Mandarin Writing Trainer with Java

Although the author and publisher have made every effort to ensure that the information in this writing was correct at press time, the author and publisher do not assume and hereby disclaim any liability to any party for any loss, damage, or disruption caused by errors or omissions, whether such errors or omissions result from negligence, accident, or any other cause.

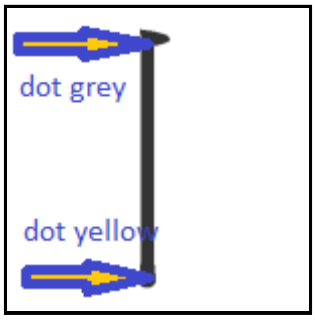
paid link

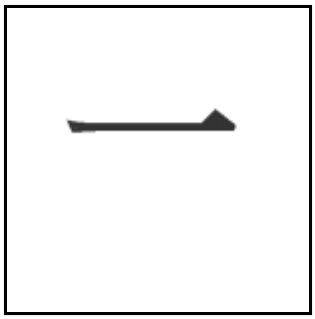
Mandarin is one of the most used languages. In order to memorize Chinese characters, one of the best approaches is to practice handwriting. In this short writing, we create a program that allows us to do it using our PC and mouse.

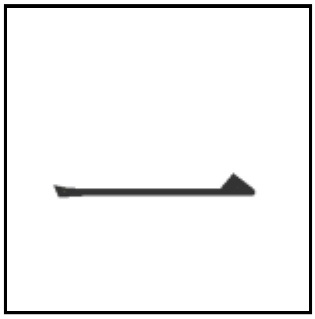
To start with, each of the characters are divided into smaller parts. Then we determine the start and finish points. In this case, the start point is a gray colored dot (RGB = 127,127,127; int = -8421505). The finish point is a yellow color dot (RGB = 255,242,0; int = -3584). Mark each of the part with the start and finish point.

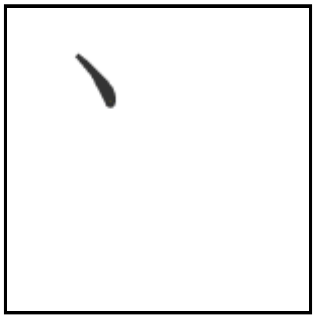
Main image:

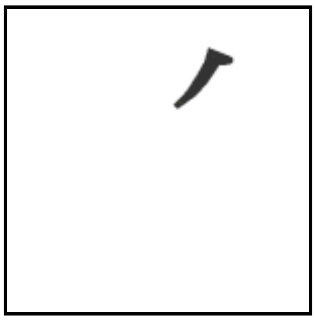
Sub images:











Here is the full code:

```
import javax.swing.*;
import javax.swing.event.*;
import java.awt.*;
import java.awt.event.*;
import java.awt.image.BufferedImage;
import java.io.File;
```

```
import java.io.IOException;
import java.util.ArrayList;
import javax.imageio.ImageIO;

public class MandarinTrainer extends JFrame{
    private JPanel pMain = null;
    private BufferedImage bufImg[] = null;
    private ArrayList<ArrayList> pixXY = null;
    private ArrayList<Point> startXY = null;
    private ArrayList<Point> endXY = null;
    private int curIdx;
    private boolean trueArea = false;

    public MandarinTrainer(){
        super('Mandarin Trainer');
        setSize(450,450);

        pMain = new DrawPanel();
        pMain.setPreferredSize(new Dimension(450,450));
        pMain.setBackground(Color.WHITE);
        pMain.addMouseListener(new MouseHandler());
        pMain.addMouseMotionListener(new MouseHandler());
        add(pMain);

        bufImg = new BufferedImage[5];
        try {
            bufImg[0] = ImageIO.read(new File('incomplete1.png'));
            bufImg[1] = ImageIO.read(new File('incomplete2.png'));
            bufImg[2] = ImageIO.read(new File('incomplete3.png'));
            bufImg[3] = ImageIO.read(new File('incomplete4.png'));
            bufImg[4] = ImageIO.read(new File('incomplete5.png'));
        } catch (IOException e) {
        }

        pixXY = new ArrayList<ArrayList>();
        startXY = new ArrayList<>();
        endXY = new ArrayList<>();

        for(int i=0;i<bufImg.length;i++){
            getPixel(i);
        }

        curIdx = 0;

        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        show();
    }

    class DrawPanel extends JPanel{
        public void paintComponent(Graphics g){
            super.paintComponent(g);
            Graphics2D g2d = (Graphics2D)g;

            Composite c = AlphaComposite.getInstance(AlphaComposite.SRC_OVER,0.1f);
            g2d.setComposite(c);
            for(int i=0;i<pixXY.size();i++){
                for(int j=0;j<pixXY.get(i).size();j++){
                    Point p = (Point) pixXY.get(i).get(j);
                    g2d.drawOval(p.x, p.y, 1, 1);
                }
            }

            if(startXY != null){
                g2d.setColor(Color.BLACK);
                for(int i=0;i<startXY.size();i++){
                    Point p = (Point) startXY.get(i);
                    g2d.drawOval(p.x-5, p.y-5, 10, 10);
                    p = (Point) endXY.get(i);
                    g2d.drawOval(p.x-5, p.y-5, 10, 10);
                }
            }
```

```
    }
}

private void getPixel(int idx){
    pixXY.add(new ArrayList<Point>());

    for(int i = 0;i < bufImg[idx].getHeight();i++){
        for(int j = 0;j < bufImg[idx].getWidth();j++){
            int curpixel = bufImg[idx].getRGB(j,i);
            if(curpixel != -1){
                if(curpixel == -8421505){
                    startXY.add(new Point(j,i));
                }else if(curpixel == -3584){
                    endXY.add(new Point(j,i));
                }else{
                    pixXY.get(idx).add(new Point(j,i));
                }
            }
        }
    }

    pMain.repaint();
}

class MouseHandler implements MouseMotionListener, MouseListener{
    public void mouseMoved(MouseEvent e){
    }
    public void mouseEntered(MouseEvent e){
    }
    public void mouseExited(MouseEvent e){
    }
    public void mousePressed(MouseEvent e){
        trueArea = false;
        Point p = (Point) startXY.get(curIdx);

        if((e.getPoint().x >= (p.x - 5)) %26;%26;
           (e.getPoint().x <= (p.x + 5)) %26;%26;
           (e.getPoint().y >= (p.y - 5)) %26;%26;
           (e.getPoint().y <= (p.y + 5))
           ){
            trueArea = true;
        }
    }
    public void mouseDragged(MouseEvent e){
        if(trueArea == true){
            if(pixXY.get(curIdx).contains(e.getPoint())){
            }else{
                trueArea = false;
                return;
            }
        }
    }
    public void mouseReleased(MouseEvent e){
        Point p = (Point) endXY.get(curIdx);

        if(trueArea == true){
            if((e.getPoint().x >= (p.x - 5)) %26;%26;
               (e.getPoint().x <= (p.x + 5)) %26;%26;
               (e.getPoint().y >= (p.y - 5)) %26;%26;
               (e.getPoint().y <= (p.y + 5))){
                System.out.println('Right');
                if(curIdx == bufImg.length-1){
                    System.out.println('Finish!');
                }else{
                    curIdx += 1;
                }
            }else{
                trueArea = false;
                return;
            }
        }
```

```
        }
        public void mouseClicked(MouseEvent e){
        }
    }

    public static void main(String[] args){
        SwingUtilities.invokeLater(new Runnable(){
            @Override
            public void run() {
                new MandarinTrainer();
            }
        });
    }
}
```

Explanation

bufImg : for the smaller image parts.
pixXY : the pixels of the images.
startXY : the starting point of each image (grey colored dot)
endXY : the finish/end point of each image (yellow colored dot)
curIdx : index of the current image part
trueArea : is the user clicking/dragging on the right area?

```
bufImg = new BufferedImage[5];
        try {
            bufImg[0] = ImageIO.read(new File('incomplete1.png'));
            bufImg[1] = ImageIO.read(new File('incomplete2.png'));
            bufImg[2] = ImageIO.read(new File('incomplete3.png'));
            bufImg[3] = ImageIO.read(new File('incomplete4.png'));
            bufImg[4] = ImageIO.read(new File('incomplete5.png'));
        } catch (IOException e) {
        }
```

In the above lines of code, image parts are stored in the variable *bufImg*. Keep in mind that different characters may have a different number of image parts.

```
for(int i=0;i<bufImg.length;i++){
    getPixel(i);
}
```

Get the pixel values of each of the image parts.

```
Composite c = AlphaComposite.getInstance(AlphaComposite.SRC_OVER,0.1f);
g2d.setComposite(c);
for(int i=0;i<pixXY.size();i++){
    for(int j=0;j<pixXY.get(i).size();j++)
{
        Point p = (Point) pixXY.get(i).get(j);
        g2d.drawOval(p.x, p.y, 1, 1);
    }
}
```

Draw the image parts.

```
if(startXY != null){
    g2d.setColor(Color.BLACK);
    for(int i=0;i<startXY.size();i++){
        Point p = (Point) startXY.get(i);
        g2d.drawOval(p.x-5, p.y-5, 10, 10);
        p = (Point) endXY.get(i);
        g2d.drawOval(p.x-5, p.y-5, 10, 10);
    }
}
```

Determine the start and finish areas.

```
for(int i = 0;i < bufImg[idx].getHeight();i++){
    for(int j = 0;j < bufImg[idx].getWidth();j++){
        int curpixel = bufImg[idx].getRGB(j,i);
        if(curpixel != -1){
            if(curpixel == -8421505){
                startXY.add(new Point(j,i));
            }else if(curpixel == -3584){
                endXY.add(new Point(j,i));
            }else{
                pixXY.get(idx).add(new Point(j,i));
            }
        }
    }
}
```

For each of the image parts, get the RGB values. Also get the start and finish points.

```
public void mousePressed(MouseEvent e){
    trueArea = false;
    Point p = (Point) startXY.get(curIdx);

    if((e.getPoint().x >= (p.x - 5)) %26;%26;
       (e.getPoint().x <= (p.x + 5)) %26;%26;
       (e.getPoint().y >= (p.y - 5)) %26;%26;
       (e.getPoint().y <= (p.y + 5))
){
           trueArea = true;
    }
}
```

Determine whether the mouse is clicked in the right place (inside the start area).

```
public void mouseDragged(MouseEvent e){
    if(trueArea == true){
        if(pixXY.get(curIdx).contains(e.getPoint())){
        }else{
            trueArea = false;
            return;
        }
    }
}
```

Determine whether the mouse is dragged inside the correct area.

```
public void mouseReleased(MouseEvent e){
    Point p = (Point)  endXY.get(curIdx);

    if(trueArea == true){
        if((e.getPoint().x >= (p.x - 5)) %26;%26;
           (e.getPoint().x <= (p.x + 5)) %26;%26;
           (e.getPoint().y >= (p.y - 5)) %26;%26;
           (e.getPoint().y <= (p.y + 5))){
               System.out.println('Right');
               if(curIdx == bufImg.length-1){
                    System.out.println('Finish!');
               }else{
                   curIdx += 1;
               }
        }else{
            trueArea = false;
            return;
        }
    }
}
```

If the mouse is released when it is inside the finish area, the word 'Right' is displayed. If it is the last image part, the word 'Finish' is displayed.
