }

 }

 for(int i = 4;i<input.getWidth()-4;i++){

 for(int j = 4;j<input.getHeight()-4;j++){

 if(p[i][j].getRGB()<(getAverageGray(i,j))){

 output.setRGB(i, j, 0\*256\*256+0\*256+0);

 }else{

 output.setRGB(i, j, 255\*256\*256+255\*256+255);

 }

 }

 }

 for(int i = 0; i<output.getWidth();i++){

 for(int j = 0; j<output.getHeight();j++){

 int rgb = output.getRGB(i, j)&0x000000ff;

 p[i][j].setRGB(rgb);

 }

 }

 return output;

 }

 public Pixel[][] getBinaryMatrix(){

 for(int i = 0 ;i<input.getWidth();i++){

 for(int j = 0; j<input.getHeight();j++){

 if(p[i][j].getRGB()==0){

 p[i][j].setMask(1);

 }else{

 p[i][j].setMask(0);

 }

 }

 }

 return p;

 }

}

## 2.6 Gaussian.java

package demo;

import java.awt.image.BufferedImage;

/\*\*This class is generate a Gaussian smooth filter, it is a part of the fix ridge

 \* algorithm. I do Gaussian smoonthing after normalization

 \*/

public class Gaussian {

 private int[][] g;

 private BufferedImage input;

 // init a 3X3 Gaussian filter

 public Gaussian(BufferedImage bi){

 this.input = bi;

 g = new int[3][3];

 g[0][0] = 1;

 g[1][0] = 2;

 g[2][0] = 1;

 g[0][1] = 2;

 g[1][1] = 4;

 g[2][1] = 2;

 g[0][2] = 1;

 g[1][2] = 2;

 g[2][2] = 1;

 }

 public BufferedImage getSmoothImage(){

 BufferedImage output = new BufferedImage(input.getWidth(),input.getHeight(),BufferedImage.TYPE\_BYTE\_GRAY);

 for(int i = 0;i<output.getWidth();i++){

 for(int j = 0;j<output.getHeight();j++){

 output.setRGB(i, j, 0\*256\*256+0\*256+0);

 }

 }

 for(int i = 1; i<input.getWidth()-1;i++){

 for(int j =1 ;j<input.getHeight()-1;j++){

 int rgb = (input.getRGB(i-1, j-1)&0x000000ff)\*getFilter(0,0)+(input.getRGB(i, j-1)&0x000000ff)\*getFilter(1,0)+(input.getRGB(i+1, j-1)&0x000000ff)\*getFilter(2,0)

 +(input.getRGB(i-1, j)&0x000000ff)\*getFilter(0,1)+(input.getRGB(i, j)&0x000000ff)\*getFilter(1,1)+(input.getRGB(i+1, j)&0x000000ff)\*getFilter(2,1)

 +(input.getRGB(i-1, j+1)&0x000000ff)\*getFilter(0,2)+(input.getRGB(i, j+1)&0x000000ff)\*getFilter(1,2)+(input.getRGB(i+1, j+1)&0x000000ff)\*getFilter(2,2);

 rgb = rgb/16;

 output.setRGB(i, j, rgb\*256\*256+rgb\*256+rgb);

 }

 }

 return output;

 }

 private int getFilter(int i, int j){

 return g[i][j];

 }

}

## 2.7 FixRidge.java

package demo;

import java.awt.image.BufferedImage;

import java.util.ArrayList;

/\*\*This class is for fix the ridge to remove some hold and dot

 \*/

public class FixRidge {

 private BufferedImage input;

 private Pixel[][] p;

 public FixRidge(BufferedImage bi, Pixel[][] pixel){

 this.input = bi;

 this.p = pixel;

 }

 private ArrayList removePoint(){

 ArrayList remove = new ArrayList();

 for(int i = 1; i<input.getWidth()-1;i++){

 for(int j = 1; j<input.getHeight()-1;j++){

 if(p[i][j].getMask()==1){

 int count = 0;

 count = (p[i][j-1].getMask()+p[i+1][j-1].getMask()+p[i+1][j].getMask())

 \*(p[i][j+1].getMask()+p[i-1][j+1].getMask()+p[i-1][j].getMask())

 +(p[i+1][j].getMask()+p[i+1][j+1].getMask()+p[i][j+1].getMask())

 \*(p[i-1][j].getMask()+p[i-1][j-1].getMask()+p[i][j-1].getMask());

 if(count ==0){

 remove.add(i);

 remove.add(j);

 }

 }

 }

 }

 return remove;

 }

 private ArrayList fillPoint(){

 ArrayList fill = new ArrayList();

 for(int i = 1; i<input.getWidth()-1;i++){

 for(int j = 1; j<input.getHeight()-1;j++){

 if(p[i][j].getMask()==0){

 int count = 0;

 if(p[i][j-1].getMask()==1){

 count++;

 }

 if(p[i+1][j].getMask()==1){

 count++;

 }

 if(p[i][j+1].getMask()==1){

 count++;

 }

 if(p[i-1][j].getMask()==1){

 count++;

 }

 if(count>=3){

 fill.add(i);

 fill.add(j);

 }

 }

 }

 }

 return fill;

 }

 public BufferedImage getFixRidgeImage(){

 for(int m = 0;m<4;m++){

 ArrayList remove = removePoint();

 ArrayList fill = fillPoint();

 for(int i = 0; i<remove.size();i=i+2){

 int x = Integer.parseInt(remove.get(i).toString());

 int y = Integer.parseInt(remove.get(i+1).toString());

 input.setRGB(x, y, 255\*256\*256+255\*256+255);

 p[x][y].setRGB(255);

 p[x][y].setMask(0);

 }

 for(int i = 0; i<fill.size();i=i+2){

 int x = Integer.parseInt(fill.get(i).toString());

 int y = Integer.parseInt(fill.get(i+1).toString());

 input.setRGB(x, y, 0\*256\*256+0\*256+0);

 p[x][y].setRGB(0);

 p[x][y].setMask(1);

 }

 }

 return input;

 }

 public Pixel[][] getBinaryMatrix(){

 return p;

 }

}

## 2.8 Thinning.java

package demo;

import java.awt.image.BufferedImage;

import java.util.ArrayList;

/\*\*This class implement the thinning algorithm

 \*/

public class Thinning {

 private BufferedImage input;

 private Pixel[][] p ;

 public Thinning(BufferedImage bi, Pixel[][] pixel){

 this.input = bi;

 this.p = pixel;

 }

 private boolean conditionA(Pixel[][] neighbour){

 boolean b = false;

 int count = neighbour[0][0].getMask() + neighbour[1][0].getMask() + neighbour[2][0].getMask()

 + neighbour[0][1].getMask() + neighbour[2][1].getMask()

 + neighbour[0][2].getMask() + neighbour[1][2].getMask() + neighbour[2][2].getMask();

 if( count>=2&&count<=6){

 b = true;

 }

 return b;

 }

 private boolean conditionB(Pixel[][] neighbour){

 boolean b = false;

 int count = 0;

 if(neighbour[1][0].getMask()==0&&neighbour[2][0].getMask()==1){

 count = count + 1;

 }

 if(neighbour[2][0].getMask()==0&&neighbour[2][1].getMask()==1){

 count = count + 1;

 }

 if(neighbour[2][1].getMask()==0&&neighbour[2][2].getMask()==1){

 count = count + 1;

 }

 if(neighbour[2][2].getMask()==0&&neighbour[1][2].getMask()==1){

 count = count + 1;

 }

 if(neighbour[1][2].getMask()==0&&neighbour[0][2].getMask()==1){

 count = count + 1;

 }

 if(neighbour[0][2].getMask()==0&&neighbour[0][1].getMask()==1){

 count = count + 1;

 }

 if(neighbour[0][1].getMask()==0&&neighbour[0][0].getMask()==1){

 count = count + 1;

 }

 if(neighbour[0][0].getMask()==0&&neighbour[1][0].getMask()==1){

 count = count + 1;

 }

 if(count==1){

 b = true;

 }

 return b;

 }

 private boolean conditionC(Pixel[][] neighbour){

 boolean b = false;

 int r = neighbour[1][0].getMask()\*neighbour[2][1].getMask()\*neighbour[1][2].getMask();

 if(r == 0){

 b = true;

 }

 return b;

 }

 private boolean conditionD(Pixel[][] neighbour){

 boolean b = false;

 int r = neighbour[0][1].getMask()\*neighbour[2][1].getMask()\*neighbour[1][2].getMask();

 if(r == 0){

 b = true;

 }

 return b;

 }

 private boolean conditionCTwo(Pixel[][] neighbour){

 boolean b = false;

 int r = neighbour[1][0].getMask()\*neighbour[2][1].getMask()\*neighbour[0][1].getMask();

 if(r == 0){

 b = true;

 }

 return b;

 }

 private boolean conditionDTwo(Pixel[][] neighbour){

 boolean b = false;

 int r = neighbour[1][0].getMask()\*neighbour[0][1].getMask()\*neighbour[1][2].getMask();

 if(r == 0){

 b = true;

 }

 return b;

 }

 private Pixel[][] getNeighourMask(int i, int j){

 Pixel[][] neighbour = new Pixel[3][3];

 neighbour[0][0] = p[i-1][j-1];

 neighbour[1][0] = p[i][j-1];

 neighbour[2][0] = p[i+1][j-1];

 neighbour[0][1] = p[i-1][j];

 neighbour[1][1] = p[i][j];

 neighbour[2][1] = p[i+1][j];

 neighbour[0][2] = p[i-1][j+1];

 neighbour[1][2] = p[i][j+1];

 neighbour[2][2] = p[i+1][j+1];

 return neighbour;

 }

 private ArrayList subIterationOne(){

 ArrayList mark = new ArrayList();

 for(int i = 1; i< input.getWidth()-1;i++){

 for(int j = 1;j<input.getHeight()-1;j++){

 if(p[i][j].getMask()==1){

 Pixel[][] pixel = getNeighourMask(i,j);

 if(conditionA(pixel)&&conditionB(pixel)&&conditionC(pixel)&&conditionD(pixel)){

 mark.add(i);

 mark.add(j);

 }

 }

 }

 }

 return mark;

 }

 private ArrayList subIterationTwo(){

 ArrayList mark = new ArrayList();

 for(int i = 1; i< input.getWidth()-1;i++){

 for(int j = 1;j<input.getHeight()-1;j++){

 if(p[i][j].getMask()==1){

 Pixel[][] pixel = getNeighourMask(i,j);

 if(conditionA(pixel)&&conditionB(pixel)&&conditionCTwo(pixel)&&conditionDTwo(pixel)){

 mark.add(i);

 mark.add(j);

 }

 }

 }

 }

 return mark;

 }

 private void removeMark(ArrayList a){

 for(int i = 0;i<a.size();i = i+2){

 int x = Integer.parseInt(a.get(i).toString());

 int y = Integer.parseInt(a.get(i+1).toString());

 p[x][y].setMask(0);

 }

 }

 // thinning

 public BufferedImage getThinnedImage(){

 BufferedImage output = new BufferedImage(input.getWidth(),input.getHeight(),BufferedImage.TYPE\_BYTE\_GRAY);

 for(int m = 0;m<output.getWidth();m++){

 for(int n = 0;n<output.getHeight();n++){

 output.setRGB(m, n, 255\*256\*256+255\*256+255);

 }

 }

 boolean thin = true;

 while(thin==true){

 ArrayList array\_1 = subIterationOne();

 removeMark(array\_1);

 ArrayList array\_2 = subIterationTwo();

 removeMark(array\_2);

 if(array\_1.isEmpty()&&array\_2.isEmpty()){

 thin = false;

 }

 }

 for(int i = 1;i<output.getWidth()-1;i++){

 for(int j = 1;j<output.getHeight()-1;j++){

 if(p[i][j].getMask()==1){

 output.setRGB(i, j, 0\*256\*256+0\*256+0);

 }

 }

 }

 return output;

 }

 public Pixel[][] getThinningMatrix(){

 for(int i = 0; i<input.getWidth();i++){

 for(int j = 0; j<input.getHeight();j++){

 if(p[i][j].getMask()==1){

 p[i][j].setRGB(0);

 }else{

 p[i][j].setRGB(255);

 }

 }

 }

 return p;

 }

}

## 2.9 MinutiaeExtraction.java

package demo;

import java.awt.image.BufferedImage;

import java.util.ArrayList;

/\*\*This class is for extracting the minutiae

 \*/

public class MinutiaeExtraction {

 private BufferedImage input;

 private Pixel[][] p;

 public MinutiaeExtraction(BufferedImage bi,Pixel[][] pixel){

 this.input = bi;

 this.p = pixel;

 }

 private Pixel[] getNeighbourArray(int i, int j){

 Pixel[] array = new Pixel[8];

 array[0] = p[i-1][j-1];

 array[1] = p[i][j-1];

 array[2] = p[i+1][j-1];

 array[3] = p[i+1][j];

 array[4] = p[i+1][j+1];

 array[5] = p[i][j+1];

 array[6] = p[i-1][j+1];

 array[7] = p[i-1][j];

 return array;

 }

 public ArrayList getMinutiae(){

 ArrayList<Minutia> minutiae = new ArrayList();

 for(int i = 1; i<input.getWidth()-1;i++){

 for(int j = 1;j<input.getHeight()-1; j++){

 if(p[i][j].getMask()==1){

 Pixel[] neighbour = getNeighbourArray(i,j);

 int sum = 0;

 for(int index = 1; index<8;index++){

 sum = sum + Math.abs(neighbour[index].getMask()-neighbour[index-1].getMask());

 }

 sum = sum+Math.abs(neighbour[7].getMask()-neighbour[0].getMask());

 if(sum==2){

 Minutia m = new Minutia(i,j,1);

 m.setAngle(p[i][j].getAngle());

 minutiae.add(m);

 p[i][j].setType(1);

 }

 if(sum==6){

 Minutia m = new Minutia(i,j,2);

 m.setAngle(p[i][j].getAngle());

 minutiae.add(m);

 p[i][j].setType(2);

 }

 }

 }

 }

 return minutiae;

 }

}

## 2.10 RemoveFalseMinutiae.java

package demo;

import java.awt.image.BufferedImage;

import java.util.ArrayList;

/\*\*This class is for remove the false minutiae

 \*/

public class RemoveFalseMinutiae {

 private BufferedImage input;

 private ArrayList<Minutia> m;

 private Pixel[][] p;

 public RemoveFalseMinutiae(BufferedImage bi,ArrayList<Minutia> a, Pixel[][] pixel){

 this.input = bi;

 this.m = a;

 this.p = pixel;

 }

 private Pixel[][] getNeighbourMatrix(int i , int j){

 Pixel[][] n = new Pixel[25][25];

 int x = -12;

 for(int u = 0;u<25;u++){// matrxi x coordinate

 int y = -12;

 for(int v = 0; v<25;v++){//matrix y coordinate

 n[u][v] = p[i+x][j+y];

 y = y+1;

 }

 x = x + 1;

 }

 return n;

 }

 private Pixel[] getNeighbourArray(int i, int j){

 Pixel[] a = new Pixel[8];

 a[0]= p[i-1][j-1];

 a[1]= p[i][j-1];

 a[2]= p[i+1][j-1];

 a[3]= p[i+1][j];

 a[4]= p[i+1][j+1];

 a[5]= p[i][j+1];

 a[6]= p[i-1][j+1];

 a[7]= p[i-1][j];

 return a;

 }

 private void initMatrix(){

 for(int i = 0; i<input.getWidth();i++){

 for(int j = 0; j<input.getHeight();j++){

 p[i][j].setCheck(0);

 }

 }

 }

 // check spike, short ridge

 private boolean checkEnding(int x, int y){

 boolean b = true;

 Pixel current = p[x][y];

 int count = 0;// pixel count

 boolean find = false;

 while(find == false&&count<10){

 Pixel[] neighbour = getNeighbourArray(current.getX(),current.getY());//find neighbour

 current.setCheck(1);// mark current is checked

 if(findMinutiaeInNeighbour(neighbour)){

 find = true;

 b = false;

 }

 current = findNext(current, neighbour);

 count = count+1;

 }

 initMatrix();

 return b;

 }

 // check break ridge

 private boolean checkBreakEnding(int i , int j){

 boolean b = true;

 Pixel[][] pixel = getNeighbourMatrix(i,j);

 for(int u = 0; u<25; u++){

 for(int v = 0; v<25;v++){

 int index = findMinutia(pixel[u][v]);

 if(pixel[u][v].getType()==1&&m.get(index).getMark()){//without centre

 if(pixel[u][v].getY()!=j||pixel[u][v].getX()!=i){

 int y = pixel[u][v].getY()-j;

 int x = pixel[u][v].getX()-i;

 double angle1 = p[i][j].getAngle();

 double angle2 ;

 if(x==0){

 angle2 = Math.PI/2;

 }else{

 angle2 = Math.atan(y/x);

 }

 if(Math.abs(angle2-angle1)<Math.PI/12){

 m.get(index).setMark();

 b = false;

 }

 }

 }

 }

 }

 return b;

 }

 //remove bridge

 private boolean checkBridge(int i , int j){

 boolean b = true;

 p[i][j].setCheck(1);

 Pixel[]pixel = getNeighbourArray(i,j);

 ArrayList<Pixel> path = new ArrayList();

 for(int index = 1; index<pixel.length;index=index+2){

 if(pixel[index].getMask()==1){

 pixel[index].setCheck(1);

 path.add(pixel[index]) ;

 }

 }

 for(int index = 0; index<pixel.length;index=index+2){

 if(pixel[index].getMask()==1&&path.size()<3){

 pixel[index].setCheck(1);

 path.add(pixel[index]) ;

 }

 }

 boolean path1 = checkPath(path.get(0));

 boolean path2 = checkPath(path.get(1));

 boolean path3 = checkPath(path.get(2));

 if(path1==false||path2==false||path3==false){

 b = false;

 }

 initMatrix();

 return b;

 }

 // check bifurcation path

 private boolean checkPath(Pixel path){

 boolean b = true;

 Pixel current = path;

 int count = 0;// pixel count

 boolean find = false;

 while(find == false&&count<10){

 Pixel[] neighbour = getNeighbourArray(current.getX(),current.getY());//find neighbour

 current.setCheck(1);// mark current is checked

 if(findBifurcationInNeighbour(neighbour)){

 find = true;

 b = false;

 }

 current = findNext(current, neighbour);

 if(current==null){

 find = true;

 }

 count = count+1;

 }

 return b;

 }

 // find next pixel for checking

 private Pixel findNext(Pixel current, Pixel[] a){

 Pixel next = null;

 ArrayList<Pixel> p1 = new ArrayList();

 ArrayList<Pixel> p2 = new ArrayList();

 int i = 0;

 while(i<a.length){

 if(a[i].getMask()==1&&a[i].getCheck()==0){

 int x1 = a[i].getX();

 int x2 = current.getX();

 int y1 = a[i].getY();

 int y2 = current.getY();

 if(Math.abs((x1-x2)+(y1-y2))==1){

 p1.add(a[i]);

 }else{

 p2.add(a[i]);

 }

 }

 i = i +1;

 }

 if(p1.isEmpty()&&p2.isEmpty()){

 next = null;

 }else if(p1.isEmpty()){

 next = p2.get(0);

 }else {

 next = p1.get(0);

 }

 return next;

 }

 private boolean findMinutiaeInNeighbour(Pixel[] a){

 boolean b = false;

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

 if(a[i].getType()!=0&&a[i].getCheck()==0){

 int index = findMinutia(a[i]);

 m.get(index).setMark();

 b = true;// mark the checking ,mutiae is false;

 }

 }

 return b;

 }

 private boolean findBifurcationInNeighbour(Pixel[] a){

 boolean b = false;

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

 if(a[i].getType()==2&&a[i].getCheck()==0){

 int index = findMinutia(a[i]);

 m.get(index).setMark();

 b = true;// mark the checking ,mutiae is false;

 }

 }

 return b;

 }

 private boolean checkBound(int i, int j){

 boolean isBound = false;

 boolean up = false;

 boolean down = false;

 boolean left = false;

 boolean right = false;

 //check up

 boolean findUp = false;

 int k = 1;

 while(findUp==false&&k<19){

 if(j-k<=1){

 up=true;

 findUp = true;

 }

 k = k+1;

 }

 if(j>=input.getHeight()-10){

 down=true;

 }

 //check left

 boolean findLeft = false;

 int l = 1;

 while(findLeft==false&&l<19){

 if(i-l<=4){

 left=true;

 findLeft = true;

 }

 if(p[i-l][j].getMask()==1){

 findLeft = true;

 }

 l = l+1;

 if(l==19){

 left=true;

 }

 }

 //check right

 boolean findRight = false;

 int r = 1;

 while(findRight==false&&r<19){

 if(i+r>=input.getWidth()-6){

 right=true;

 findRight = true;

 }

 if(p[i+r][j].getMask()==1){

 findRight = true;

 }

 r = r+1;

 if(r==19){

 right=true;

 }

 }

 if(i<=input.getWidth()/2-50){

 if(left){

 isBound=true;

 }

 }else if(i>input.getWidth()/2+50){

 if(right){

 isBound=true;

 }

 }

 if(up||down){

 isBound = true;

 }

 return isBound;

 }

 private int findMinutia(Pixel pixel){

 int index = 0;

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

 int x = m.get(i).getX();

 int y = m.get(i).getY();

 if(pixel.getX() == x && pixel.getY() == y){

 index = i;

 }

 }

 return index;

 }

 public ArrayList<Minutia> getRemovalMinutiae(){

 ArrayList<Minutia> new\_minutiae = new ArrayList();

 //remove bound

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

 if(m.get(i).getType()!=0&&m.get(i).getMark()){

 int x = m.get(i).getX();

 int y = m.get(i).getY();

 if(checkBound(x,y)){

 m.get(i).setMark();

 }

 }

 }

 //remove spike, dot

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

 if(m.get(i).getType()==1&&m.get(i).getMark()){

 int x = m.get(i).getX();

 int y = m.get(i).getY();

 if(checkEnding(x,y)==false){

 m.get(i).setMark();

 }

 }

 }

 //remove break

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

 if(m.get(i).getType()==1&&m.get(i).getMark()){

 int x = m.get(i).getX();

 int y = m.get(i).getY();

 if(x>13&&y>13&&x<246&&y<286){

 boolean b = checkBreakEnding(x,y);

 if(b==false){

 m.get(i).setMark();

 }

 }

 }

 }

 //remove bridge

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

 if(m.get(i).getType()==2&&m.get(i).getMark()){

 int x = m.get(i).getX();

 int y = m.get(i).getY();

 if(x>11&&y>11&&x<248&&y<288){

 boolean b = checkBridge(x,y);

 if(b==false){

 m.get(i).setMark();

 }

 }

 }

 }

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

 if(m.get(i).getMark()){

 new\_minutiae.add(m.get(i));

 }

 }

 return new\_minutiae;

 }

}

## 2.11 PointDirection.java

package demo;

import java.awt.image.BufferedImage;

/\*\*This class compute the point orientation for core detection

 \*/

public class PointDirection {

 private BufferedImage input;

 public Point[][] p;

 public PointDirection(BufferedImage bi){

 this.input = bi;

 init();

 getDirection();

 getBlockAngle();

 SmoothDirection();

 }

 private void init(){

 p = new Point[input.getWidth()][input.getHeight()];

 for(int i = 0;i<input.getWidth();i++){

 for(int j = 0;j<input.getHeight();j++){

 int rgb = (input.getRGB(i, j)&0x000000ff);

 Point po = new Point(i,j,rgb);

 p[i][j]= po;

 }

 }

 }

 private Point[][] getMactrix(int i, int j){

 Point[][] m = new Point[9][9];

 int w = -4;

 for(int x = 0; x<9;x++){

 int h = -4;

 for(int y = 0; y<9;y++){

 m[x][y] = p[i+w][j+h];

 h = h +1;

 }

 w = w+1;

 }

 return m;

 }

 private double[] getGray(Point[][] m){

 double[] g = new double[8];

 g[0] = (m[0][4].getRGB()+m[2][4].getRGB()+m[6][4].getRGB()+m[8][4].getRGB())/4;

 g[1] = (m[0][6].getRGB()+m[2][5].getRGB()+m[6][3].getRGB()+m[8][2].getRGB())/4;

 g[2] = (m[0][8].getRGB()+m[2][6].getRGB()+m[6][2].getRGB()+m[8][0].getRGB())/4;

 g[3] = (m[2][8].getRGB()+m[3][6].getRGB()+m[5][2].getRGB()+m[6][0].getRGB())/4;

 g[4] = (m[4][0].getRGB()+m[4][2].getRGB()+m[4][6].getRGB()+m[4][8].getRGB())/4;

 g[5] = (m[2][0].getRGB()+m[2][3].getRGB()+m[5][6].getRGB()+m[6][8].getRGB())/4;

 g[6] = (m[0][0].getRGB()+m[2][2].getRGB()+m[6][6].getRGB()+m[8][8].getRGB())/4;

 g[7] = (m[0][2].getRGB()+m[2][3].getRGB()+m[6][5].getRGB()+m[8][6].getRGB())/4;

 return g;

 }

 private double[] getCha(double[] i){

 double[] g = new double[4];

 g[0] = Math.abs(i[0]-i[4]);

 g[1] = Math.abs(i[1]-i[5]);

 g[2] = Math.abs(i[2]-i[6]);

 g[3] = Math.abs(i[3]-i[7]);

 return g;

 }

 private int[] findMax(double[] g){

 int[] d = new int[2];

 double temp = 0;

 int index = 0;

 for(int i = 0; i<4;i++){

 if(temp<g[i]){

 temp = g[i];

 index = i;

 }

 }

 d[0] = index;

 d[1] = index+4;

 return d;

 }

 private void getDirection(){

 for(int i = 4;i<input.getWidth()-5;i++){

 for(int j = 4; j<input.getHeight()-5;j++){

 Point[][] n = getMactrix(i,j);

 double[] average = getGray(n);

 double[] abs = getCha(average);

 int[] max = findMax(abs);

 int rgb = p[i][j].getRGB();

 int rgb\_1 = (int) average[max[0]];

 int rgb\_2 = (int) average[max[1]];

 if(Math.abs(rgb-rgb\_1)<Math.abs(rgb-rgb\_2)){

 p[i][j].setD(max[0]);

 }else{

 p[i][j].setD(max[1]);

 }

 }

 }

 }

 private Point[][] getSmoothMatrix(Point[][] clone,int i, int j){

 Point[][] ma = new Point[13][13];

 int m = -6;

 for(int x = 0; x<13;x++){

 int n = -6;

 for(int y = 0; y<13;y++){

 ma[x][y] = clone[i+m][j+n];

 n = n+1;

 }

 m = m+1;

 }

 return ma;

 }

 private int getMaxIndex(Point[][] po){

 int maxIndex = 0;

 int[] x = new int[8];

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

 x[i] = 0;

 }

 for(int i = 0; i<13;i++){

 for(int j = 0; j<13;j++){

 int d = po[i][j].getDirection();

 x[d] = x[d]+1;

 }

 }

 int temp = 0;

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

 if(temp<x[i]){

 temp = x[i];

 maxIndex = i;

 }

 }

 return maxIndex;

 }

 private void SmoothDirection(){

 Point[][] clone = p;

 for(int i = 10; i<input.getWidth()-10;i++){

 for(int j = 10;j<input.getHeight()-10;j++){

 Point[][] m = getSmoothMatrix(clone,i,j);

 int d = getMaxIndex(m);

 p[i][j].setD(d);

 }

 }

 }

 private void getBlockAngle(){

 Point[][] clone = p;

 for(int i = 10; i+13<input.getWidth()-10;i=i+13){

 for(int j = 10;j+13<input.getHeight()-10;j=j+13){

 Point[][] m = getBlockMatrix(clone,i,j);

 int d = getMaxIndex(m);

 setAngle(i,j,d);

 }

 }

 }

 private void setAngle(int x, int y,int d){

 for(int i = -6;i<7;i++){

 for(int j = -6;j<7;j++){

 p[x+i][y+j].setD(d);

 }

 }

 }

 private Point[][] getBlockMatrix(Point[][] clone,int i, int j){

 Point[][] ma = new Point[13][13];

 int m = -6;

 for(int x = 0; x<13;x++){

 int n = -6;

 for(int y = 0; y<13;y++){

 ma[x][y] = clone[i+m][j+n];

 n = n+1;

 }

 m = m+1;

 }

 return ma;

 }

}

## 2.12 FindCore.java

package demo;

import java.awt.image.BufferedImage;

import java.util.ArrayList;

/\*\*This class is Core point detection algorithm

 \*/

public class FindCore {

 private BufferedImage input;

 private Point[][] p = null;

 public FindCore(BufferedImage bi, Point[][] p){

 this.input = bi;

 this.p = p;

 }

 private Point[] getFourArray(int i, int j){

 Point[] m = new Point[4];

 m[0] = p[i][j];

 m[1] = p[i+1][j];

 m[2] = p[i+1][j-1];

 m[3] = p[i][j-1];

 return m;

 }

 private int getPoinCareOne(Point[] poi){

 int pc = 0;

 for(int m = 1;m<4;m++){

 int k = poi[m].getDirection()-poi[m-1].getDirection();

 if(Math.abs(k)<4){

 ;

 }else if(k<=-4){

 k = k+8;

 }else{

 k = k-8;

 }

 pc = pc+k;

 }

 int last = poi[0].getDirection()-poi[3].getDirection();

 if(Math.abs(last)<4){

 ;

 }else if(last<=-4){

 last = last+8;

 }else{

 last = last-8;

 }

 pc = pc+last;

 return pc;

 }

 public Point getCore(){

 Point po = p[1][1];

 ArrayList<Point> array = new ArrayList();

 for(int i = 8; i<input.getWidth()-8;i++){

 for(int j = 8; j<input.getHeight()-8;j++){

 Point[] a = getFourArray(i,j);

 int poincare = getPoinCareOne(a);

 Point[][] m\_1 = getCircle(i,j);

 Point[] m\_2 = get24Point(m\_1);

 int poincare\_2 = getPoinCareTWO(m\_2);

 if(poincare==8&&poincare\_2>=8){

 if(i<220&&j<230){

 array.add(p[i][j]);

 }

 }

 }

 }

 if(array.size()==2){

 double angle = array.get(0).getDirection()\*Math.PI/8;

 double angle\_2 = array.get(1).getDirection()\*Math.PI/8;

 double x\_1 = array.get(0).y\*Math.cos(angle)-array.get(0).x\*Math.sin(angle);

 double x\_2 = array.get(1).y\*Math.cos(angle\_2)-array.get(1).x\*Math.sin(angle\_2);

 if(x\_1<x\_2){

 po = array.get(1);

 }else{

 po = array.get(0);

 }

 }else if(array.size()==1){

 po = array.get(0);

 }

 return po;

 }

 private Point[][] getCircle(int i, int j){

 Point[][] m = new Point[9][9];

 int w = -4;

 for(int x = 0; x<9;x++){

 int h = -4;

 for(int y = 0; y<9;y++){

 m[x][y] = p[i+w][j+h];

 h = h +1;

 }

 w = w+1;

 }

 return m;

 }

 private Point[] get24Point(Point[][] m){

 Point[] array = new Point[24];

 array[0] = m[0][4];

 array[1] = m[0][5];

 array[2] = m[0][6];

 array[3] = m[1][7];

 array[4] = m[2][8];

 array[5] = m[3][8];

 array[6] = m[4][8];

 array[7] = m[5][8];

 array[8] = m[6][8];

 array[9] = m[7][7];

 array[10] = m[8][6];

 array[11] = m[8][5];

 array[12] = m[8][4];

 array[13] = m[8][3];

 array[14] = m[8][2];

 array[15] = m[7][1];

 array[16] = m[0][6];

 array[17] = m[0][5];

 array[18] = m[0][4];

 array[19] = m[0][3];

 array[20] = m[0][2];

 array[21] = m[1][1];

 array[22] = m[0][2];

 array[23] = m[0][3];

 return array;

 }

 private int getPoinCareTWO(Point[] poi){

 int pc = 0;

 for(int m = 1;m<24;m++){

 int k = poi[m].getDirection()-poi[m-1].getDirection();

 if(Math.abs(k)<4){

 ;

 }else if(k<=-4){

 k = k+8;

 }else{

 k = k-8;

 }

 pc = pc+k;

 }

 int last = poi[0].getDirection()-poi[23].getDirection();

 if(Math.abs(last)<4){

 ;

 }else if(last<=-4){

 last = last+8;

 }else{

 last = last-8;

 }

 pc = pc+last;

 return pc;

 }

}

## 2.13 GenerateFeatrue.java

package demo;

import java.util.ArrayList;

/\*\*This class is generate the featrue vector for matching

 \*/

public class GenerateFeatrue {

 private Point core;

 private ArrayList<Minutia> array;

 public GenerateFeatrue(Point c, ArrayList<Minutia> a){

 this.core = c;

 this.array = a;

 }

 private int getDirection(Minutia min){// distance

 int d = 0;

 int x = (int) Math.pow((min.getX()-core.getX()), 2);

 int y = (int) Math.pow((min.getY()-core.getY()), 2);

 d = (int) Math.sqrt(x+y);

 return d;

 }

 private double getAngle(Minutia min){// Pola angle

 double a = 0;

 int x = min.getX()-core.getX();

 int y = min.getY()-core.getY();

 double theta = 0;

 if(x==0){

 theta = Math.PI/2;

 }else{

 theta = Math.atan(y/x);

 }

 a = theta - core.getDirection()\*Math.PI/8;

 return a;

 }

 private double getDifferentAngle(Minutia min){// different angle between core and minutia, it is a rotated angle

 double a = 0;

 a = min.getAngle() - core.getDirection()\*Math.PI/8;

 return a;

 }

 public ArrayList<Featrue> getMathFeatrueList(){

 ArrayList<Featrue> f = new ArrayList();

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

 Minutia m = array.get(i);

 int di = getDirection(m);

 double an = getAngle(m);

 int type = m.getType();

 double diff = getDifferentAngle(m);

 Featrue ft = new Featrue(m,di,an,type,diff);

 f.add(ft);

 }

 return f;

 }

}

## 2.14 Match.java

package demo;

import java.util.ArrayList;

/\*\*This class if match two image's featrue vector

 \*/

public class Match {

 private ArrayList<Featrue> input;//input image featrue list

 private ArrayList<Featrue> temp;//template image featrue list

 private Point core\_i, core\_t;

 private int pair = 0;

 private ArrayList<Featrue> m;

 public Match(ArrayList<Featrue> i, ArrayList<Featrue> t,Point c\_1, Point c\_2){

 this.input = i;

 this.temp = t;

 this.core\_i = c\_1;

 this.core\_t = c\_2;

 }

 public boolean isMatch(){

 ArrayList<Featrue> af = new ArrayList();

 boolean b = false;

 int count = 0;

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

 ArrayList<Featrue> tf = new ArrayList();

 boolean isMatch = false;

 int n = 0;

 Featrue input\_featrue = input.get(i);

 while(isMatch == false&&n<temp.size()){

 Featrue temp\_featrue = temp.get(n);

 if(temp\_featrue.getMatch()==false){

 if(input\_featrue.getType()==temp\_featrue.getType()){// if is the same type

 int input\_d = input\_featrue.getDirection();

 int temp\_d = temp\_featrue.getDirection();

 if(input\_d<=20){// the distance is less than 20 pixel very close to core

 if(Math.abs(input\_d-temp\_d)<=8){// if the direaction is the less than 8 pixel

 double input\_a = input\_featrue.getAngle();

 double temp\_a = temp\_featrue.getAngle();

 if(Math.abs(input\_a-temp\_a)<(Math.PI/4)){//if the poistion is less than PI/4;

 double diff\_input = input\_featrue.getDifferentAngle();

 double diff\_temp = temp\_featrue.getDifferentAngle();

 if(Math.abs(diff\_input-diff\_temp)<=(Math.PI/4)){//if the different angle is less than PI/4

 tf.add(temp\_featrue);

 if(tf.size()>1){// if these have more than 1 features , check the more similar one

 temp\_featrue = getMinDistanceFeatrue(tf, input\_featrue);

 int index = findFeatrueInTemplate(temp\_featrue);

 count = count+1;

 temp.get(index).setMatch(true);

 af.add(input\_featrue);

 af.add(temp\_featrue);

 isMatch = true;

 }else if(tf.size()==1){

 count = count+1;

 temp\_featrue = tf.get(0);

 int index = findFeatrueInTemplate(temp\_featrue);

 temp.get(index).setMatch(true);

 af.add(input\_featrue);

 af.add(temp\_featrue);

 isMatch = true;

 }

 }

 }

 }

 }else if(input\_d>20&&input\_d<=60&&checkPosition(input\_featrue,temp\_featrue)){//the distance is less than 60

 if(Math.abs(input\_d-temp\_d)<=10){// if the direaction is the less than 10 pixel

 double input\_a = input\_featrue.getAngle();

 double temp\_a = temp\_featrue.getAngle();

 if(Math.abs(input\_a-temp\_a)<(Math.PI/8)){//if the poistion is less than PI/8;

 double diff\_input = input\_featrue.getDifferentAngle();

 double diff\_temp = temp\_featrue.getDifferentAngle();

 if(Math.abs(diff\_input-diff\_temp)<=(Math.PI/10)){//if the different angle is less than PI/10

 tf.add(temp\_featrue);

 if(tf.size()>1){// if these have more than 1 features , check the more similar one

 temp\_featrue = getMinDistanceFeatrue(tf, input\_featrue);

 int index = findFeatrueInTemplate(temp\_featrue);

 count = count+1;

 temp.get(index).setMatch(true);

 af.add(input\_featrue);

 af.add(temp\_featrue);

 isMatch = true;

 }else if(tf.size()==1){

 count = count+1;

 temp\_featrue = tf.get(0);

 int index = findFeatrueInTemplate(temp\_featrue);

 temp.get(index).setMatch(true);

 af.add(input\_featrue);

 af.add(temp\_featrue);

 isMatch = true;

 }

 }

 }

 }

 }else if(input\_d>60&&input\_d<=120&&checkPosition(input\_featrue,temp\_featrue)){//the distance is less than 120

 if(Math.abs(input\_d-temp\_d)<=12){// if the direaction is the less than 12 pixel

 double input\_a = input\_featrue.getAngle();

 double temp\_a = temp\_featrue.getAngle();

 if(Math.abs(input\_a-temp\_a)<(Math.PI/8)){//if the poistion is less than PI/8;

 double diff\_input = input\_featrue.getDifferentAngle();

 double diff\_temp = temp\_featrue.getDifferentAngle();

 if(Math.abs(diff\_input-diff\_temp)<=(Math.PI/14)){//if the different angle is less than PI/14

 tf.add(temp\_featrue);

 if(tf.size()>1){// if these have more than 1 features , check the more similar one

 temp\_featrue = getMinDistanceFeatrue(tf, input\_featrue);

 int index = findFeatrueInTemplate(temp\_featrue);

 count = count+1;

 temp.get(index).setMatch(true);

 af.add(input\_featrue);

 af.add(temp\_featrue);

 isMatch = true;

 }else if(tf.size()==1){

 count = count +1;

 temp\_featrue = tf.get(0);

 int index = findFeatrueInTemplate(temp\_featrue);

 temp.get(index).setMatch(true);

 af.add(input\_featrue);

 af.add(temp\_featrue);

 isMatch = true;

 }

 }

 }

 }

 }else if(input\_d>120&&checkPosition(input\_featrue,temp\_featrue)){//the distahce is greater than 120

 if(Math.abs(input\_d-temp\_d)<=12){// if the direaction is the less than 12 pixel

 double input\_a = input\_featrue.getAngle();

 double temp\_a = temp\_featrue.getAngle();

 if(Math.abs(input\_a-temp\_a)<(Math.PI/18)){//if the poistion is less than PI/18;

 double diff\_input = input\_featrue.getDifferentAngle();

 double diff\_temp = temp\_featrue.getDifferentAngle();

 if(Math.abs(diff\_input-diff\_temp)<=(Math.PI/14)){//if the different angle is less than PI/14

 tf.add(temp\_featrue);

 if(tf.size()>1){// if these have more than 1 features , check the more similar one

 temp\_featrue = getMinDistanceFeatrue(tf, input\_featrue);

 int index = findFeatrueInTemplate(temp\_featrue);

 count = count+1;

 temp.get(index).setMatch(true);

 af.add(input\_featrue);

 af.add(temp\_featrue);

 isMatch = true;

 }else if(tf.size()==1){

 count = count+1;

 temp\_featrue = tf.get(0);

 int index = findFeatrueInTemplate(temp\_featrue);

 temp.get(index).setMatch(true);

 af.add(input\_featrue);

 af.add(temp\_featrue);

 isMatch = true;

 }

 }

 }

 }

 }

 }

 }

 n = n+1;

 }

 }

 m = af;

 pair = count;

 if(count >=12){

 b = true;

 }

 return b;

 }

 private Featrue getMinDistanceFeatrue(ArrayList<Featrue> tf, Featrue i){

 Featrue f = null;

 double t = 1000;

 int index = 0;

 for(int j = 0; j<tf.size();j++){

 double angle = Math.abs(tf.get(j).getAngle()-i.getAngle());

 if(angle<t){

 t = angle;

 index = j;

 }

 }

 f = tf.get(index);

 return f;

 }

 private boolean checkPosition(Featrue i, Featrue t){

 boolean b = false;

 int input\_x = i.getMinutia().getX();

 int input\_y = i.getMinutia().getY();

 input\_x = input\_x-core\_i.getX();

 input\_y = input\_y-core\_i.getY();

 int rotate\_x = (int) (input\_x \* Math.cos(core\_i.getDirection() \* Math.PI / 8) + input\_y \* Math.sin(core\_i.getDirection() \* Math.PI / 8));

 int rotate\_y = (int) (input\_y \* Math.cos(core\_i.getDirection() \* Math.PI / 8) - input\_x \* Math.sin(core\_i.getDirection() \* Math.PI / 8));

 input\_x = rotate\_x + core\_i.getX();

 input\_y = rotate\_y + core\_i.getY();

 int temp\_x = t.getMinutia().getX();

 int temp\_y = t.getMinutia().getY();

 temp\_x = temp\_x - core\_t.getX();

 temp\_y = temp\_y - core\_t.getY();

 int rotate\_tx = (int) (temp\_x \* Math.cos(core\_t.getDirection() \* Math.PI / 8) + temp\_y \* Math.sin(core\_t.getDirection() \* Math.PI / 8));

 int rotate\_ty = (int) (temp\_y \* Math.cos(core\_t.getDirection() \* Math.PI / 8) - temp\_x \* Math.sin(core\_t.getDirection() \* Math.PI / 8));

 temp\_x = rotate\_tx + core\_t.getX();

 temp\_y = rotate\_ty + core\_t.getY();

 if(input\_x<=core\_i.getX()&&temp\_x<=core\_t.getX()){//in left of the axies

 if(input\_y<=core\_i.getY()&&temp\_y<=core\_t.getY()){

 b = true;

 }else if(input\_y>=core\_i.getY()&&temp\_y>=core\_i.getY()){

 b = true;

 }

 }else if(input\_x>=core\_i.getX()&&temp\_x>=core\_t.getX()){// in right of the axies

 if(input\_y<=core\_i.getY()&&temp\_y<=core\_t.getY()){

 b = true;

 }else if(input\_y>=core\_i.getY()&&temp\_y>=core\_i.getY()){

 b = true;

 }

 }

 return b;

 }

 private int findFeatrueInTemplate(Featrue f){

 int index = 0;

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

 int temp\_x = temp.get(i).getMinutia().getX();

 int temp\_y = temp.get(i).getMinutia().getY();

 int x = f.getMinutia().getX();

 int y = f.getMinutia().getY();

 if(temp\_x == x&&temp\_y == y){

 index = i;

 }

 }

 return index;

 }

 public int getPair(){

 return pair;

 }

 public ArrayList<Featrue> getSameArray(){

 return m;

 }

}

# Database access part

## 3.1 DAO.java

package demo;

import java.sql.\*;

import java.util.Properties;

/\*\*This is database access object class, it provides a standard to access database

 \*/

public class DAO {

 static {

 try {

 Class.forName("org.apache.derby.jdbc.EmbeddedDriver");

 } catch (ClassNotFoundException e) {

 e.printStackTrace();

 }

 }

 private DAO() {

 }

 public static Connection getConnection() {

 Connection myCon = null;

 Properties props = new Properties();

 props.put("user", "fingerprintRec"); props.put("password", "fingerprint123");

 try {

 myCon = DriverManager.getConnection("jdbc:derby:FingerprintsDB;create=false", props);

 myCon.setAutoCommit(false);

 } catch (SQLException e) {

 e.printStackTrace();

 }

 return myCon;

 }

 public static Statement getStatement(Connection conn) {

 Statement myStatement = null;

 try {

 myStatement = conn.createStatement();

 } catch (SQLException e) {

 e.printStackTrace();

 }

 return myStatement;

 }

 public static int executeUpdate(Connection conn, String sql) {

 int a = 0;

 try {

 a = conn.createStatement().executeUpdate(sql);

 } catch (SQLException e) {

 e.printStackTrace();

 }

 return a;

 }

 public static boolean execute(Connection conn, String sql) {

 boolean flag = false;

 try {

 flag = conn.createStatement().execute(sql);

 } catch (SQLException e) {

 e.printStackTrace();

 }

 return flag;

 }

 public static ResultSet executeQuery(Statement stmt, String sql) {

 ResultSet rs = null;

 try {

 rs = stmt.executeQuery(sql);

 } catch (SQLException e) {

 e.printStackTrace();

 }

 return rs;

 }

 public static ResultSet executeQuery(Connection conn, String sql) {

 ResultSet rs = null;

 try {

 rs = conn.createStatement().executeQuery(sql);

 } catch (SQLException e) {

 e.printStackTrace();

 }

 return rs;

 }

 public static PreparedStatement getPstmt(Connection conn, String sql) {

 PreparedStatement pstmt = null;

 try {

 pstmt = conn.prepareStatement(sql);

 } catch (SQLException e) {

 e.printStackTrace();

 }

 return pstmt;

 }

 public static void closeRS(ResultSet rs) {

 try {

 if(rs != null) {

 rs.close();

 rs = null;

 }

 } catch (SQLException e) {

 e.printStackTrace();

 }

 }

 public static void closePstmt(PreparedStatement pstmt) {

 try {

 if(pstmt != null) {

 pstmt.close();

 pstmt = null;

 }

 } catch (SQLException e) {

 e.printStackTrace();

 }

 }

 public static void closeStmt(Statement stmt) {

 try {

 if(stmt != null) {

 stmt.close();

 stmt = null;

 }

 } catch (SQLException e) {

 e.printStackTrace();

 }

 }

 public static void closeConn(Connection conn) {

 try {

 if(conn != null) {

 conn.commit();

 conn.close();

 conn = null;

 }

 } catch (SQLException e) {

 e.printStackTrace();

 }

 }

}

## 3.2 FeatrueDAO.java

package demo;

import java.sql.Connection;

import java.sql.PreparedStatement;

import java.sql.ResultSet;

import java.sql.SQLException;

import java.sql.Statement;

import java.util.ArrayList;

import javax.swing.JOptionPane;

/\*\*This class is provides the database access function for enroll the featrue point

 \* in database

 \*/

public class FeatrueDAO {

 public static void enrollFeatrue(Featrue f, int id,String image\_id){

 Connection conn=null;

 String sql = "INSERT INTO fingerprintMinutiae VALUES (?,?,?,?,?,?,?,?,?)";

 PreparedStatement pstmt = null;

 Minutia m = f.getMinutia();

 try{

 conn = DAO.getConnection();

 pstmt = DAO.getPstmt(conn, sql);

 pstmt.setInt(1, id);

 pstmt.setString(2, image\_id);

 pstmt.setInt(3, m.getX());

 pstmt.setInt(4, m.getY());

 pstmt.setDouble(5, m.getAngle());

 pstmt.setInt(6, f.getType());

 pstmt.setInt(7, f.getDirection());

 pstmt.setDouble(8, f.getAngle());

 pstmt.setDouble(9, f.getDifferentAngle());

 pstmt.executeUpdate();

 }catch (SQLException exp) {

 JOptionPane.showMessageDialog(null, "Error");

 } finally {

 DAO.closePstmt(pstmt);

 DAO.closeConn(conn);

 }

 }

 public static ArrayList<Featrue> getFeatrue(int type, int d, double angle, double diff\_angle){

 ArrayList<Featrue> array = new ArrayList();

 Connection conn=null;

 ResultSet rs=null;

 int d\_1 = 0;

 int d\_2 = 0;

 double angle\_1 =0;

 double angle\_2 = 0;

 double diff\_1 = 0;

 double diff\_2 = 0;

 if(d<=20){

 d\_1 = d+8;

 d\_2 = d-8;

 angle\_1 = angle + Math.PI/4;

 angle\_2 = angle - Math.PI/4;

 diff\_1 = diff\_angle + Math.PI/4;

 diff\_2 = diff\_angle - Math.PI/4;

 }else if(d>20&&d<=60){

 d\_1 = d+10;

 d\_2 = d-10;

 angle\_1 = angle + Math.PI/8;

 angle\_2 = angle - Math.PI/8;

 diff\_1 = diff\_angle + Math.PI/10;

 diff\_2 = diff\_angle - Math.PI/10;

 }else if(d>60&&d<=120){

 d\_1 = d+12;

 d\_2 = d-12;

 angle\_1 = angle + Math.PI/8;

 angle\_2 = angle - Math.PI/8;

 diff\_1 = diff\_angle + Math.PI/14;

 diff\_2 = diff\_angle - Math.PI/14;

 }else if(d>120){

 d\_1 = d+12;

 d\_2 = d-12;

 angle\_1 = angle + Math.PI/18;

 angle\_2 = angle - Math.PI/18;

 diff\_1 = diff\_angle + Math.PI/14;

 diff\_2 = diff\_angle - Math.PI/14;

 }

 String sql = "SELECT \* FROM fingerprintMinutiae WHERE TYPE="+type+" AND DISTANCE <="+d\_1+"AND DISTANCE>="+d\_2+"AND LOCAL\_ANGLE <="+angle\_1+"AND LOCAL\_ANGLE>="+angle\_2+" AND DIFF\_ANGLE<="+diff\_1+"AND DIFF\_ANGLE >="+diff\_2+"";

 try{

 conn=DAO.getConnection();

 rs=conn.createStatement().executeQuery(sql);

 while(rs.next()){

 Minutia myMinutia = new Minutia(rs.getInt("COOR\_X"),rs.getInt("COOR\_Y"),rs.getInt("TYPE"));

 Featrue myFeature = new Featrue(myMinutia, rs.getInt("DISTANCE"),rs.getDouble("LOCAL\_ANGLE"),rs.getInt("TYPE"),rs.getDouble("DIFF\_ANGLE"));

 myFeature.setImage(rs.getString("FINGERPRINT\_ID"));

 myFeature.setID(rs.getInt("ID"));

 array.add(myFeature);

 }

 }catch(SQLException exp){

 JOptionPane.showMessageDialog(null, "Match Error");

 exp.printStackTrace();;

 }finally{

 DAO.closeRS(rs);

 DAO.closeConn(conn);

 }

 return array;

}

 public static void deleteFeatrue(String iamge\_id){

 Connection conn = null;

 Statement myStatement=null;

 String sql = "DELETE FROM fingerprintMinutiae WHERE FINGERPRINT\_ID='"+iamge\_id+"'";

 try{

 conn = DAO.getConnection();

 myStatement=DAO.getStatement(conn);

 myStatement.executeUpdate(sql);

 } catch (SQLException exp) {

 JOptionPane.showMessageDialog(null, "Deletion Error!");

 } finally {

 DAO.closeStmt(myStatement);

 DAO.closeConn(conn);

 }

 }

 public static int getMaxId(){

 int max = 0;

 ArrayList array = new ArrayList();

 Connection conn=null;

 ResultSet rs=null;

 String sql = "SELECT \* FROM fingerprintMinutiae ORDER BY ID DESC ";

 try{

 conn=DAO.getConnection();

 rs=conn.createStatement().executeQuery(sql);

 while(rs.next()){

 array.add(rs.getInt(1));

 }

 if(array.isEmpty()){

 max = 0;

 }else{

 max = Integer.parseInt(array.get(0).toString());

 }

 }catch(SQLException exp){

 JOptionPane.showMessageDialog(null, "Error in find the max id");

 exp.printStackTrace();

 }finally{

 DAO.closeRS(rs);

 DAO.closeConn(conn);

 }

 return max;

 }

 public static int getFeatrueNO(String image\_id){

 ArrayList<Featrue> array = new ArrayList();

 Connection conn=null;

 ResultSet rs=null;

 String sql = "SELECT \* FROM fingerprintMinutiae WHERE FINGERPRINT\_ID='"+image\_id+"' ";

 try{

 conn=DAO.getConnection();

 rs=conn.createStatement().executeQuery(sql);

 while(rs.next()){

 Minutia myMinutia = new Minutia(rs.getInt("COOR\_X"),rs.getInt("COOR\_Y"),rs.getInt("TYPE"));

 Featrue myFeature = new Featrue(myMinutia, rs.getInt("DISTANCE"),rs.getDouble("LOCAL\_ANGLE"),rs.getInt("TYPE"),rs.getDouble("DIFF\_ANGLE"));

 myFeature.setImage(rs.getString("FINGERPRINT\_ID"));

 myFeature.setID(rs.getInt("ID"));

 array.add(myFeature);

 }

 }catch(SQLException exp){

 JOptionPane.showMessageDialog(null, "Match Error");

 exp.printStackTrace();;

 }finally{

 DAO.closeRS(rs);

 DAO.closeConn(conn);

 }

 return array.size();

 }

}

## 3.3 FingerprintImageDAO.java

package demo;

import java.sql.\*;

import javax.swing.JOptionPane;

/\*\*This class provides datavase access function for enroll the image information

 \* into database

 \*/

public class FingerprintImageDAO {

 public static FingerprintImage getImage(String image\_id){

 FingerprintImage image = null;

 Connection conn=null;

 ResultSet rs = null;

 String sql = "SELECT \* FROM fingerprintImage WHERE FINGERPRINT\_ID='"+image\_id+"'";

 try{

 conn=DAO.getConnection();

 rs=conn.createStatement().executeQuery(sql);

 while(rs.next()){

 image = new FingerprintImage(rs.getString("FINGERPRINT\_ID"),rs.getString("IMAGE\_DIRECTORY"),rs.getString("DESCRIPTION"),rs.getInt("CORE\_X"),rs.getInt("CORE\_Y"),rs.getDouble("CORE\_ANGLE"));

 }

 }catch(SQLException exp){

 exp.printStackTrace();

 }finally{

 DAO.closeRS(rs);

 DAO.closeConn(conn);

 }

 return image;

 }

 public static void saveImage(FingerprintImage image){

 Connection conn=null;

 String sql = "INSERT INTO fingerprintImage VALUES (?,?,?,?,?,?)";

 PreparedStatement pstmt = null;

 try {

 conn = DAO.getConnection();

 pstmt = DAO.getPstmt(conn, sql);

 pstmt.setString(1, image.getID());

 pstmt.setString(2, image.getDirectory());

 pstmt.setString(3, image.getDescription());

 pstmt.setInt(4, image.getCoreX());

 pstmt.setInt(5, image.getCoreY());

 pstmt.setDouble(6, image.getCoreAngle());

 pstmt.executeUpdate();

 } catch (SQLException exp) {

 exp.printStackTrace();

 } finally {

 DAO.closePstmt(pstmt);

 DAO.closeConn(conn);

 }

 }

 public static void deleteImage(String image\_id){

 Connection conn = null;

 Statement myStatement=null;

 String sql = "DELETE FROM fingerprintImage WHERE FINGERPRINT\_ID='"+image\_id+"' ";

 try{

 conn = DAO.getConnection();

 myStatement=DAO.getStatement(conn);

 myStatement.executeUpdate(sql);

 } catch (SQLException exp) {

 JOptionPane.showMessageDialog(null, "Delete Error!");

 exp.printStackTrace();

 } finally {

 DAO.closeStmt(myStatement);

 DAO.closeConn(conn);

 JOptionPane.showMessageDialog(null, "Delete completed");

 }

 }

 public static void updataImage(String id,String des){

 Connection conn = null;

 String sql = "UPDATE fingerprintImage SET DESCRIPTION=? WHERE FINGERPRINT\_ID='" + id + "'";

 PreparedStatement pstmt = null;

 try {

 conn = DAO.getConnection();

 pstmt = DAO.getPstmt(conn, sql);

 pstmt.setString(1, des);

 pstmt.executeUpdate();

 } catch (SQLException exp) {

 JOptionPane.showMessageDialog(null, "Update error");

 } finally {

 DAO.closePstmt(pstmt);

 DAO.closeConn(conn);

 JOptionPane.showMessageDialog(null, "Update completed");

 }

 }

}

## 3.4 MatchDB.java

package demo;

import java.util.ArrayList;

/\*\*This class is match a image in database and arrange the result to output

 \*/

public class MatchDB {

 private ArrayList<Featrue> input = null;

 private ArrayList<String> image ;

 private ArrayList pair;

 public MatchDB(ArrayList<Featrue> a){

 this.input = a;

 image = new ArrayList();

 pair = new ArrayList();

 ArrayList<Featrue> first = getResultArray();

 ArrayList<Featrue> second = getArrangeResult(first);

 findImage(second);

 }

 private ArrayList<Featrue> getResultArray(){

 ArrayList<Featrue> array = new ArrayList();

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

 int type = input.get(i).getType();

 int d = input.get(i).getDirection();

 double angle = input.get(i).getAngle();

 double diff = input.get(i).getDifferentAngle();

 ArrayList<Featrue> temp = FeatrueDAO.getFeatrue(type, d, angle, diff);

 if(!temp.isEmpty()){

 for(int j = 0; j<temp.size();j++){

 array.add(temp.get(j));

 }

 }

 }

 return array;

 }

 private ArrayList<Featrue> getArrangeResult(ArrayList<Featrue> a){

 ArrayList<Featrue> new\_array = new ArrayList();

 for(int i = 0; i<a.size()-1;i++){

 if(a.get(i).getMatch()==false){

 for(int j = i+1;j<a.size();j++){

 if(a.get(i).getID()==a.get(j).getID()){

 a.get(j).setMatch(true);

 }

 }

 }

 }

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

 if(a.get(i).getMatch()==false){

 new\_array.add(a.get(i));

 }

 }

 return new\_array;

 }

 public ArrayList<String> getImage(){

 return image;

 }

 public ArrayList getPair(){

 return pair;

 }

 private void findImage(ArrayList<Featrue> a){

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

 if(a.get(i).getMatch()==false){

 String name = a.get(i).getImage();

 int count = 1;

 for(int j = i+1;j<a.size();j++){

 if(a.get(j).getImage().equals(name)){

 count = count+1;

 a.get(j).setMatch(true);

 }

 }

 if(count >=12){

 pair.add(count);

 image.add(a.get(i).getImage());

 }

 }

 }

 }

}

# GUI Frame

The most GUI component code in this part is generated by GUI generated in NetBeans IDE.

## 4.1 FRSMainApp.java

package demo;

import java.awt.Color;

import java.awt.FileDialog;

import java.awt.Graphics;

import java.io.File;

import java.util.ArrayList;

import java.util.Calendar;

import java.util.GregorianCalendar;

import javax.imageio.ImageIO;

import javax.swing.JOptionPane;

import javax.swing.UIManager;

/\*\*Main user interface

 \*/

public class FRSMainApp extends javax.swing.JFrame {

 private FileDialog import\_image;

 private Pixel[][] input\_pixel;

 private Pixel[][] temp\_pixel;

 private Pixel[][] db\_pixel;

 ArrayList<Minutia> input\_minutiae = new ArrayList();

 ArrayList<Minutia> temp\_minutiae = new ArrayList();

 ArrayList<Minutia> db\_m = new ArrayList();

 private Graphics input\_g,temp\_g,db\_image\_g;

 private Point input\_core,temp\_core,core;

 private static String directory;

 private static String image\_name;

 /\*\* Creates new form FPSMainApp \*/

 public FRSMainApp() {

 try {

 UIManager .setLookAndFeel("com.sun.java.swing.plaf.windows.WindowsLookAndFeel");

 } catch (Exception e) {

 e.printStackTrace();

 }

 initComponents();

 this.setTitle("Fingerprint Recognition System");

 this.setResizable(false);

 this.setLocationRelativeTo(null);

 import\_image = new FileDialog (this,"Load Image",FileDialog .LOAD);

 input\_g = input\_image\_panel.getGraphics();

 temp\_g = template\_image\_panel.getGraphics();

 db\_image\_g = db\_pro.getGraphics();

 }

 /\*\* This method is called from within the constructor to

 \* initialize the form.

 \* WARNING: Do NOT modify this code. The content of this method is

 \* always regenerated by the Form Editor.

 \*/

 @SuppressWarnings("unchecked")

 // <editor-fold defaultstate="collapsed" desc="Generated Code">

 private void initComponents() {

 radioButtonGroup = new javax.swing.ButtonGroup();

 mainTebbedPanel = new javax.swing.JTabbedPane();

 FRSPanel = new javax.swing.JPanel();

 inputPanel = new javax.swing.JPanel();

 input = new javax.swing.JRadioButton();

 template = new javax.swing.JRadioButton();

 load = new javax.swing.JButton();

 prePanel = new javax.swing.JPanel();

 normal = new javax.swing.JToggleButton();

 orientation = new javax.swing.JToggleButton();

 enhancement = new javax.swing.JToggleButton();

 ridgeDetection = new javax.swing.JToggleButton();

 thinning = new javax.swing.JToggleButton();

 minutiae = new javax.swing.JToggleButton();

 remove = new javax.swing.JToggleButton();

 rePanel = new javax.swing.JPanel();

 match = new javax.swing.JButton();

 corePoint = new javax.swing.JButton();

 input\_image\_panel = new InputImagePanel();

 template\_image\_panel = new TemplateImagePanel();

 resultPanel = new javax.swing.JPanel();

 result\_1 = new javax.swing.JLabel();

 result\_2 = new javax.swing.JLabel();

 result\_3 = new javax.swing.JLabel();

 result\_4 = new javax.swing.JLabel();

 DBPanel = new javax.swing.JPanel();

 db\_input\_p = new javax.swing.JPanel();

 db\_load = new javax.swing.JButton();

 jPanel2 = new javax.swing.JPanel();

 db\_normal = new javax.swing.JToggleButton();

 db\_orientation = new javax.swing.JToggleButton();

 db\_enhance = new javax.swing.JToggleButton();

 db\_binary = new javax.swing.JToggleButton();

 db\_thinning = new javax.swing.JToggleButton();

 db\_minutiae = new javax.swing.JToggleButton();

 db\_remove = new javax.swing.JToggleButton();

 db\_core = new javax.swing.JButton();

 enroll = new javax.swing.JButton();

 select = new javax.swing.JButton();

 db\_or = new DBOriginalImagePanel();

 db\_pro = new DBProcessedImagePanel();

 jLabel1 = new javax.swing.JLabel();

 jScrollPane1 = new javax.swing.JScrollPane();

 description = new javax.swing.JTextArea();

 jLabel2 = new javax.swing.JLabel();

 name = new javax.swing.JLabel();

 radioButtonGroup.add(input);

 radioButtonGroup.add(template);

 setDefaultCloseOperation(javax.swing.WindowConstants.EXIT\_ON\_CLOSE);

 FRSPanel.setBackground(new java.awt.Color(255, 255, 255));

 inputPanel.setBackground(new java.awt.Color(255, 255, 255));

 inputPanel.setBorder(javax.swing.BorderFactory.createTitledBorder(null, "Input Image", javax.swing.border.TitledBorder.DEFAULT\_JUSTIFICATION, javax.swing.border.TitledBorder.DEFAULT\_POSITION, new java.awt.Font("Calibri", 1, 12))); // NOI18N

 input.setBackground(new java.awt.Color(255, 255, 255));

 input.setText("Input");

 template.setBackground(new java.awt.Color(255, 255, 255));

 template.setText("Template");

 load.setText("Load");

 load.addActionListener(new java.awt.event.ActionListener() {

 public void actionPerformed(java.awt.event.ActionEvent evt) {

 loadActionPerformed(evt);

 }

 });

 javax.swing.GroupLayout inputPanelLayout = new javax.swing.GroupLayout(inputPanel);

 inputPanel.setLayout(inputPanelLayout);

 inputPanelLayout.setHorizontalGroup(

 inputPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

 .addGroup(inputPanelLayout.createSequentialGroup()

 .addGroup(inputPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

 .addComponent(input)

 .addComponent(template))

 .addContainerGap(54, Short.MAX\_VALUE))

 .addComponent(load, javax.swing.GroupLayout.DEFAULT\_SIZE, 127, Short.MAX\_VALUE)

 );

 inputPanelLayout.setVerticalGroup(

 inputPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

 .addGroup(inputPanelLayout.createSequentialGroup()

 .addComponent(input)

 .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)

 .addComponent(template)

 .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)

 .addComponent(load))

 );

 prePanel.setBackground(new java.awt.Color(255, 255, 255));

 prePanel.setBorder(javax.swing.BorderFactory.createTitledBorder(null, "Pre-processing", javax.swing.border.TitledBorder.DEFAULT\_JUSTIFICATION, javax.swing.border.TitledBorder.DEFAULT\_POSITION, new java.awt.Font("Calibri", 1, 12))); // NOI18N

 normal.setText("Normalization");

 normal.addActionListener(new java.awt.event.ActionListener() {

 public void actionPerformed(java.awt.event.ActionEvent evt) {

 normalActionPerformed(evt);

 }

 });

 orientation.setText("Orientation");

 orientation.addActionListener(new java.awt.event.ActionListener() {

 public void actionPerformed(java.awt.event.ActionEvent evt) {

 orientationActionPerformed(evt);

 }

 });

 enhancement.setText("Enhancement");

 enhancement.addActionListener(new java.awt.event.ActionListener() {

 public void actionPerformed(java.awt.event.ActionEvent evt) {

 enhancementActionPerformed(evt);

 }

 });

 ridgeDetection.setText("Binarization");

 ridgeDetection.addActionListener(new java.awt.event.ActionListener() {

 public void actionPerformed(java.awt.event.ActionEvent evt) {

 ridgeDetectionActionPerformed(evt);

 }

 });

 thinning.setText("Thinning");

 thinning.addActionListener(new java.awt.event.ActionListener() {

 public void actionPerformed(java.awt.event.ActionEvent evt) {

 thinningActionPerformed(evt);

 }

 });

 minutiae.setText("Minutiae");

 minutiae.addActionListener(new java.awt.event.ActionListener() {

 public void actionPerformed(java.awt.event.ActionEvent evt) {

 minutiaeActionPerformed(evt);

 }

 });

 remove.setText("Remove false");

 remove.addActionListener(new java.awt.event.ActionListener() {

 public void actionPerformed(java.awt.event.ActionEvent evt) {

 removeActionPerformed(evt);

 }

 });

 javax.swing.GroupLayout prePanelLayout = new javax.swing.GroupLayout(prePanel);

 prePanel.setLayout(prePanelLayout);

 prePanelLayout.setHorizontalGroup(

 prePanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

 .addComponent(remove, javax.swing.GroupLayout.DEFAULT\_SIZE, 127, Short.MAX\_VALUE)

 .addComponent(minutiae, javax.swing.GroupLayout.DEFAULT\_SIZE, 127, Short.MAX\_VALUE)

 .addComponent(thinning, javax.swing.GroupLayout.DEFAULT\_SIZE, 127, Short.MAX\_VALUE)

 .addComponent(ridgeDetection, javax.swing.GroupLayout.DEFAULT\_SIZE, 127, Short.MAX\_VALUE)

 .addComponent(enhancement, javax.swing.GroupLayout.DEFAULT\_SIZE, 127, Short.MAX\_VALUE)

 .addComponent(orientation, javax.swing.GroupLayout.DEFAULT\_SIZE, 127, Short.MAX\_VALUE)

 .addComponent(normal, javax.swing.GroupLayout.DEFAULT\_SIZE, 127, Short.MAX\_VALUE)

 );

 prePanelLayout.setVerticalGroup(

 prePanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

 .addGroup(prePanelLayout.createSequentialGroup()

 .addComponent(normal)

 .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)

 .addComponent(orientation)

 .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)

 .addComponent(enhancement)

 .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)

 .addComponent(ridgeDetection)

 .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)

 .addComponent(thinning)

 .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)

 .addComponent(minutiae)

 .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED, javax.swing.GroupLayout.DEFAULT\_SIZE, Short.MAX\_VALUE)

 .addComponent(remove))

 );

 rePanel.setBackground(new java.awt.Color(255, 255, 255));

 rePanel.setBorder(javax.swing.BorderFactory.createTitledBorder(null, "Recognition", javax.swing.border.TitledBorder.DEFAULT\_JUSTIFICATION, javax.swing.border.TitledBorder.DEFAULT\_POSITION, new java.awt.Font("Calibri", 1, 12))); // NOI18N

 rePanel.setForeground(new java.awt.Color(255, 255, 255));

 match.setText("Match");

 match.addActionListener(new java.awt.event.ActionListener() {

 public void actionPerformed(java.awt.event.ActionEvent evt) {

 matchActionPerformed(evt);

 }

 });

 corePoint.setText("Core point");

 corePoint.addActionListener(new java.awt.event.ActionListener() {

 public void actionPerformed(java.awt.event.ActionEvent evt) {

 corePointActionPerformed(evt);

 }

 });

 javax.swing.GroupLayout rePanelLayout = new javax.swing.GroupLayout(rePanel);

 rePanel.setLayout(rePanelLayout);

 rePanelLayout.setHorizontalGroup(

 rePanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

 .addComponent(corePoint, javax.swing.GroupLayout.DEFAULT\_SIZE, 127, Short.MAX\_VALUE)

 .addComponent(match, javax.swing.GroupLayout.DEFAULT\_SIZE, 127, Short.MAX\_VALUE)

 );

 rePanelLayout.setVerticalGroup(

 rePanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

 .addGroup(rePanelLayout.createSequentialGroup()

 .addComponent(corePoint)

 .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)

 .addComponent(match, javax.swing.GroupLayout.PREFERRED\_SIZE, 52, javax.swing.GroupLayout.PREFERRED\_SIZE)

 .addContainerGap(javax.swing.GroupLayout.DEFAULT\_SIZE, Short.MAX\_VALUE))

 );

 input\_image\_panel.setBackground(new java.awt.Color(255, 255, 255));

 input\_image\_panel.setBorder(javax.swing.BorderFactory.createTitledBorder(null, "Input Image", javax.swing.border.TitledBorder.DEFAULT\_JUSTIFICATION, javax.swing.border.TitledBorder.DEFAULT\_POSITION, new java.awt.Font("Calibri", 1, 18))); // NOI18N

 input\_image\_panel.setPreferredSize(new java.awt.Dimension(300, 340));

 javax.swing.GroupLayout input\_image\_panelLayout = new javax.swing.GroupLayout(input\_image\_panel);

 input\_image\_panel.setLayout(input\_image\_panelLayout);

 input\_image\_panelLayout.setHorizontalGroup(

 input\_image\_panelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

 .addGap(0, 284, Short.MAX\_VALUE)

 );

 input\_image\_panelLayout.setVerticalGroup(

 input\_image\_panelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

 .addGap(0, 340, Short.MAX\_VALUE)

 );

 template\_image\_panel.setBackground(new java.awt.Color(255, 255, 255));

 template\_image\_panel.setBorder(javax.swing.BorderFactory.createTitledBorder(null, "Template Image", javax.swing.border.TitledBorder.DEFAULT\_JUSTIFICATION, javax.swing.border.TitledBorder.DEFAULT\_POSITION, new java.awt.Font("Calibri", 1, 18))); // NOI18N

 template\_image\_panel.setPreferredSize(new java.awt.Dimension(300, 340));

 javax.swing.GroupLayout template\_image\_panelLayout = new javax.swing.GroupLayout(template\_image\_panel);

 template\_image\_panel.setLayout(template\_image\_panelLayout);

 template\_image\_panelLayout.setHorizontalGroup(

 template\_image\_panelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

 .addGap(0, 284, Short.MAX\_VALUE)

 );

 template\_image\_panelLayout.setVerticalGroup(

 template\_image\_panelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

 .addGap(0, 340, Short.MAX\_VALUE)

 );

 resultPanel.setBackground(new java.awt.Color(255, 255, 255));

 resultPanel.setBorder(javax.swing.BorderFactory.createTitledBorder(null, "Match Result", javax.swing.border.TitledBorder.DEFAULT\_JUSTIFICATION, javax.swing.border.TitledBorder.DEFAULT\_POSITION, new java.awt.Font("Calibri", 1, 18))); // NOI18N

 result\_1.setFont(new java.awt.Font("Calibri", 1, 18));

 result\_2.setFont(new java.awt.Font("Calibri", 1, 18));

 result\_3.setFont(new java.awt.Font("Calibri", 1, 18));

 result\_4.setFont(new java.awt.Font("Calibri", 1, 18));

 javax.swing.GroupLayout resultPanelLayout = new javax.swing.GroupLayout(resultPanel);

 resultPanel.setLayout(resultPanelLayout);

 resultPanelLayout.setHorizontalGroup(

 resultPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

 .addGroup(javax.swing.GroupLayout.Alignment.TRAILING, resultPanelLayout.createSequentialGroup()

 .addContainerGap()

 .addGroup(resultPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRAILING)

 .addComponent(result\_1, javax.swing.GroupLayout.Alignment.LEADING, javax.swing.GroupLayout.DEFAULT\_SIZE, 570, Short.MAX\_VALUE)

 .addComponent(result\_2, javax.swing.GroupLayout.Alignment.LEADING, javax.swing.GroupLayout.DEFAULT\_SIZE, 570, Short.MAX\_VALUE)

 .addComponent(result\_3, javax.swing.GroupLayout.Alignment.LEADING, javax.swing.GroupLayout.DEFAULT\_SIZE, 570, Short.MAX\_VALUE)

 .addComponent(result\_4, javax.swing.GroupLayout.Alignment.LEADING, javax.swing.GroupLayout.DEFAULT\_SIZE, 570, Short.MAX\_VALUE))

 .addContainerGap())

 );

 resultPanelLayout.setVerticalGroup(

 resultPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

 .addGroup(resultPanelLayout.createSequentialGroup()

 .addComponent(result\_1, javax.swing.GroupLayout.PREFERRED\_SIZE, 29, javax.swing.GroupLayout.PREFERRED\_SIZE)

 .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)

 .addComponent(result\_2, javax.swing.GroupLayout.PREFERRED\_SIZE, 27, javax.swing.GroupLayout.PREFERRED\_SIZE)

 .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)

 .addComponent(result\_3, javax.swing.GroupLayout.DEFAULT\_SIZE, 26, Short.MAX\_VALUE)

 .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)

 .addComponent(result\_4, javax.swing.GroupLayout.PREFERRED\_SIZE, 25, javax.swing.GroupLayout.PREFERRED\_SIZE))

 );

 javax.swing.GroupLayout FRSPanelLayout = new javax.swing.GroupLayout(FRSPanel);

 FRSPanel.setLayout(FRSPanelLayout);

 FRSPanelLayout.setHorizontalGroup(

 FRSPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

 .addGroup(FRSPanelLayout.createSequentialGroup()

 .addGroup(FRSPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING, false)

 .addComponent(inputPanel, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, Short.MAX\_VALUE)

 .addComponent(rePanel, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, Short.MAX\_VALUE)

 .addComponent(prePanel, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, Short.MAX\_VALUE))

 .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED, 16, Short.MAX\_VALUE)

 .addGroup(FRSPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

 .addGroup(FRSPanelLayout.createSequentialGroup()

 .addComponent(input\_image\_panel, javax.swing.GroupLayout.PREFERRED\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.PREFERRED\_SIZE)

 .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)

 .addComponent(template\_image\_panel, javax.swing.GroupLayout.PREFERRED\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.PREFERRED\_SIZE)

 .addGap(10, 10, 10))

 .addGroup(FRSPanelLayout.createSequentialGroup()

 .addComponent(resultPanel, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, Short.MAX\_VALUE)

 .addContainerGap())))

 );

 FRSPanelLayout.setVerticalGroup(

 FRSPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

 .addGroup(FRSPanelLayout.createSequentialGroup()

 .addGroup(FRSPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

 .addGroup(FRSPanelLayout.createSequentialGroup()

 .addComponent(inputPanel, javax.swing.GroupLayout.PREFERRED\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.PREFERRED\_SIZE)

 .addGap(2, 2, 2)

 .addComponent(prePanel, javax.swing.GroupLayout.PREFERRED\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.PREFERRED\_SIZE)

 .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)

 .addComponent(rePanel, javax.swing.GroupLayout.PREFERRED\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.PREFERRED\_SIZE))

 .addGroup(FRSPanelLayout.createSequentialGroup()

 .addGroup(FRSPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRAILING, false)

 .addComponent(template\_image\_panel, javax.swing.GroupLayout.Alignment.LEADING, javax.swing.GroupLayout.DEFAULT\_SIZE, 375, Short.MAX\_VALUE)

 .addComponent(input\_image\_panel, javax.swing.GroupLayout.Alignment.LEADING, javax.swing.GroupLayout.DEFAULT\_SIZE, 375, Short.MAX\_VALUE))

 .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)

 .addComponent(resultPanel, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, Short.MAX\_VALUE)))

 .addContainerGap())

 );

 mainTebbedPanel.addTab("FingerprintRecognition", FRSPanel);

 DBPanel.setBackground(new java.awt.Color(255, 255, 255));

 db\_input\_p.setBackground(new java.awt.Color(255, 255, 255));

 db\_input\_p.setBorder(javax.swing.BorderFactory.createTitledBorder(null, "Input Image", javax.swing.border.TitledBorder.DEFAULT\_JUSTIFICATION, javax.swing.border.TitledBorder.DEFAULT\_POSITION, new java.awt.Font("Calibri", 1, 12))); // NOI18N

 db\_load.setText("Load");

 db\_load.addActionListener(new java.awt.event.ActionListener() {

 public void actionPerformed(java.awt.event.ActionEvent evt) {

 db\_loadActionPerformed(evt);

 }

 });

 javax.swing.GroupLayout db\_input\_pLayout = new javax.swing.GroupLayout(db\_input\_p);

 db\_input\_p.setLayout(db\_input\_pLayout);

 db\_input\_pLayout.setHorizontalGroup(

 db\_input\_pLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

 .addComponent(db\_load, javax.swing.GroupLayout.DEFAULT\_SIZE, 132, Short.MAX\_VALUE)

 );

 db\_input\_pLayout.setVerticalGroup(

 db\_input\_pLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

 .addGroup(db\_input\_pLayout.createSequentialGroup()

 .addComponent(db\_load)

 .addContainerGap(javax.swing.GroupLayout.DEFAULT\_SIZE, Short.MAX\_VALUE))

 );

 jPanel2.setBackground(new java.awt.Color(255, 255, 255));

 jPanel2.setBorder(javax.swing.BorderFactory.createTitledBorder(null, "Pre-processing", javax.swing.border.TitledBorder.DEFAULT\_JUSTIFICATION, javax.swing.border.TitledBorder.DEFAULT\_POSITION, new java.awt.Font("Calibri", 1, 12))); // NOI18N

 db\_normal.setText("Normalization");

 db\_normal.addActionListener(new java.awt.event.ActionListener() {

 public void actionPerformed(java.awt.event.ActionEvent evt) {

 db\_normalActionPerformed(evt);

 }

 });

 db\_orientation.setText("Orientation");

 db\_orientation.addActionListener(new java.awt.event.ActionListener() {

 public void actionPerformed(java.awt.event.ActionEvent evt) {

 db\_orientationActionPerformed(evt);

 }

 });

 db\_enhance.setText("Enhancement");

 db\_enhance.addActionListener(new java.awt.event.ActionListener() {

 public void actionPerformed(java.awt.event.ActionEvent evt) {

 db\_enhanceActionPerformed(evt);

 }

 });

 db\_binary.setText("Binarization");

 db\_binary.addActionListener(new java.awt.event.ActionListener() {

 public void actionPerformed(java.awt.event.ActionEvent evt) {

 db\_binaryActionPerformed(evt);

 }

 });

 db\_thinning.setText("Thinning");

 db\_thinning.addActionListener(new java.awt.event.ActionListener() {

 public void actionPerformed(java.awt.event.ActionEvent evt) {

 db\_thinningActionPerformed(evt);

 }

 });

 db\_minutiae.setText("Minutiae");

 db\_minutiae.addActionListener(new java.awt.event.ActionListener() {

 public void actionPerformed(java.awt.event.ActionEvent evt) {

 db\_minutiaeActionPerformed(evt);

 }

 });

 db\_remove.setText("Remove false");

 db\_remove.addActionListener(new java.awt.event.ActionListener() {

 public void actionPerformed(java.awt.event.ActionEvent evt) {

 db\_removeActionPerformed(evt);

 }

 });

 db\_core.setText("Core point");

 db\_core.addActionListener(new java.awt.event.ActionListener() {

 public void actionPerformed(java.awt.event.ActionEvent evt) {

 db\_coreActionPerformed(evt);

 }

 });

 javax.swing.GroupLayout jPanel2Layout = new javax.swing.GroupLayout(jPanel2);

 jPanel2.setLayout(jPanel2Layout);

 jPanel2Layout.setHorizontalGroup(

 jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

 .addComponent(db\_thinning, javax.swing.GroupLayout.DEFAULT\_SIZE, 132, Short.MAX\_VALUE)

 .addComponent(db\_binary, javax.swing.GroupLayout.DEFAULT\_SIZE, 132, Short.MAX\_VALUE)

 .addComponent(db\_enhance, javax.swing.GroupLayout.DEFAULT\_SIZE, 132, Short.MAX\_VALUE)

 .addComponent(db\_orientation, 0, 0, Short.MAX\_VALUE)

 .addComponent(db\_normal, javax.swing.GroupLayout.DEFAULT\_SIZE, 132, Short.MAX\_VALUE)

 .addComponent(db\_core, javax.swing.GroupLayout.DEFAULT\_SIZE, 132, Short.MAX\_VALUE)

 .addComponent(db\_minutiae, javax.swing.GroupLayout.DEFAULT\_SIZE, 132, Short.MAX\_VALUE)

 .addComponent(db\_remove, javax.swing.GroupLayout.DEFAULT\_SIZE, 132, Short.MAX\_VALUE)

 );

 jPanel2Layout.setVerticalGroup(

 jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

 .addGroup(jPanel2Layout.createSequentialGroup()

 .addComponent(db\_normal)

 .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)

 .addComponent(db\_orientation)

 .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)

 .addComponent(db\_enhance)

 .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)

 .addComponent(db\_binary)

 .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)

 .addComponent(db\_thinning)

 .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)

 .addComponent(db\_minutiae)

 .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)

 .addComponent(db\_remove)

 .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED, javax.swing.GroupLayout.DEFAULT\_SIZE, Short.MAX\_VALUE)

 .addComponent(db\_core))

 );

 enroll.setText("Enroll");

 enroll.addActionListener(new java.awt.event.ActionListener() {

 public void actionPerformed(java.awt.event.ActionEvent evt) {

 enrollActionPerformed(evt);

 }

 });

 select.setText("Match In DB");

 select.addActionListener(new java.awt.event.ActionListener() {

 public void actionPerformed(java.awt.event.ActionEvent evt) {

 selectActionPerformed(evt);

 }

 });

 db\_or.setBackground(new java.awt.Color(255, 255, 255));

 db\_or.setBorder(javax.swing.BorderFactory.createTitledBorder(null, "Original Image", javax.swing.border.TitledBorder.DEFAULT\_JUSTIFICATION, javax.swing.border.TitledBorder.DEFAULT\_POSITION, new java.awt.Font("Calibri", 1, 18))); // NOI18N

 db\_or.setPreferredSize(new java.awt.Dimension(300, 340));

 javax.swing.GroupLayout db\_orLayout = new javax.swing.GroupLayout(db\_or);

 db\_or.setLayout(db\_orLayout);

 db\_orLayout.setHorizontalGroup(

 db\_orLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

 .addGap(0, 284, Short.MAX\_VALUE)

 );

 db\_orLayout.setVerticalGroup(

 db\_orLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

 .addGap(0, 295, Short.MAX\_VALUE)

 );

 db\_pro.setBackground(new java.awt.Color(255, 255, 255));

 db\_pro.setBorder(javax.swing.BorderFactory.createTitledBorder(null, "Processed Image", javax.swing.border.TitledBorder.DEFAULT\_JUSTIFICATION, javax.swing.border.TitledBorder.DEFAULT\_POSITION, new java.awt.Font("Calibri", 1, 18))); // NOI18N

 db\_pro.setPreferredSize(new java.awt.Dimension(300, 340));

 javax.swing.GroupLayout db\_proLayout = new javax.swing.GroupLayout(db\_pro);

 db\_pro.setLayout(db\_proLayout);

 db\_proLayout.setHorizontalGroup(

 db\_proLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

 .addGap(0, 284, Short.MAX\_VALUE)

 );

 db\_proLayout.setVerticalGroup(

 db\_proLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

 .addGap(0, 295, Short.MAX\_VALUE)

 );

 jLabel1.setText("Description:");

 description.setColumns(20);

 description.setRows(5);

 jScrollPane1.setViewportView(description);

 jLabel2.setText("Image name:");

 javax.swing.GroupLayout DBPanelLayout = new javax.swing.GroupLayout(DBPanel);

 DBPanel.setLayout(DBPanelLayout);

 DBPanelLayout.setHorizontalGroup(

 DBPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

 .addGroup(javax.swing.GroupLayout.Alignment.TRAILING, DBPanelLayout.createSequentialGroup()

 .addGroup(DBPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

 .addGroup(DBPanelLayout.createSequentialGroup()

 .addContainerGap()

 .addComponent(select, javax.swing.GroupLayout.DEFAULT\_SIZE, 138, Short.MAX\_VALUE))

 .addGroup(DBPanelLayout.createSequentialGroup()

 .addContainerGap()

 .addComponent(enroll, javax.swing.GroupLayout.DEFAULT\_SIZE, 138, Short.MAX\_VALUE))

 .addComponent(db\_input\_p, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, Short.MAX\_VALUE)

 .addComponent(jPanel2, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, Short.MAX\_VALUE))

 .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)

 .addGroup(DBPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

 .addComponent(name, javax.swing.GroupLayout.PREFERRED\_SIZE, 192, javax.swing.GroupLayout.PREFERRED\_SIZE)

 .addComponent(jLabel2, javax.swing.GroupLayout.DEFAULT\_SIZE, 611, Short.MAX\_VALUE)

 .addGroup(javax.swing.GroupLayout.Alignment.TRAILING, DBPanelLayout.createSequentialGroup()

 .addComponent(db\_or, javax.swing.GroupLayout.PREFERRED\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.PREFERRED\_SIZE)

 .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)

 .addComponent(db\_pro, javax.swing.GroupLayout.PREFERRED\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.PREFERRED\_SIZE))

 .addComponent(jScrollPane1, javax.swing.GroupLayout.DEFAULT\_SIZE, 611, Short.MAX\_VALUE)

 .addComponent(jLabel1, javax.swing.GroupLayout.PREFERRED\_SIZE, 81, javax.swing.GroupLayout.PREFERRED\_SIZE))

 .addContainerGap())

 );

 DBPanelLayout.setVerticalGroup(

 DBPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

 .addGroup(DBPanelLayout.createSequentialGroup()

 .addGroup(DBPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRAILING)

 .addComponent(db\_pro, javax.swing.GroupLayout.Alignment.LEADING, javax.swing.GroupLayout.DEFAULT\_SIZE, 330, Short.MAX\_VALUE)

 .addGroup(javax.swing.GroupLayout.Alignment.LEADING, DBPanelLayout.createSequentialGroup()

 .addComponent(db\_input\_p, javax.swing.GroupLayout.PREFERRED\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.PREFERRED\_SIZE)

 .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)

 .addComponent(jPanel2, javax.swing.GroupLayout.PREFERRED\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.PREFERRED\_SIZE))

 .addComponent(db\_or, javax.swing.GroupLayout.Alignment.LEADING, javax.swing.GroupLayout.DEFAULT\_SIZE, 330, Short.MAX\_VALUE))

 .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED, 10, javax.swing.GroupLayout.PREFERRED\_SIZE)

 .addComponent(jLabel2)

 .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)

 .addComponent(name, javax.swing.GroupLayout.PREFERRED\_SIZE, 25, javax.swing.GroupLayout.PREFERRED\_SIZE)

 .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)

 .addComponent(jLabel1, javax.swing.GroupLayout.PREFERRED\_SIZE, 25, javax.swing.GroupLayout.PREFERRED\_SIZE)

 .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)

 .addGroup(DBPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING, false)

 .addGroup(javax.swing.GroupLayout.Alignment.TRAILING, DBPanelLayout.createSequentialGroup()

 .addComponent(enroll, javax.swing.GroupLayout.PREFERRED\_SIZE, 35, javax.swing.GroupLayout.PREFERRED\_SIZE)

 .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED, javax.swing.GroupLayout.DEFAULT\_SIZE, Short.MAX\_VALUE)

 .addComponent(select, javax.swing.GroupLayout.PREFERRED\_SIZE, 36, javax.swing.GroupLayout.PREFERRED\_SIZE))

 .addComponent(jScrollPane1, javax.swing.GroupLayout.Alignment.TRAILING, javax.swing.GroupLayout.PREFERRED\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.PREFERRED\_SIZE))

 .addGap(26, 26, 26))

 );

 mainTebbedPanel.addTab("FingerprintDatabaseManagement", DBPanel);

 javax.swing.GroupLayout layout = new javax.swing.GroupLayout(getContentPane());

 getContentPane().setLayout(layout);

 layout.setHorizontalGroup(

 layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

 .addComponent(mainTebbedPanel, javax.swing.GroupLayout.DEFAULT\_SIZE, 780, Short.MAX\_VALUE)

 );

 layout.setVerticalGroup(

 layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

 .addGroup(layout.createSequentialGroup()

 .addComponent(mainTebbedPanel, javax.swing.GroupLayout.DEFAULT\_SIZE, 584, Short.MAX\_VALUE)

 .addContainerGap())

 );

 pack();

 }// </editor-fold>

 private void loadActionPerformed(java.awt.event.ActionEvent evt) {

 if(input.isSelected()){

 loadInputImage();

 input\_image\_panel.updateUI();

 }else if(template.isSelected()){

 loadTemplateImage();

 template\_image\_panel.updateUI();

 }

 initResultPanel();

 }

 private void normalActionPerformed(java.awt.event.ActionEvent evt) {

 if(InputImage.original\_image==null||TemplateImage.original\_image==null){

 JOptionPane.showMessageDialog(null, "You must import input image and template image both");

 }else{

 //normalization

 Normalization normal\_input = new Normalization(InputImage.original\_image);

 Normalization normal\_template = new Normalization(TemplateImage.original\_image);

 InputImage.normalized\_image = normal\_input.getNormalizedImage();

 TemplateImage.normalized\_image = normal\_template.getNormalizedImage();

 //Gaussian smoonth

 Gaussian input\_ga = new Gaussian(InputImage.normalized\_image);

 Gaussian temp\_ga = new Gaussian(TemplateImage.normalized\_image);

 InputImage.normalized\_image = input\_ga.getSmoothImage();

 TemplateImage.normalized\_image = temp\_ga.getSmoothImage();

 input\_image\_panel.updateUI();

 template\_image\_panel.updateUI();

 }

 }

 private void orientationActionPerformed(java.awt.event.ActionEvent evt) {

 OrientationEstimation input\_oe = new OrientationEstimation(InputImage.normalized\_image);

 OrientationEstimation temp\_oe = new OrientationEstimation(TemplateImage.normalized\_image);

 input\_pixel = input\_oe.getOrientationField();

 temp\_pixel = temp\_oe.getOrientationField();

 drawInputOrientation(input\_pixel);

 drawTemplateOrientation(temp\_pixel);

 }

 private void enhancementActionPerformed(java.awt.event.ActionEvent evt) {

 EdgeDetection input\_ed = new EdgeDetection(InputImage.normalized\_image,input\_pixel);

 EdgeDetection temp\_ed = new EdgeDetection(TemplateImage.normalized\_image,temp\_pixel);

 InputImage.enhanced\_image = input\_ed.getEnhancementImage();

 TemplateImage.enhanced\_image = temp\_ed.getEnhancementImage();

 input\_pixel = input\_ed.getPixelMatrix();

 temp\_pixel = temp\_ed.getPixelMatrix();

 input\_image\_panel.updateUI();

 template\_image\_panel.updateUI();

 }

 private void ridgeDetectionActionPerformed(java.awt.event.ActionEvent evt) {

 //ridge detection

 RidgeDetection input\_rd = new RidgeDetection(InputImage.enhanced\_image,input\_pixel);

 RidgeDetection temp\_rd = new RidgeDetection(TemplateImage.enhanced\_image,temp\_pixel);

 InputImage.binarization\_image = input\_rd.getRidge();

 TemplateImage.binarization\_image = temp\_rd.getRidge();

 input\_pixel = input\_rd.getBinaryMatrix();

 temp\_pixel = temp\_rd.getBinaryMatrix();

 //fix ridge

 FixRidge input\_fix = new FixRidge(InputImage.binarization\_image,input\_pixel);

 FixRidge temp\_fix = new FixRidge(TemplateImage.binarization\_image,temp\_pixel);

 InputImage.binarization\_image = input\_fix.getFixRidgeImage();

 TemplateImage.binarization\_image = temp\_fix.getFixRidgeImage();

 input\_pixel = input\_fix.getBinaryMatrix();

 temp\_pixel = temp\_fix.getBinaryMatrix();

 input\_image\_panel.updateUI();

 template\_image\_panel.updateUI();

 }

 private void thinningActionPerformed(java.awt.event.ActionEvent evt) {

 Thinning input\_th = new Thinning(InputImage.binarization\_image,input\_pixel);

 Thinning temp\_th = new Thinning(TemplateImage.binarization\_image,temp\_pixel);

 InputImage.thinning\_image = input\_th.getThinnedImage();

 TemplateImage.thinning\_image = temp\_th.getThinnedImage();

 input\_pixel = input\_th.getThinningMatrix();

 temp\_pixel = temp\_th.getThinningMatrix();

 input\_image\_panel.updateUI();

 template\_image\_panel.updateUI();

 }

 private void minutiaeActionPerformed(java.awt.event.ActionEvent evt) {

 MinutiaeExtraction input\_me = new MinutiaeExtraction(InputImage.thinning\_image,input\_pixel);

 MinutiaeExtraction temp\_me = new MinutiaeExtraction(TemplateImage.thinning\_image,temp\_pixel);

 input\_minutiae = input\_me.getMinutiae();

 temp\_minutiae = temp\_me.getMinutiae();

 InputImage.minutiae = input\_minutiae;

 TemplateImage.minutiae = temp\_minutiae;

 InputImagePanel.showMinutiae = true;

 TemplateImagePanel.showMinutiae = true;

 input\_image\_panel.updateUI();

 template\_image\_panel.updateUI();

 }

 private void removeActionPerformed(java.awt.event.ActionEvent evt) {

 RemoveFalseMinutiae input\_remove = new RemoveFalseMinutiae(InputImage.thinning\_image,input\_minutiae,input\_pixel);

 RemoveFalseMinutiae temp\_remove = new RemoveFalseMinutiae(TemplateImage.thinning\_image,temp\_minutiae,temp\_pixel);

 input\_minutiae = input\_remove.getRemovalMinutiae();

 temp\_minutiae = temp\_remove.getRemovalMinutiae();

 InputImage.minutiae = input\_minutiae;

 TemplateImage.minutiae = temp\_minutiae;

 InputImagePanel.showMinutiae = true;

 TemplateImagePanel.showMinutiae=true;

 input\_image\_panel.updateUI();

 template\_image\_panel.updateUI();

 }

 private void corePointActionPerformed(java.awt.event.ActionEvent evt) {

 PointDirection i\_p = new PointDirection(InputImage.normalized\_image);

 PointDirection t\_p = new PointDirection(TemplateImage.normalized\_image);

 Point[][] i\_point = i\_p.p;

 Point[][] t\_point = t\_p.p;

 FindCore input\_fc = new FindCore(InputImage.normalized\_image,i\_point);

 FindCore temp\_fc = new FindCore(TemplateImage.normalized\_image,t\_point);

 input\_core = input\_fc.getCore();

 temp\_core = temp\_fc.getCore();

 InputImage.core = input\_core;

 TemplateImage.core = temp\_core;

 System.out.println(input\_pixel[input\_core.getX()][input\_core.getY()].getAngle());

 System.out.println(temp\_pixel[temp\_core.getX()][temp\_core.getY()].getAngle());

 InputImagePanel.showCore = true;

 TemplateImagePanel.showCore = true;

 input\_image\_panel.updateUI();

 template\_image\_panel.updateUI();

 }

 private void matchActionPerformed(java.awt.event.ActionEvent evt) {

 if(TemplateImage.core==null||InputImage.core ==null){

 JOptionPane.showMessageDialog(null, "You cannot start matching until find out the core point");

 }else{

 GenerateFeatrue input\_gf = new GenerateFeatrue(input\_core,input\_minutiae);

 GenerateFeatrue temp\_gf = new GenerateFeatrue(temp\_core,temp\_minutiae);

 ArrayList<Featrue> input\_list = input\_gf.getMathFeatrueList();

 ArrayList<Featrue> temp\_list = temp\_gf.getMathFeatrueList();

 Match m = new Match(input\_list,temp\_list,input\_core,temp\_core); int no\_input = input\_list.size();

 int no\_temp = temp\_list.size();

 boolean b = m.isMatch();

 int pair = m.getPair();

 ArrayList<Featrue> f = m.getSameArray();

 outPutResult(no\_input,no\_temp,pair,b);

 ArrayList<Minutia> input\_same = new ArrayList();

 ArrayList<Minutia> temp\_same = new ArrayList();

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

 if(i%2==0){

 input\_same.add(f.get(i).getMinutia());

 }else{

 temp\_same.add(f.get(i).getMinutia());

 }

 }

 InputImage.same = input\_same;

 TemplateImage.same = temp\_same;

 InputImagePanel.showSame = true;

 TemplateImagePanel.showSame = true;

 InputImagePanel.showMinutiae = false;

 TemplateImagePanel.showMinutiae = false;

 input\_image\_panel.updateUI();

 template\_image\_panel.updateUI();

 initToggleButton();

 }

 }

 private void db\_loadActionPerformed(java.awt.event.ActionEvent evt) {

 loadDBImage();

 db\_or.updateUI();

 db\_pro.updateUI();

 }

 private void db\_normalActionPerformed(java.awt.event.ActionEvent evt) {

 if(DBProcessedImage.original\_image==null){

 JOptionPane.showMessageDialog(null, "You must input image first");

 }else{

 Normalization normal\_db = new Normalization(DBProcessedImage.original\_image);

 DBProcessedImage.normalized\_image = normal\_db.getNormalizedImage();

 Gaussian g\_db = new Gaussian(DBProcessedImage.normalized\_image);

 DBProcessedImage.normalized\_image = g\_db.getSmoothImage();

 db\_pro.updateUI();

 }

 }

 private void db\_orientationActionPerformed(java.awt.event.ActionEvent evt) {

 OrientationEstimation db\_oe = new OrientationEstimation(DBProcessedImage.normalized\_image);

 db\_pixel = db\_oe.getOrientationField();

 drawDBOrientation(db\_pixel);

 }

 private void db\_enhanceActionPerformed(java.awt.event.ActionEvent evt) {

 EdgeDetection db\_ed = new EdgeDetection(DBProcessedImage.normalized\_image,db\_pixel);

 DBProcessedImage.enhanced\_image = db\_ed.getEnhancementImage();

 db\_pixel = db\_ed.getPixelMatrix();

 db\_pro.updateUI();

 }

 private void db\_binaryActionPerformed(java.awt.event.ActionEvent evt) {

 //ridge detection

 RidgeDetection db\_rd = new RidgeDetection(DBProcessedImage.enhanced\_image,db\_pixel);

 DBProcessedImage.binarization\_image = db\_rd.getRidge();

 db\_pixel = db\_rd.getBinaryMatrix();

 //fix ridge

 FixRidge db\_fix = new FixRidge(DBProcessedImage.binarization\_image,db\_pixel);

 DBProcessedImage.binarization\_image = db\_fix.getFixRidgeImage();

 db\_pixel = db\_fix.getBinaryMatrix();

 db\_pro.updateUI();

 }

 private void db\_thinningActionPerformed(java.awt.event.ActionEvent evt) {

 Thinning db\_th = new Thinning(DBProcessedImage.binarization\_image,db\_pixel);

 DBProcessedImage.thinning\_image = db\_th.getThinnedImage();

 db\_pixel = db\_th.getThinningMatrix();

 db\_pro.updateUI();

 }

 private void db\_minutiaeActionPerformed(java.awt.event.ActionEvent evt) {

 MinutiaeExtraction db\_me = new MinutiaeExtraction(DBProcessedImage.thinning\_image,db\_pixel);

 db\_m = db\_me.getMinutiae();

 DBProcessedImage.minutiae = db\_m;

 DBProcessedImagePanel.showMinutiae = true;

 db\_pro.updateUI();

 }

 private void db\_removeActionPerformed(java.awt.event.ActionEvent evt) {

 RemoveFalseMinutiae db\_r = new RemoveFalseMinutiae(DBProcessedImage.thinning\_image,db\_m,db\_pixel);

 db\_m = db\_r.getRemovalMinutiae();

 DBProcessedImage.minutiae = db\_m;

 DBProcessedImagePanel.showMinutiae = true;

 db\_pro.updateUI();

 }

 private void db\_coreActionPerformed(java.awt.event.ActionEvent evt) {

 PointDirection db\_p = new PointDirection(DBProcessedImage.normalized\_image);

 Point[][] db\_point = db\_p.p;

 FindCore db\_fc = new FindCore(DBProcessedImage.normalized\_image,db\_point);

 core = db\_fc.getCore();

 DBProcessedImage.core = core;

 DBProcessedImagePanel.showCore = true;

 db\_pro.updateUI();

 }

 private void enrollActionPerformed(java.awt.event.ActionEvent evt) {

 String id = name.getText();

 String direc = directory;

 String de = description.getText();

 int x = core.getX();

 int y = core.getY();

 double core\_angle = db\_pixel[x][y].getAngle();

 FingerprintImage fi = new FingerprintImage(id,direc,de,x,y,core\_angle);

 FingerprintImageDAO.saveImage(fi);

 GenerateFeatrue db\_gf = new GenerateFeatrue(core,db\_m);

 ArrayList<Featrue> db\_array = db\_gf.getMathFeatrueList();

 int f\_id = FeatrueDAO.getMaxId();

 for(int i = 0; i<db\_array.size();i++){

 f\_id = f\_id+1;

 FeatrueDAO.enrollFeatrue(db\_array.get(i), f\_id, image\_name);

 }

 JOptionPane.showMessageDialog(null, "Enroll fingerprint completed!");

 }

 private void selectActionPerformed(java.awt.event.ActionEvent evt) {

 if(DBProcessedImage.minutiae.isEmpty()){

 JOptionPane.showMessageDialog(null, "You cannot start matching until your preprocessing finished");

 }else{

 GenerateFeatrue db\_gf = new GenerateFeatrue(core,db\_m);

 ArrayList<Featrue> db\_array = db\_gf.getMathFeatrueList();

 MatchDB db\_match = new MatchDB(db\_array);

 ArrayList<String> s = db\_match.getImage();

 ArrayList p = db\_match.getPair();

 if(s.isEmpty()){

 JOptionPane.showMessageDialog(null, "There are no similar fingerprints in database");

 }else{

 ResultFrame rf = new ResultFrame(s,p);

 }

 }

 }

 private void drawInputOrientation(Pixel[][] p){

 input\_image\_panel.update(input\_g);

 int y = 5;

 while(y+5<InputImage.original\_image.getHeight()-2){

 int x = 5;

 while(x+5<InputImage.original\_image.getWidth()-6){

 int i = (int) (5 \* Math.cos(p[x][y].getAngle()))+x;

 int j = (int) (5 \* Math.sin(p[x][y].getAngle()))+y;

 input\_g.setColor(Color.WHITE);

 input\_g.drawLine(x+22, y+22, i+22, j+22);

 x= x+11;

 }

 y = y+11;

 }

 }

 private void drawTemplateOrientation(Pixel[][] p){

 template\_image\_panel.update(temp\_g);

 int y = 5;

 while(y+5<TemplateImage.original\_image.getHeight()-2){

 int x = 5;

 while(x+5<TemplateImage.original\_image.getWidth()-6){

 int i = (int) (5 \* Math.cos(p[x][y].getAngle()))+x;

 int j = (int) (5 \* Math.sin(p[x][y].getAngle()))+y;

 temp\_g.setColor(Color.WHITE);

 temp\_g.drawLine(x+22, y+22, i+22, j+22);

 x= x+11;

 }

 y = y+11;

 }

 }

 private void drawDBOrientation(Pixel[][] p){

 db\_pro.update(db\_image\_g);

 int y = 5;

 while(y+5<DBProcessedImage.original\_image.getHeight()-2){

 int x = 5;

 while(x+5<DBProcessedImage.original\_image.getWidth()-6){

 int i = (int) (5 \* Math.cos(p[x][y].getAngle()))+x;

 int j = (int) (5 \* Math.sin(p[x][y].getAngle()))+y;

 db\_image\_g.setColor(Color.WHITE);

 db\_image\_g.drawLine(x+22, y+22, i+22, j+22);

 x= x+11;

 }

 y = y+11;

 }

 }

 private void loadInputImage(){

 InputImage.original\_image=null;

 InputImage.normalized\_image=null;

 InputImage.enhanced\_image=null;

 InputImage.binarization\_image=null;

 InputImage.thinning\_image=null;

 InputImage.core=null;

 InputImagePanel.showCore = false;

 InputImagePanel.showMinutiae = false;

 InputImagePanel.showSame = false;

 import\_image.setVisible(true);

 try{

 File file=new File(import\_image.getDirectory(),import\_image.getFile());

 InputImage.original\_image = ImageIO.read(file);

 }catch(Exception ex){

 ex.printStackTrace();

 }

 initToggleButton();

 }

 private void loadTemplateImage(){

 TemplateImage.original\_image=null;

 TemplateImage.normalized\_image=null;

 TemplateImage.enhanced\_image=null;

 TemplateImage.binarization\_image=null;

 TemplateImage.thinning\_image=null;

 TemplateImage.core=null;

 TemplateImagePanel.showCore = false;

 TemplateImagePanel.showMinutiae = false;

 TemplateImagePanel.showSame = false;

 import\_image.setVisible(true);

 try{

 File file=new File(import\_image.getDirectory(),import\_image.getFile());

 TemplateImage.original\_image = ImageIO.read(file);

 }catch(Exception ex){

 ex.printStackTrace();

 }

 initToggleButton();

 }

 private void loadDBImage(){

 DBOriginalImage.original\_image=null;

 DBProcessedImage.original\_image=null;

 DBProcessedImage.normalized\_image=null;

 DBProcessedImage.enhanced\_image=null;

 DBProcessedImage.binarization\_image=null;

 DBProcessedImage.thinning\_image=null;

 DBProcessedImagePanel.showCore = false;

 DBProcessedImagePanel.showMinutiae = false;

 import\_image.setVisible(true);

 try{

 File file=new File(import\_image.getDirectory(),import\_image.getFile());

 DBProcessedImage.original\_image = ImageIO.read(file);

 DBOriginalImage.original\_image = ImageIO.read(file);

 }catch(Exception ex){

 ex.printStackTrace();

 }

 directory = import\_image.getDirectory();

 image\_name = import\_image.getFile();

 GregorianCalendar c = new GregorianCalendar();

 int year=c.get(Calendar.YEAR);

 int month=c.get(Calendar.MONTH)+1;

 int day=c.get(Calendar.DAY\_OF\_MONTH);

 directory = directory+image\_name;

 image\_name = image\_name.substring(0, image\_name.length()-4)+"-"+year+"-"+month+"-"+day;

 name.setText(image\_name);

 initDBToggleButton();

 }

 private void initToggleButton(){

 normal.setSelected(false);

 orientation.setSelected(false);

 enhancement.setSelected(false);

 ridgeDetection.setSelected(false);

 thinning.setSelected(false);

 minutiae.setSelected(false);

 remove.setSelected(false);

 }

 private void initDBToggleButton(){

 db\_normal.setSelected(false);

 db\_orientation.setSelected(false);

 db\_enhance.setSelected(false);

 db\_binary.setSelected(false);

 db\_thinning.setSelected(false);

 db\_minutiae.setSelected(false);

 db\_remove.setSelected(false);

 }

 private void initResultPanel(){

 result\_1.setText("");

 result\_2.setText("");

 result\_3.setText("");

 result\_4.setText("");

 resultPanel.updateUI();

 }

 private void outPutResult(int no\_input,int no\_temp,int pair,boolean b){

 result\_1.setText("Number of minutiae in input image: "+no\_input);

 result\_2.setText("Number of minutiae in template image: "+no\_temp);

 result\_3.setText("Same minutia pairs: "+pair);

 if(b){

 result\_4.setText("These two image came from the same fingerprint");

 }else{

 result\_4.setText("These two image are different fingerprint image");

 }

 resultPanel.updateUI();

 }

 /\*\*

 \* @param args the command line arguments

 \*/

 public static void main(String args[]) {

 java.awt.EventQueue.invokeLater(new Runnable() {

 public void run() {

 new FRSMainApp().setVisible(true);

 }

 });

 }

 // Variables declaration - do not modify

 private javax.swing.JPanel DBPanel;

 private javax.swing.JPanel FRSPanel;

 private javax.swing.JButton corePoint;

 private javax.swing.JToggleButton db\_binary;

 private javax.swing.JButton db\_core;

 private javax.swing.JToggleButton db\_enhance;

 private javax.swing.JPanel db\_input\_p;

 private javax.swing.JButton db\_load;

 private javax.swing.JToggleButton db\_minutiae;

 private javax.swing.JToggleButton db\_normal;

 private javax.swing.JPanel db\_or;

 private javax.swing.JToggleButton db\_orientation;

 private javax.swing.JPanel db\_pro;

 private javax.swing.JToggleButton db\_remove;

 private javax.swing.JToggleButton db\_thinning;

 private javax.swing.JTextArea description;

 private javax.swing.JToggleButton enhancement;

 private javax.swing.JButton enroll;

 private javax.swing.JRadioButton input;

 private javax.swing.JPanel inputPanel;

 private javax.swing.JPanel input\_image\_panel;

 private javax.swing.JLabel jLabel1;

 private javax.swing.JLabel jLabel2;

 private javax.swing.JPanel jPanel2;

 private javax.swing.JScrollPane jScrollPane1;

 private javax.swing.JButton load;

 private javax.swing.JTabbedPane mainTebbedPanel;

 private javax.swing.JButton match;

 private javax.swing.JToggleButton minutiae;

 private javax.swing.JLabel name;

 private javax.swing.JToggleButton normal;

 private javax.swing.JToggleButton orientation;

 private javax.swing.JPanel prePanel;

 private javax.swing.ButtonGroup radioButtonGroup;

 private javax.swing.JPanel rePanel;

 private javax.swing.JToggleButton remove;

 private javax.swing.JPanel resultPanel;

 private javax.swing.JLabel result\_1;

 private javax.swing.JLabel result\_2;

 private javax.swing.JLabel result\_3;

 private javax.swing.JLabel result\_4;

 private javax.swing.JToggleButton ridgeDetection;

 private javax.swing.JButton select;

 private javax.swing.JRadioButton template;

 private javax.swing.JPanel template\_image\_panel;

 private javax.swing.JToggleButton thinning;

 // End of variables declaration

}

## 4.2 ResultFrame.java

package demo;

import java.util.ArrayList;

import javax.swing.UIManager;

import javax.swing.table.DefaultTableModel;

/\*\*This frame shows the result list to user

 \*/

public class ResultFrame extends javax.swing.JFrame {

 private ArrayList<String> name;

 private ArrayList pair;

 DefaultTableModel tableModel;

 /\*\* Creates new form ResultFrame \*/

 public ResultFrame(ArrayList<String> s, ArrayList i) {

 this.name = s;

 this.pair = i;

 try {

 UIManager .setLookAndFeel("com.sun.java.swing.plaf.windows.WindowsLookAndFeel");

 } catch (Exception e) {

 e.printStackTrace();

 }

 initComponents();

 this.setVisible(true);

 this.setTitle("Result Frame");

 this.setResizable(false);

 this.setLocationRelativeTo(null);

 DefaultTableModel dtm=new DefaultTableModel( new Object[][]{}, new String[]{"Fingerprint\_ID","Same minutiae pair"} ){

 @Override

 public boolean isCellEditable(int row, int column) {

 return false;

 }

 };

 result.setModel(dtm);

 tableModel=(DefaultTableModel) result.getModel();

 generateRsultTable();

 }

 private void generateRsultTable(){

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

 tableModel.addRow(new Object[]{name.get(i).toString(),pair.get(i)});

 }

 }

 /\*\* This method is called from within the constructor to

 \* initialize the form.

 \* WARNING: Do NOT modify this code. The content of this method is

 \* always regenerated by the Form Editor.

 \*/

 @SuppressWarnings("unchecked")

 // <editor-fold defaultstate="collapsed" desc="Generated Code">

 private void initComponents() {

 jScrollPane1 = new javax.swing.JScrollPane();

 result = new javax.swing.JTable();

 select = new javax.swing.JButton();

 close = new javax.swing.JButton();

 setDefaultCloseOperation(javax.swing.WindowConstants.EXIT\_ON\_CLOSE);

 addWindowListener(new java.awt.event.WindowAdapter() {

 public void windowClosed(java.awt.event.WindowEvent evt) {

 formWindowClosed(evt);

 }

 public void windowClosing(java.awt.event.WindowEvent evt) {

 formWindowClosing(evt);

 }

 });

 result.setModel(new javax.swing.table.DefaultTableModel(

 ));

 jScrollPane1.setViewportView(result);

 select.setText("Select");

 select.addActionListener(new java.awt.event.ActionListener() {

 public void actionPerformed(java.awt.event.ActionEvent evt) {

 selectActionPerformed(evt);

 }

 });

 close.setText("Close");

 close.addActionListener(new java.awt.event.ActionListener() {

 public void actionPerformed(java.awt.event.ActionEvent evt) {

 closeActionPerformed(evt);

 }

 });

 javax.swing.GroupLayout layout = new javax.swing.GroupLayout(getContentPane());

 getContentPane().setLayout(layout);

 layout.setHorizontalGroup(

 layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

 .addGroup(layout.createSequentialGroup()

 .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

 .addGroup(layout.createSequentialGroup()

 .addContainerGap()

 .addComponent(jScrollPane1, javax.swing.GroupLayout.PREFERRED\_SIZE, 375, javax.swing.GroupLayout.PREFERRED\_SIZE))

 .addGroup(javax.swing.GroupLayout.Alignment.TRAILING, layout.createSequentialGroup()

 .addComponent(select)

 .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)

 .addComponent(close, javax.swing.GroupLayout.PREFERRED\_SIZE, 71, javax.swing.GroupLayout.PREFERRED\_SIZE)))

 .addContainerGap(javax.swing.GroupLayout.DEFAULT\_SIZE, Short.MAX\_VALUE))

 );

 layout.setVerticalGroup(

 layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

 .addGroup(layout.createSequentialGroup()

 .addComponent(jScrollPane1, javax.swing.GroupLayout.PREFERRED\_SIZE, 129, javax.swing.GroupLayout.PREFERRED\_SIZE)

 .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)

 .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)

 .addComponent(close)

 .addComponent(select))

 .addContainerGap(javax.swing.GroupLayout.DEFAULT\_SIZE, Short.MAX\_VALUE))

 );

 pack();

 }// </editor-fold>

 private void closeActionPerformed(java.awt.event.ActionEvent evt) {

 this.dispose();

 }

 private void selectActionPerformed(java.awt.event.ActionEvent evt) {

 String image\_id = result.getValueAt(result.getSelectedRow(), 0).toString();

 FingerprintImage fi = FingerprintImageDAO.getImage(image\_id);

 ResultWindow rw = new ResultWindow(fi);

 this.dispose();

 }

 private void formWindowClosed(java.awt.event.WindowEvent evt) {

 }

 private void formWindowClosing(java.awt.event.WindowEvent evt) {

 this.setDefaultCloseOperation(DISPOSE\_ON\_CLOSE);

 }

 // Variables declaration - do not modify

 private javax.swing.JButton close;

 private javax.swing.JScrollPane jScrollPane1;

 private javax.swing.JTable result;

 private javax.swing.JButton select;

 // End of variables declaration

}

## 4.2 ResultWindow.java

package demo;

import java.awt.image.BufferedImage;

import java.io.File;

import javax.imageio.ImageIO;

import javax.swing.UIManager;

/\*\*This Frame shows one of the mathc result in database

 \*/

public class ResultWindow extends javax.swing.JFrame {

 public static BufferedImage result = null;

 private FingerprintImage input;

 /\*\* Creates new form ResultWindow \*/

 public ResultWindow(FingerprintImage fi) {

 this.input = fi;

 try {

 UIManager .setLookAndFeel("com.sun.java.swing.plaf.windows.WindowsLookAndFeel");

 } catch (Exception e) {

 e.printStackTrace();

 }

 initComponents();

 this.setTitle("Result Image");

 this.setVisible(true);

 this.setResizable(false);

 this.setLocationRelativeTo(null);

 getImage();

 initData();

 }

 private void initData(){

 this.id.setText(input.getID());

 this.des.setText(input.getDescription());

 int no = FeatrueDAO.getFeatrueNO(input.getID());

 this.no\_m.setText("There are "+ no + " minutiaes in this image");

 }

 private void getImage(){

 try{

 File file=new File(input.getDirectory());

 result = ImageIO.read(file);

 }catch(Exception ex){

 ex.printStackTrace();

 }

 }

 /\*\* This method is called from within the constructor to

 \* initialize the form.

 \* WARNING: Do NOT modify this code. The content of this method is

 \* always regenerated by the Form Editor.

 \*/

 @SuppressWarnings("unchecked")

 // <editor-fold defaultstate="collapsed" desc="Generated Code">

 private void initComponents() {

 rp\_panel = new ResultImagePanel();

 jLabel1 = new javax.swing.JLabel();

 no\_m = new javax.swing.JLabel();

 jLabel3 = new javax.swing.JLabel();

 jScrollPane1 = new javax.swing.JScrollPane();

 des = new javax.swing.JTextArea();

 delete = new javax.swing.JButton();

 up = new javax.swing.JButton();

 close = new javax.swing.JButton();

 id = new javax.swing.JLabel();

 setDefaultCloseOperation(javax.swing.WindowConstants.EXIT\_ON\_CLOSE);

 setBackground(new java.awt.Color(255, 255, 255));

 addWindowListener(new java.awt.event.WindowAdapter() {

 public void windowClosed(java.awt.event.WindowEvent evt) {

 formWindowClosed(evt);

 }

 public void windowClosing(java.awt.event.WindowEvent evt) {

 formWindowClosing(evt);

 }

 });

 rp\_panel.setBackground(new java.awt.Color(255, 255, 255));

 rp\_panel.setBorder(javax.swing.BorderFactory.createTitledBorder(null, "ResultPanel", javax.swing.border.TitledBorder.DEFAULT\_JUSTIFICATION, javax.swing.border.TitledBorder.DEFAULT\_POSITION, new java.awt.Font("Calibri", 1, 18))); // NOI18N

 rp\_panel.setPreferredSize(new java.awt.Dimension(300, 340));

 javax.swing.GroupLayout rp\_panelLayout = new javax.swing.GroupLayout(rp\_panel);

 rp\_panel.setLayout(rp\_panelLayout);

 rp\_panelLayout.setHorizontalGroup(

 rp\_panelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

 .addGap(0, 284, Short.MAX\_VALUE)

 );

 rp\_panelLayout.setVerticalGroup(

 rp\_panelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

 .addGap(0, 305, Short.MAX\_VALUE)

 );

 jLabel1.setText("Image\_ID:");

 jLabel3.setText("Description:");

 des.setColumns(20);

 des.setRows(5);

 jScrollPane1.setViewportView(des);

 delete.setText("Delet");

 delete.addActionListener(new java.awt.event.ActionListener() {

 public void actionPerformed(java.awt.event.ActionEvent evt) {

 deleteActionPerformed(evt);

 }

 });

 up.setText("Update");

 up.addActionListener(new java.awt.event.ActionListener() {

 public void actionPerformed(java.awt.event.ActionEvent evt) {

 upActionPerformed(evt);

 }

 });

 close.setText("Close");

 close.addActionListener(new java.awt.event.ActionListener() {

 public void actionPerformed(java.awt.event.ActionEvent evt) {

 closeActionPerformed(evt);

 }

 });

 javax.swing.GroupLayout layout = new javax.swing.GroupLayout(getContentPane());

 getContentPane().setLayout(layout);

 layout.setHorizontalGroup(

 layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

 .addGroup(layout.createSequentialGroup()

 .addComponent(rp\_panel, javax.swing.GroupLayout.PREFERRED\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.PREFERRED\_SIZE)

 .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRAILING)

 .addGroup(javax.swing.GroupLayout.Alignment.LEADING, layout.createSequentialGroup()

 .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)

 .addComponent(jLabel1, javax.swing.GroupLayout.PREFERRED\_SIZE, 67, javax.swing.GroupLayout.PREFERRED\_SIZE)

 .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)

 .addComponent(id, javax.swing.GroupLayout.PREFERRED\_SIZE, 237, javax.swing.GroupLayout.PREFERRED\_SIZE))

 .addGroup(javax.swing.GroupLayout.Alignment.LEADING, layout.createSequentialGroup()

 .addGap(72, 72, 72)

 .addComponent(delete, javax.swing.GroupLayout.PREFERRED\_SIZE, 86, javax.swing.GroupLayout.PREFERRED\_SIZE)

 .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)

 .addComponent(up, javax.swing.GroupLayout.PREFERRED\_SIZE, 91, javax.swing.GroupLayout.PREFERRED\_SIZE)

 .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)

 .addComponent(close, javax.swing.GroupLayout.DEFAULT\_SIZE, 85, Short.MAX\_VALUE))

 .addGroup(javax.swing.GroupLayout.Alignment.LEADING, layout.createSequentialGroup()

 .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)

 .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

 .addComponent(jLabel3, javax.swing.GroupLayout.PREFERRED\_SIZE, 80, javax.swing.GroupLayout.PREFERRED\_SIZE)

 .addComponent(no\_m, javax.swing.GroupLayout.DEFAULT\_SIZE, 346, Short.MAX\_VALUE)))

 .addGroup(layout.createSequentialGroup()

 .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)

 .addComponent(jScrollPane1, javax.swing.GroupLayout.DEFAULT\_SIZE, 344, Short.MAX\_VALUE)))

 .addContainerGap())

 );

 layout.setVerticalGroup(

 layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

 .addGroup(layout.createSequentialGroup()

 .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING, false)

 .addComponent(rp\_panel, javax.swing.GroupLayout.PREFERRED\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.PREFERRED\_SIZE)

 .addGroup(layout.createSequentialGroup()

 .addGap(10, 10, 10)

 .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)

 .addComponent(jLabel1, javax.swing.GroupLayout.PREFERRED\_SIZE, 32, javax.swing.GroupLayout.PREFERRED\_SIZE)

 .addComponent(id, javax.swing.GroupLayout.PREFERRED\_SIZE, 32, javax.swing.GroupLayout.PREFERRED\_SIZE))

 .addGap(18, 18, 18)

 .addComponent(no\_m, javax.swing.GroupLayout.PREFERRED\_SIZE, 47, javax.swing.GroupLayout.PREFERRED\_SIZE)

 .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)

 .addComponent(jLabel3, javax.swing.GroupLayout.PREFERRED\_SIZE, 27, javax.swing.GroupLayout.PREFERRED\_SIZE)

 .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)

 .addComponent(jScrollPane1, javax.swing.GroupLayout.PREFERRED\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.PREFERRED\_SIZE)

 .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED, javax.swing.GroupLayout.DEFAULT\_SIZE, Short.MAX\_VALUE)

 .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)

 .addComponent(delete)

 .addComponent(up)

 .addComponent(close))))

 .addContainerGap(javax.swing.GroupLayout.DEFAULT\_SIZE, Short.MAX\_VALUE))

 );

 pack();

 }// </editor-fold>

 private void closeActionPerformed(java.awt.event.ActionEvent evt) {

 this.dispose();

 }

 private void deleteActionPerformed(java.awt.event.ActionEvent evt) {

 FingerprintImageDAO.deleteImage(id.getText());

 FeatrueDAO.deleteFeatrue(id.getText());

 this.dispose();

 }

 private void upActionPerformed(java.awt.event.ActionEvent evt) {

 FingerprintImageDAO.updataImage(id.getText(), des.getText());

 this.dispose();

 }

 private void formWindowClosing(java.awt.event.WindowEvent evt) {

 this.setDefaultCloseOperation(DISPOSE\_ON\_CLOSE);

 }

 private void formWindowClosed(java.awt.event.WindowEvent evt) {

 }

 // Variables declaration - do not modify

 private javax.swing.JButton close;

 private javax.swing.JButton delete;

 private javax.swing.JTextArea des;

 private javax.swing.JLabel id;

 private javax.swing.JLabel jLabel1;

 private javax.swing.JLabel jLabel3;

 private javax.swing.JScrollPane jScrollPane1;

 private javax.swing.JLabel no\_m;

 private javax.swing.JPanel rp\_panel;

 private javax.swing.JButton up;

 // End of variables declaration

}

## 4.4 DBOriginalImagePanel.java

package demo;

import java.awt.Graphics;

import javax.swing.JPanel;

/\*\*This is the original image drwaing panel in database function

 \*/

public class DBOriginalImagePanel extends JPanel{

 @Override

 public void paintComponent(Graphics g){

 setOpaque(true);

 super.paintComponent(g);

 if(DBOriginalImage.original\_image!=null){

 g.drawImage(DBOriginalImage.original\_image, 20, 20, DBOriginalImage.original\_image.getWidth(), DBOriginalImage.original\_image.getHeight(), this);

 }

 }

}

## 4.5 DBProcessedImagePanel.java

package demo;

import java.awt.Color;

import java.awt.Graphics;

import javax.swing.JPanel;

/\*\*This is the processed image drawing panel in database function

 \*/

public class DBProcessedImagePanel extends JPanel{

 public static boolean showMinutiae = false;

 public static boolean showCore = false;

 @Override

 public void paintComponent(Graphics g){

 setOpaque(true);

 super.paintComponent(g);

 if(DBProcessedImage.thinning\_image!=null){

 g.drawImage(DBProcessedImage.thinning\_image, 20, 20, DBProcessedImage.thinning\_image.getWidth(), DBProcessedImage.thinning\_image.getHeight(), this);

 }else if(DBProcessedImage.binarization\_image!=null){

 g.drawImage(DBProcessedImage.binarization\_image, 20, 20, DBProcessedImage.binarization\_image.getWidth(), DBProcessedImage.binarization\_image.getHeight(), this);

 }else if(DBProcessedImage.enhanced\_image!=null){

 g.drawImage(DBProcessedImage.enhanced\_image, 20, 20, DBProcessedImage.enhanced\_image.getWidth(), DBProcessedImage.enhanced\_image.getHeight(), this);

 }else if(DBProcessedImage.normalized\_image!=null){

 g.drawImage(DBProcessedImage.normalized\_image, 20, 20, DBProcessedImage.normalized\_image.getWidth(), DBProcessedImage.normalized\_image.getHeight(), this);

 }else if(DBProcessedImage.original\_image!=null){

 g.drawImage(DBProcessedImage.original\_image, 20, 20, DBProcessedImage.original\_image.getWidth(), DBProcessedImage.original\_image.getHeight(), this);

 }

 if(showMinutiae){

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

 if(DBProcessedImage.minutiae.get(i).getType()==1){

 g.setColor(Color.RED);

 g.drawOval(DBProcessedImage.minutiae.get(i).getX()+17, DBProcessedImage.minutiae.get(i).getY()+17, 5, 5);

 }else{

 g.setColor(Color.BLUE);

 g.drawOval(DBProcessedImage.minutiae.get(i).getX()+17, DBProcessedImage.minutiae.get(i).getY()+17, 5, 5);

 }

 }

 }

 if(showCore){

 g.setColor(Color.CYAN);

 g.drawOval(DBProcessedImage.core.getX()+17, DBProcessedImage.core.getY()+17, 10, 10);

 }

 }

}

## 4.6 InputImagePanel.java

package demo;

import java.awt.Color;

import java.awt.Graphics;

import javax.swing.JPanel;

/\*\*This is the input image drawing panel

 \*/

public class InputImagePanel extends JPanel{

 public static boolean showMinutiae = false;

 public static boolean showCore = false;

 public static boolean showSame = false;

 @Override

 public void paintComponent(Graphics g){

 setOpaque(true);

 super.paintComponent(g);

 if(InputImage.thinning\_image!=null){

 g.drawImage(InputImage.thinning\_image, 20, 20, InputImage.thinning\_image.getWidth(), InputImage.thinning\_image.getHeight(), this);

 }else if(InputImage.binarization\_image!=null){

 g.drawImage(InputImage.binarization\_image, 20, 20, InputImage.binarization\_image.getWidth(), InputImage.binarization\_image.getHeight(), this);

 }else if(InputImage.enhanced\_image!=null){

 g.drawImage(InputImage.enhanced\_image, 20, 20, InputImage.enhanced\_image.getWidth(), InputImage.enhanced\_image.getHeight(), this);

 }else if(InputImage.normalized\_image!=null){

 g.drawImage(InputImage.normalized\_image, 20, 20, InputImage.normalized\_image.getWidth(), InputImage.normalized\_image.getHeight(), this);

 }else if(InputImage.original\_image!=null){

 g.drawImage(InputImage.original\_image, 20, 20, InputImage.original\_image.getWidth(), InputImage.original\_image.getHeight(), this);

 }

 if(showMinutiae){

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

 if(InputImage.minutiae.get(i).getType()==1){

 g.setColor(Color.RED);

 g.drawOval(InputImage.minutiae.get(i).getX()+17, InputImage.minutiae.get(i).getY()+17, 5, 5);

 }else{

 g.setColor(Color.BLUE);

 g.drawOval(InputImage.minutiae.get(i).getX()+17, InputImage.minutiae.get(i).getY()+17, 5, 5);

 }

 }

 }

 if(showCore){

 g.setColor(Color.CYAN);

 g.drawOval(InputImage.core.getX()+17, InputImage.core.getY()+17, 10, 10);

 }

 if(showSame){

 int r\_1 = 34;

 int r\_2 = 187;

 int r\_3 = 49;

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

 r\_1 = r\_1+i\*20;

 if(r\_1>255){

 r\_2 = r\_2 + i\*20;

 r\_1 = 0;

 }

 if(r\_2>255){

 r\_3 = r\_3+i\*20;

 r\_2 = 0;

 }

 if(r\_3>255){

 r\_3 = 0;

 }

 g.setColor(new Color(r\_1,r\_2,r\_3));

 g.drawOval(InputImage.same.get(i).getX()+17, InputImage.same.get(i).getY()+17, 5, 5);

 }

 }

 }

}

## 4.7 TemplateImagePanel.java

package demo;

import java.awt.Color;

import java.awt.Graphics;

import javax.swing.JPanel;

/\*\*This is the template image drawing panel

 \*/

public class TemplateImagePanel extends JPanel{

 public static boolean showMinutiae = false;

 public static boolean showCore = false;

 public static boolean showSame = false;

 @Override

 public void paintComponent(Graphics g){

 setOpaque(true);

 super.paintComponent(g);

 if(TemplateImage.thinning\_image!=null){

 g.drawImage(TemplateImage.thinning\_image, 20, 20, TemplateImage.thinning\_image.getWidth(), TemplateImage.thinning\_image.getHeight(), this);

 }else if(TemplateImage.binarization\_image!=null){

 g.drawImage(TemplateImage.binarization\_image, 20, 20, TemplateImage.binarization\_image.getWidth(), TemplateImage.binarization\_image.getHeight(), this);

 }else if(TemplateImage.enhanced\_image!=null){

 g.drawImage(TemplateImage.enhanced\_image, 20, 20, TemplateImage.enhanced\_image.getWidth(), TemplateImage.enhanced\_image.getHeight(), this);

 }else if(TemplateImage.normalized\_image!=null){

 g.drawImage(TemplateImage.normalized\_image, 20, 20, TemplateImage.normalized\_image.getWidth(), TemplateImage.normalized\_image.getHeight(), this);

 }else if(TemplateImage.original\_image!=null){

 g.drawImage(TemplateImage.original\_image, 20, 20, TemplateImage.original\_image.getWidth(), TemplateImage.original\_image.getHeight(), this);

 }

 if(showMinutiae){

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

 if(TemplateImage.minutiae.get(i).getType()==1){

 g.setColor(Color.RED);

 g.drawOval(TemplateImage.minutiae.get(i).getX()+17, TemplateImage.minutiae.get(i).getY()+17, 5, 5);

 }else{

 g.setColor(Color.BLUE);

 g.drawOval(TemplateImage.minutiae.get(i).getX()+17, TemplateImage.minutiae.get(i).getY()+17, 5, 5);

 }

 }

 }

 if(showCore){

 g.setColor(Color.CYAN);

 g.drawOval(TemplateImage.core.getX()+17, TemplateImage.core.getY()+17, 10, 10);

 }

 if(showSame){

 int r\_1 = 34;

 int r\_2 = 187;

 int r\_3 = 49;

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

 r\_1 = r\_1+i\*20;

 if(r\_1>255){

 r\_2 = r\_2 + i\*20;

 r\_1 = 0;

 }

 if(r\_2>255){

 r\_3 = r\_3 + i\*20;

 r\_2 = 0;

 }

 if(r\_3>255){

 r\_3 = 0;

 }

 g.setColor(new Color(r\_1,r\_2,r\_3));

 g.drawOval(TemplateImage.same.get(i).getX()+17, TemplateImage.same.get(i).getY()+17, 5, 5);

 }

 }

 }

}

## 4.8 ResultImagePanel.java

package demo;

import java.awt.Graphics;

import javax.swing.JPanel;

/\*\*This is for drawing the result image

 \*/

public class ResultImagePanel extends JPanel{

 @Override

 public void paintComponent(Graphics g){

 setOpaque(true);

 super.paintComponent(g);

 if(ResultWindow.result!=null){

 g.drawImage(ResultWindow.result, 20, 20, ResultWindow.result.getWidth(), ResultWindow.result.getHeight(), this);

 }

 }

}