



面向对象程序设计课程大作业
《截图小工具》
结题报告

班级： 地信 1801

姓名： 刘乐

学号： 8211180103

指导： 王盼成

2020 年 7 月 3 日

刘乐

目录

一、	概述	1
1.	功能	1
2.	范围界定	1
二、	系统结构及开发环境	1
1.	结构	1
2.	开发环境	1
3.	外部依赖	2
4.	数据记录	2
三、	可行性分析及测试	2
1.	截图	2
2.	QRCode 二维码识别	2
3.	OCR 图片文字识别	3
4.	数据库连接	4
四、	功能和界面介绍	5
1.	主界面:	5
2.	系统托盘	5
3.	截图界面	5
4.	识别结果界面	6
5.	历史记录界面	6
五、	开发实现	7
1.	图形实体类: (Package ml.liule.screenShotTool.graelement)	7
1.1.	类图	7
1.2.	基类 (GraElement)	7
1.3.	子类举例	8
2.	界面类: (Package ml.liule.screenShotTool.ui)	10
2.1.	MainWindow (extends JWindow)	10
2.2.	RecordsWindow(extends JFrame)	11
2.3.	ScrShotWindow(extends JWindow)	13
2.4.	ControlWindow(extends JFrame)	13
2.5.	TextWindow(extends JFrame)	15
2.6.	ChosenRectangle(extends Rectangle)	15
六、	遇到的问题解决及优化	16
1.	截图分辨率降低	16
2.	添加按钮图标后编译报错	16
3.	批量添加 RadioButton 事件	16
4.	多线程优化	17
5.	使用内部类 MyDate(extends Date)	17
6.	关于太极图案主界面	17

一、概述

1. 功能

为方便日常生活、办公等需求，拟开发一款截图小工具。该工具具备以下功能：

- 1) 能够对屏幕指定区域进行截取，并保存为图片文件或放入剪贴板中
- 2) 能够在截取时，对图像进行简单编辑。包括调整截取范围、自由画笔、绘制折线、添加文字。
- 3) 能够对截取的图片进行二维码扫描
- 4) 能够对截取的图片进行 OCR 识别
- 5) 截图后能够将截图时间、图片、识别结果进行记录保存
- 6) 能够根据截图时间对保存的记录进行查询、复原和删除

2. 范围界定

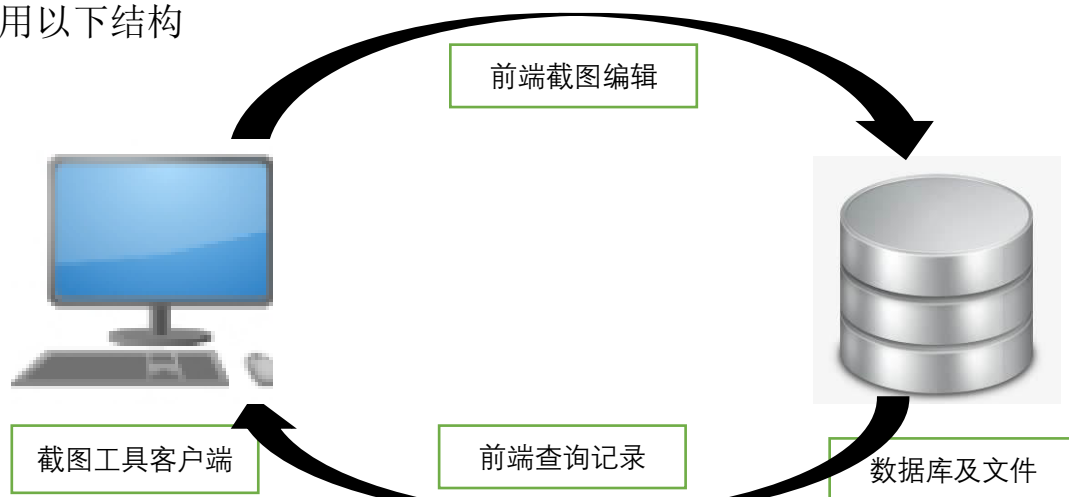
为了开发的简单，对功能范围进行界定

- 1) 二维码扫描功能只识别图片中的单个 QR 二维码
- 2) 一次截图只记录二维码扫描和 OCR 识别中的一个结果
- 3) 截图图片以及 json 文本以文件形式存放，不存放数据库中
- 4) 添加文字仅限单行文字，且不进行字体设置
- 5) 不加入用户自定义设置
- 6) 不对记录的数据进行加密和备份
- 7) 不设置快捷键

二、系统结构及开发环境

1. 结构

本工具采用以下结构



2. 开发环境

IDE	NetBeans 11.3
Database Server	SQLite 3
Language	Java 1.8

客户端采用 Java + Swing 开发，采用 maven 进行项目管理。由于工具的作用并不是管理数据，并且为本地轻量应用，故数据库采用 SQLite 3。

图片文件采用 jpg 文件格式存储，对图片的操作信息使用 json 格式文件存储。

3. 外部依赖

项目采用 Maven 管理，可以方便地添加外部依赖。当前项目添加的依赖如下：

groupId	artifactId	版本	作用
com.baidu.aip	java-sdk	4.12.0	进行 OCR
org.xerial	sqlite-jdbc	3.31.1	连接 SQLite3 数据库
org.swinglabs.swingx	swingx-core	1.6.5-1	日期选择器控件
com.google.zxing	core, javase	3.4.0	进行 QR 二维码识别

4. 数据记录

由于记录的时间戳精确到毫秒，故假设记录不会重复。由于 id 字段并不打算显示在 jTable 上，所以 id 字段实际可省，将时间戳作为唯一标识。

● SQLite 3 记录

字段名	字段类型	含义
id	INT	唯一自增 ID
time	BIGINT	截图时的时间戳
result	TEXT	OCR 或二维码结果

● 图片及 json 文本存储

路径	SQLite 3 同路径下的 file 文件夹
文件名	时间 yy 年 MM 月 dd 日 HH 时 mm 分 ss 秒+后缀

三、可行性分析及测试

1. 截图

java.awt.Robot 类的 createScreenCapture() 方法可以获得屏幕截图。

测试代码：

```

1. public class ShotTest {
2.     public static void main(String[] args) throws AWTException, IOException {
3.         Dimension screenSize = Toolkit.getDefaultToolkit().getScreenSize();
4.         Robot robot = new Robot();
5.         Rectangle screenRec=new Rectangle(0, 0, screenSize.width, screenSize.height);
6.         BufferedImage screenCapture = robot.createScreenCapture(screenRec);
7.         ImageIO.write(screenCapture, "png", new File("D:\\shotTest.png"));
8.     }
9. }

```

经测试，该函数可将当前屏幕截取下来并保存，截图的速度及清晰度均能满足要求。

2. QRCode 二维码识别

使用 com.google.zxing 外部依赖可以实现二维码的识别。

● Maven 配置

```

<dependency>
  <groupId>com.google.zxing</groupId>
  <artifactId>core</artifactId>
  <version>3.4.0</version>

```

```

</dependency>
<dependency>
    <groupId>com.google.zxing</groupId>
    <artifactId>javase</artifactId>
    <version>3.4.0</version>
</dependency>

```

● 测试代码

```

1. public class QRCodeTools {
2.
3.     public static void main(String[] args) throws IOException, NotFoundException, ChecksumExcept
        ion, FormatException {
4.         BufferedImage image = ImageIO.read(new File("D:\\OneDriveBackup\\QR.png"));
5.         LuminanceSource source = new BufferedImageLuminanceSource(image);
6.         Binarizer binarizer = new HybridBinarizer(source);
7.         BinaryBitmap binaryBitmap = new BinaryBitmap(binarizer);
8.         Map<DecodeHintType, Object> hints = new HashMap<>();
9.         hints.put(DecodeHintType.CHARACTER_SET, "utf-8");
10.        hints.put(DecodeHintType.TRY_HARDER, Boolean.TRUE);
11.        Result result = new QRCodeReader().decode(binaryBitmap, hints); //解码
12.        System.out.println(result);
13.    }
14. }

```

经测试，上述代码可以对二维码进行解码，且成功率较高。若未识别二维码，会抛出 NotFoundException。

3. OCR 图片文字识别

使用百度开发者平台提供的通用文字识别服务。

● Maven 依赖配置

```

<dependency>
    <groupId>com.baidu.aip</groupId>
    <artifactId>java-sdk</artifactId>
    <version>4.12.0</version>
</dependency>

```

● 测试代码（该 API 秘钥为个人申请，识别次数有限，请勿滥用）

```

1. public class OCRTest {
2.     public static void main(String[] args) {
3.         HashMap<String, String> options = new HashMap<String, String>();
4.         options.put("language_type", "CHN_ENG");
5.         String appId = "20222761";
6.         String apiKey = "uLRTAip8RqvGNTTykovfr80g";
7.         String secretKey = "Qm71QdQMLkKjzcafZr2Gy3KxF1CVUUFN";
8.         AipOcr client = new AipOcr(appId, apiKey, secretKey);
9.         String image = "D:\\OneDriveBackup\\OCRTest.png";
10.        JSONObject res = client.basicGeneral(image, options);
11.        System.out.println(res.toString(2));
12.    }
13. }

```

识别准确率高，且使用门槛低。该服务每天有 50000 次的免费额度，能满足该工具的使用。

4. 数据库连接

需要对截图记录进行存取，采用 SQLite 3 就足够满足要求。

● Maven 配置

```
<dependency>
  <groupId>org.xerial</groupId>
  <artifactId>sqlite-jdbc</artifactId>
  <version>3.31.1</version>
</dependency>
```

● 测试代码

```
1. public class SQLiteTest {
2.
3.     public static void main(String[] args) throws SQLException, InterruptedException {
4.         String createTableSql = "CREATE TABLE records(id INTEGER PRIMARY KEY autoincrement,time
      INTEGER,result nvarchar (1024))";
5.         String insertSql = "insert into records(time,result) values(%, '%s')";
6.         SimpleDateFormat dateFormat=new SimpleDateFormat("MM月dd日 :mm:ss");
7.         try (Connection connection = DriverManager.getConnection("jdbc:sqlite:D:\\Test.db"))
8.         {
9.             Statement statement = (Statement) connection.createStatement();
10.            statement.execute(createTableSql);
11.            statement.execute(String.format(insertSql, new Date().getTime(), "第一条记录"));
12.            Thread.sleep(3000);
13.            statement.execute(String.format(insertSql, String.valueOf(new Date().getTime()), "第
      二条记录"));
14.            try (ResultSet rSet = statement.executeQuery("select * from records")) {
15.                while (rSet.next()) {
16.                    System.out.println("id: " + rSet.getString("id"));
17.                    System.out.println("时间:
      " + dateFormat.format(new Date(rSet.getLong("time"))));
18.                    System.out.println("结果: " + rSet.getString("result"));
19.                }
20.                rSet.close();//关闭数据集
21.                connection.close();//关闭数据库连接
22.            }
23.        }
24.    }
25. }
```

● 输出结果

```
id: 1
时间: 06月06日 21:37:35
结果: 第一条记录
id: 2
时间: 06月06日 21:37:38
结果: 第二条记录
```

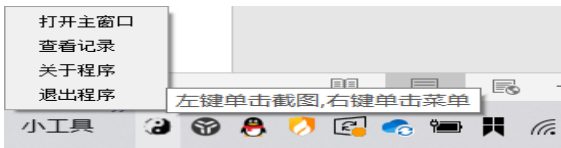
四、功能和界面介绍

1. 主界面：



- 该窗口为半透明圆形窗口
- 点击白色区域，出现截图界面
- 点击黑色区域，出现历史记录界面
- 拖拽移动窗口；右键隐藏窗口

2. 系统托盘







- 左键单击开始截图，右键单击出现弹出菜单
- 菜单各个选项对应其含义上功能










3. 截图界面



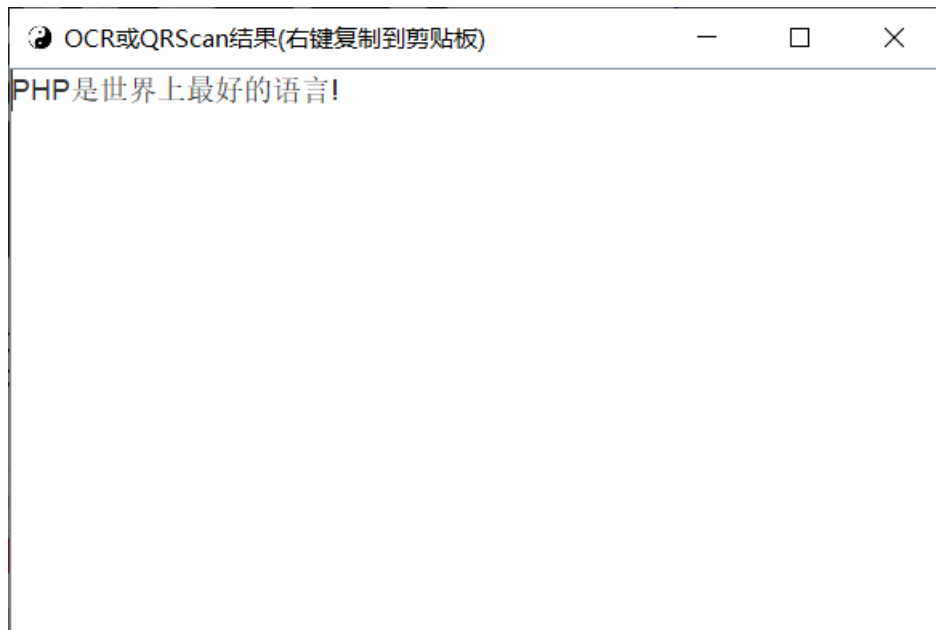
该窗口为全屏，在窗口内进行框选，框选区域以外变暗。框选结束后出现下方按钮。各功能按钮如下：



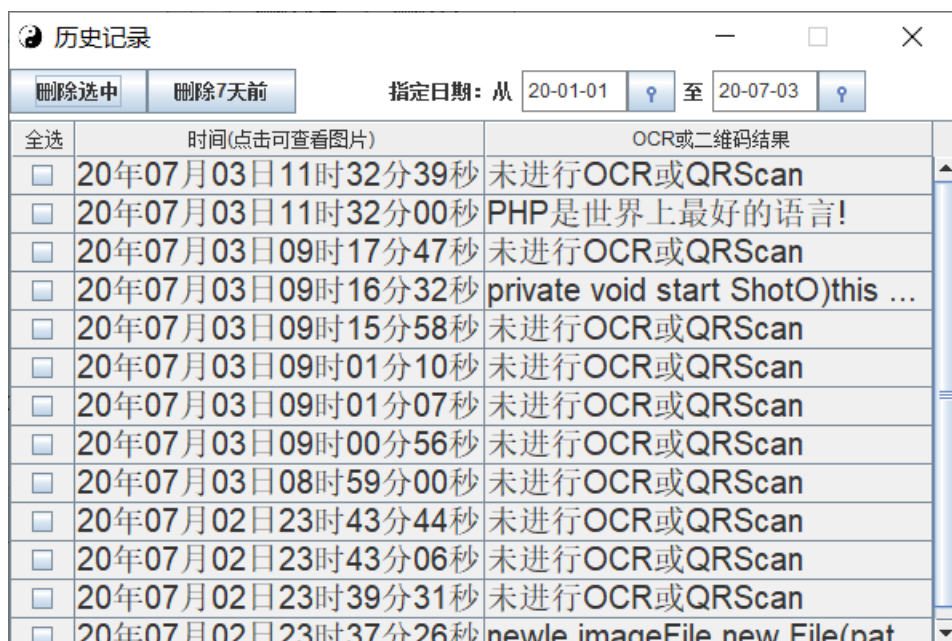
- 点击 ，弹出系统保存文件 FileChooser
- 点击 ，复制所选区域截图到剪贴板
- 点击 ，出现 OCR 识别界面
- 点击 ，出现二维码扫描结果界面

- 点击  或 ，进行绘画标记
- 点击 ，退出截屏界面
- 点击 ，撤销上次操作
- 点击  调整截图区域
- 点击  切换画笔颜色
- 点击 ，添加文本框内的文字
- 点击 ，绘制矩形
- 点击 ，重做撤销的操作
- 滚动鼠标滚轮调整画笔粗细以及字体大小

4. 识别结果界面



5. 历史记录界面



- 双击“全选”表头，可以全选当前记录；同时表头文字变为“反选”；双击“反选”表头，可以选择当前未选择的所有记录。

- 双击记录的时间，显示截屏界面，还原当时截图场景
- 双击 OCR 或二维码结果，弹出结果文本界面
- 点击最近三天，过滤显示近三天的记录
- 点击 指定日期：从 至 ，弹出日期选择器。选择后筛选时间范围内的记录。
- 点击“删除选中”，删除当前选择的记录；点击“删除 7 天前”，则删除七天前的记录。

五、 开发实现

1. 图形实体类：(Package ml.liule.screenShotTool.graelement)

该包内包含四个类：GraElement, MyPolyline, MyRectangle, MyStringGra。其中 GraElement 为虚类，其他类都从其继承得到。继承得到的类实现了基类的 toJson(), fromJson(), draw() 方法，以及 weight、color 字段。

1.1. 类图



1.2. 基类 (GraElement)

```

1. public abstract class GraElement {
2.
3.     public static final int LINE = 0;
4.     public static final int STR = 1;
5.     public static final int RECT = 2;
6.

```

```

7.     protected float weight = 4;
8.     protected Color color = Color.BLACK;
9.     protected int type = -1;
10.
11.    public void setWeight(float weight) {
12.        this.weight = weight;
13.    }
14.
15.    public void setColor(Color color) {
16.        this.color = color;
17.    }
18.
19.    public int getType() {
20.        return type;
21.    }
22.
23.    public abstract JSONObject toJson();
24.
25.    protected abstract void fromJson(JSONObject json);
26.
27.    public abstract void draw(Graphics2D g2d);
28.
29.    protected void setPen(Graphics2D g2d) {
30.        g2d.setStroke(new BasicStroke(weight, BasicStroke.CAP_ROUND, BasicStroke.JOIN_ROUND));
31.        g2d.setColor(color);
32.    }
33.
34.    protected String point2WKT(Point p) {
35.        return "(" + p.x + " " + p.y + ")";
36.    }
37.
38.    protected Point WKT2Point(String WKT) {
39.        Point p = new Point();
40.        String rawWKT = WKT.substring(1, WKT.length() - 1);
41.        String[] xyStrings = rawWKT.split(" ");
42.        p.x = Integer.parseInt(xyStrings[0]);
43.        p.y = Integer.parseInt(xyStrings[1]);
44.        return p;
45.    }
46. }

```

1.3. 子类举例

● MyRectangle 的 toJson() 方法

```

1. @Override
2. public JSONObject toJson() {
3.     JSONObject json = new JSONObject();
4.     json.put("Type", type);

```

```

5.     json.put("Color", color.getRGB());
6.     json.put("Weight", weight);
7.     JSONObject content = new JSONObject();
8.     content.put("StartPoint", point2WKT(startP));
9.     content.put("EndPoint", point2WKT(endP));
10.    json.put("Content", content);
11.    return json;
12. }

```

● MyPolyline 的 draw() 方法

```

1. @Override
2. public void draw(Graphics2D g2d) {
3.     this.setPen(g2d);
4.     g2d.setRenderingHint(RenderingHints.KEY_ANTIALIASING, RenderingHints.VALUE_ANTIALIAS_ON);//
    抗锯齿
5.     int ptcount = points.size();
6.     int[] xarr = new int[ptcount];
7.     int[] yarr = new int[ptcount];
8.
9.     for (int i = 0; i < ptcount; i++) {
10.        Point pt = points.get(i);
11.        xarr[i] = (int) pt.getX();
12.        yarr[i] = (int) pt.getY();
13.    }
14.    g2d.drawPolyline(xarr, yarr, ptcount);
15.    drawRubberBand(g2d);
16. }

```

● MyPolyline 的 polyline2WKT()、WKT2polyline(String WKT) 方法

```

1. private String polyline2WKT() {
2.     int pCount = points.size();
3.     StringJoiner sj = new StringJoiner(",", "(", ")");
4.     for (int i = 0; i < pCount; i++) {
5.         sj.add(point2WKT(points.get(i)));
6.     }
7.     return sj.toString();
8. }
9.
10. private void WKT2polyline(String WKT) {
11.     String subWKT = WKT.substring(1, WKT.length() - 1);
12.     String[] pointWKTs = subWKT.split(",");
13.     for (String wkt : pointWKTs) {
14.         points.add(WKT2Point(wkt));
15.     }
16. }

```

2. 界面类: (Package ml.liule.screenShotTool.ui)

2.1.MainWindow (extends JWindow)

该类继承自 JWindow, 是程序的主入口。设置窗口的 shape 为 Ellipse2D.Float 对象使得其为圆形; 设置 Opacity 使其半透明。

2.1.1. 主要字段介绍

- `public static Connection dbConnection`

用于保留与数据库的连接, 一次连接, 程序运行期间都可访问数据库。声明静态变量, 方便其他类访问。

- `public static String recordsPath`

用于设定截图记录的文本及图片, 以及 SQLite3 的 db 文件的父路径。

- `private Area yinArea`

太极黑色(阴)所在的区域。将由 `initTaiChiArea()` 进行初始化。该变量以及 `yangArea` 用于绘制太极并判断鼠标所在的区域。

2.1.2. 主要方法介绍

- `initComponents()`

初始化主窗口。包括设置窗口大小、形状、透明度等属性, 并添加事件处理函数。

核心代码:

```
1. setLocationRelativeTo(null); //设置窗口居中
2. setOpacity((float) (0.80)); //设置窗口透明
3. setShape(new Ellipse2D.Float(0, 0, this.getWidth(), this.getWidth())); //设置圆形窗口
```

- `initTray()`

初始化托盘。包括设置图标、添加 PopupMenu 及其事件处理。

核心代码:

```
1. if (!SystemTray.isSupported()) {
2.     return;
3. }
4. SystemTray tray = SystemTray.getSystemTray();
5. PopupMenu pop = new PopupMenu(); // 构造一个右键弹出式菜单
6. TrayIcon trayIcon = new TrayIcon(icon, "左键单击截图,右键单击菜单", pop); //构建一个托盘图标
7. trayIcon.setImageAutoSize(true);
8. tray.add(trayIcon); //添加到系统托盘
```

- `initTaiChiArea()`

通过若干 Shape 对象的交并补运算获得太极的阴阳区域, 方便判断鼠标点击的区域是阴还是阳。不同的区域执行不同的事件处理。

● initFile()

初始化存储记录的路径，调用 connectToDB() 函数初始化数据库连接。如果路径不存在，则新建路径并弹出首次使用的 about 文字窗口简单介绍软件的操作使用。

● connectToDB()

连接到数据库，并完成初始建表工作。将数据库的连接保存在 Connection 对象 dbConnection，后续需要读写数据库时不用再进行连接。数据库连接在关闭程序前释放。

核心代码：

```
1. private void connectToDB() throws SQLException {
2.     String createTableSql = "CREATE TABLE records("
3.         + "id INTEGER PRIMARY KEY autoincrement,"
4.         + "time INTEGER ,"
5.         + "result TEXT"
6.         + ")";
7.     dbConnection = DriverManager.getConnection("jdbc:sqlite:" + recordsPath + "records.db");
8.     Statement statement = (Statement) dbConnection.createStatement();
9.     try {
10.        statement.execute(createTableSql);
11.    } catch (SQLException e) {
12.    }
13. }
```

2.2. RecordsWindow(extends JFrame)

该类继承自 JFrame，用于显示截图历史记录。其中有一个内部类 MyDate 继承自 Date 并覆写了其 toString 方法。其中的日期选择器控件来源于外部依赖 org.swinglabs.swingx。

2.2.1. 主要方法介绍

● initTable()

初始化 tableModel 并添加事件处理函数。该事件监听可以精确到点击的每一个单元格。

```
1. private void initTable() {
2.     tableModel = (DefaultTableModel) jTable.getModel();
3.     jTable.addMouseListener(new MouseAdapter() {
4.         public void mouseClicked(MouseEvent e) {
5.             if (e.getClickCount() == 2) {
6.                 int row = jTable.rowAtPoint(e.getPoint()); //行位置
7.                 int col = jTable.columnAtPoint(e.getPoint()); //列位置
8.                 onGridDoubleClicked(row, col);
9.             } else {
10.                return;
11.            }
12.        }
13.    });
14.    JTableHeader tableHeader = jTable.getTableHeader();
15.    tableHeader.addMouseListener(new MouseAdapter() {
16.        @Override
17.        public void mouseClicked(MouseEvent e) {
```

```

18.         if (tableHeader.columnAtPoint(e.getPoint()) == 0) {
19.             onSelectSwitchClicked(tableHeader.getColumnModel().getColumn(0));
20.         }
21.     }
22. });
23. }

```

● deleteByDate(MyDate date)

通过日期作为查询条件值删除记录并删除对应的记录文件。整个类中关于删除记录的操作都基于这个函数实现。包括按时间段删除记录。如果是按时间段删除，SQL 使用 WHERE 语句 BETWEEN AND 判断，效率更高，但是不便于同步删除记录的文件。

```

1. private void deleteByDate(MyDate date) throws SQLException {
2.     String path = MainWindow.recordsPath + "\\file\\" + date;
3.     Statement statement = MainWindow.dbConnection.createStatement();
4.     String sqlString = "DELETE FROM records WHERE time= %s";
5.     statement.execute(String.format(sqlString, date.getTime()));
6.     statement.close();
7.     File pic = new File(path + ".jpg");
8.     File json = new File(path + ".json");
9.     if (pic.exists()) {
10.        pic.delete();
11.    }
12.    if (json.exists()) {
13.        json.delete();
14.    }
15. }

```

● onSelectSwitchClicked(TableColumn selectSwitch)

点击表头“全选”，全选当前所有记录并设置为“反选”；点击“反选”时将选择当前所有未选择，并且表头又变回“全选”。

```

1. private void onSelectSwitchClicked(TableColumn selectSwitch) {
2.     int rowCount = tableModel.getRowCount();
3.     if (selectSwitch.getHeaderValue() == "全选") {
4.         selectSwitch.setHeaderValue("反选");
5.         for (int i = 0; i < rowCount; i++) {
6.             tableModel.setValueAt(true, i, 0);
7.         }
8.     } else {
9.         selectSwitch.setHeaderValue("全选");
10.        for (int i = 0; i < rowCount; i++) {
11.            tableModel.setValueAt(!(boolean) tableModel.getValueAt(i, 0), i, 0);
12.        }
13.    }
14.    jTable.updateUI();
15. }

```

2.3. ScrShotWindow(extends JWindow)

该类继承自 JWindow, 是截图绘图的总窗口, 响应了大量的事件。包含一个 ControlWindow 对象。该对象对其进行绘图和操作。

2.3.1. 主要方法介绍

● onMouseWheelMoved(MouseWheelEvent e)

该函数用于捕捉鼠标滚轮事件来调整画笔粗细以及字体大小

```
1. private void onMouseWheelMoved(MouseWheelEvent e) {
2.     if (e.getWheelRotation() == -1) {
3.         if (curWeight < 30) {
4.             curWeight += 0.5;
5.         }
6.     } else if (e.getWheelRotation() == 1) {
7.         if (curWeight > 1) {
8.             curWeight -= 0.5;
9.         }
10.    }
11.    if (grasControl.tempGra == null) {
12.        return;
13.    }
14.    grasControl.tempGra.setWeight(curWeight);
15.    repaint();
16. }
```

2.4. ControlWindow(extends JFrame)

该类主要是对 ScrShotWindow 的状态等进行控制。在最初进行设计时, 该类叫做 ToolsWindow, 只显示工具条界面并处理界面相关的事件, 对 ScrShotWindow 上绘制图形元素则由一个叫做 GrasControl 的类完成。但是在编写过程中发现其与 GrasControl 类高度耦合, 为了满足“高内聚低耦合”的理念, 最终将两个类合二为一。

2.4.1. 主要方法介绍

● fromJson(JSONObject control)

从一个 JSONObject 中解析各个图形元素, 从而达到复原截图时场景。这些图元都是通过基类 GraElement 继承而来。通过向下转换, 使得 GraElement 转换成对应的图元。

```
1. public void fromJson(JSONObject control) {
2.     chosenRec = new ChosenRectangle();
3.     chosenRec.fromString(control.getString("ChosenRec"));
4.     JSONArray graArr = control.getJSONArray("Gras");
5.     for (int i = 0; i < graArr.length(); i++) {
6.         JSONObject graJson = graArr.getJSONObject(i);
7.         int type = graJson.getInt("Type");
8.         switch (type) {
9.             case GraElement.STR:
10.                tempGra = new MyStringGra(graJson);
11.                addTempGra();

```

```

12.         break;
13.         case GraElement.LINE:
14.             tempGra = new MyPolyline(graJson);
15.             addTempGra();
16.             break;
17.         case GraElement.RECT:
18.             tempGra = new MyRectangle(graJson);
19.             addTempGra();
20.             break;
21.     }
22. }
23. }

```

● draw()

将各个图元绘制到 tempImage 中，然后将 tempImage 绘制到 ScrShotWindow 窗口中。

```

1. public void draw() {
2.     if (chosenRec == null) {
3.         scrShot.getGraphics().drawImage(darkerImage, 0, 0, null);
4.         return;
5.     }
6.     createTemp();
7.     Graphics2D g2d = (Graphics2D) tempImage.getGraphics();
8.     for (int i = 0; i < cursor; i++) {
9.         graElements.get(i).draw(g2d);
10.    }
11.    if (tempGra != null) {
12.        tempGra.draw(g2d);
13.    }
14.    scrShot.getGraphics().drawImage(tempImage, 0, 0, null);
15. }

```

● saveImageClipboard()

将对象放到剪贴板中需要该对象实现 Transferable 接口，而 BufferedImage 并没有实现，因此需要利用匿名内部类实现这个接口。

```

1. public void saveImageClipboard() {
2.     drawSaveImage();
3.     Toolkit.getDefaultToolkit().getSystemClipboard().setContents(new Transferable() {
4.
5.         @Override
6.         public DataFlavor[] getTransferDataFlavors() {
7.             return new DataFlavor[]{DataFlavor.imageFlavor};
8.         }
9.
10.        @Override
11.        public boolean isDataFlavorSupported(DataFlavor flavor) {
12.            return DataFlavor.imageFlavor.equals(flavor);
13.        }

```



```

14.
15.     @Override
16.     public Object getTransferData(DataFlavor flavor)
17.         throws UnsupportedOperationException, IOException {
18.         if (!DataFlavor.imageFlavor.equals(flavor)) {
19.             throw new UnsupportedOperationException(flavor);
20.         }
21.         return saveImage;
22.     }
23. }, null);
24. }

```

2.5. TextWindow(extends JFrame)

该类比较简单，仅有一个在 JScrollPane 布局下的 JTextArea 用于显示多行文字。

2.5.1. 主要方法介绍：

● copyToClipboard()

将 String 保存到剪贴板的方法较 BufferedImage 简单地多。

```

1. private void copyToClipboard() {
2.     Clipboard clipboard = Toolkit.getDefaultToolkit().getSystemClipboard();
3.     StringSelection selection = new StringSelection(jTextArea.getText());
4.     clipboard.setContents(selection, null);
5. }

```

2.6. ChosenRectangle(extends Rectangle)

该类继承自 Rectangle。主要用于形成一个可调整的矩形用于截图的框选区域。最初直接使用 Rectangle 类，单其功能不足，例如其 outcode 方法只能获得 WESN 四个方向，而在调整截图区域时需要 8 个方位。故继承 Rectangle 并对部分函数进行覆写。

2.6.1. 主要方法介绍：

● reSizeBy(Point p)

通过一个点对其大小方位进行调整。

```

1. public void reSizeBy(Point p){
2.     if(p.x>x){
3.         width=p.x-x;
4.     }else{
5.         width=x-p.x;
6.         x=p.x;
7.     }
8.     if(p.y>y){
9.         height=p.y-y;
10.    }else{
11.        height=y-p.y;
12.        y=p.y;
13.    }

```

```

14.     width=width==0?1:width;
15.     height=height==0?1:height;
16. }

```

● toString 和 fromString 方法

便于在 ControlWindow 类的 toJson 和 fromJson 中将其保存为 JSON 格式，并从中还原。

```

1. @Override
2. public String toString() {
3.     return "(" + x + ',' + y + ',' + width + ',' + height + ")";
4. }
5.
6. public void fromString(String str) {
7.     String substr = str.substring(1, str.length() - 1);
8.     String[] xywhStrings = substr.split(",");
9.     x = Integer.parseInt(xywhStrings[0]);
10.    y = Integer.parseInt(xywhStrings[1]);
11.    width = Integer.parseInt(xywhStrings[2]);
12.    height = Integer.parseInt(xywhStrings[3]);
13. }

```

六、 遇到的问题解决及优化

1. 截图分辨率降低

在 jdk 13 下且设置了屏幕缩放的 windows 环境，Robot 对象的 createScreenCapture() 方法获得的截图会模糊。分辨率只有 $\frac{\text{真实分辨率}}{\text{缩放比例}}$ 。将 jdk 目录下 java.exe 兼容性高 DPI 设置为“系统”，或者用 jdk 8 即可解决。

2. 添加按钮图标后编译报错

使用 NetBeans 等 IDE 为界面的按钮等控件添加图片时，实际上会使用 getClass().getResource(相对于包名根目录的路径)，而编译的 class 文件在项目的 target 文件夹下，这会使得在运行时找不到图片资源。将图片资源拷贝到 target 目录下的对应位置即可解决。

3. 批量添加 RadioButton 事件

设置颜色选择 RadioButton 时，由于有 10 个 RadioButton，如果手动为每个按钮添加事件处理，则工作量大且影响可读性。添加 ButtonGroup 后可以实现 group 里单选，但是仍无法统一添加事件。尝试获得 ButtonGroup 里所有按钮，然后遍历为其添加事件（应该有更好的方法）。代码如下：

```

1. private void initColorChooser() {
2.     Enumeration<AbstractButton> colorElements = colorGroup.getElements();
3.     while (colorElements.hasMoreElements()) {
4.         AbstractButton colorButton = colorElements.nextElement();
5.         colorButton.addMouseListener(new MouseAdapter() {
6.             @Override
7.             public void mouseClicked(MouseEvent e) {
8.                 scrShot.setCurColor(colorButton.getBackground());
9.             }
10.        });

```

```
11.     }
12. }
```

4. 多线程优化

由于 OCR 识别访问服务器需要一定的时间，点击 OCR 识别后会造成程序“假死”，故需要单独开一个线程处理 OCR 识别。

```
1. private void onOCRClicked() {
2.     TextWindow textWindow = new TextWindow();
3.     Thread t = new Thread(() -> {
4.         String result = toOCR();
5.         textWindow.setText(result);
6.     });
7.     t.start();
8. }
```

5. 使用内部类 MyDate(extends Date)

在 RecordsWindow 类中定义了 MyDate 这个类。构建这个类是为了让 TableModel 存储 Date 类型并且能按照指定格式显示。之所以要存储 Date 类型，而不直接存储 String，是因为在数据库中删除记录的相关操作需要将时间戳作为查询条件。如果存储字符串，则还需字符串转 Date，然后再转时间戳来进行查。不过数据库里也能存放 date 类型，但我没有去尝试，而是只存储了时间戳。

```
1. class MyDate extends Date {
2.
3.     public MyDate(Long time) {
4.         super(time);
5.     }
6.     @Override
7.     public String toString() {
8.         return dateFormat.format(this);
9.     }
10. }
```

6. 关于太极图案主界面

最开始设计时，主窗口就两个按钮——“开始截图”和“查看记录”。想对界面进行个性化但又没合适的想法。大概是刚从一阵子体育太极拳考试的支配中走出来，突然想到可以用太极的阴阳去对应两个按钮。在某种意义上，阳代表“新”，刚好与开始截图的操作相对应；阴代表“旧”，与查看历史记录的操作相对应。当然这只是很牵强的说法，并没有比较强的设计理念，单纯觉得好看并且刚好对应两个按钮（如果有五个按钮，我或许会做个五角星）并且好画罢了。

具体实现：

由于在之前对 NetBeans 使用操作中已经大致明白窗口属性的常见属性设置，注意到有 `setShape(Shape s)` 方法，只需要传入实现 Shape 接口的对象就可设置窗口的形状。通过 jdk 文档，查阅 Shape，发现其接口下实现了 `Ellipse2D.Float` 类，即浮点精度的椭圆类。通过 `setShape(new Ellipse2D.Float(0,0,this.getWidth(),this.getWidth()))` 即可将窗口设为圆形，直径为窗口的宽。

在科学计算与 Python 语言这门课中，我曾利用 turtle 库绘制过太极图形。尽管 turtle 的绘画逻辑与 Graphics 的逻辑不同，但稍微改改就能用了。Graphics 绘制好太极图案后，想实现点击黑白区域产生不同效果，最初的想法是判断鼠标点击位置的颜色，但觉得实现起来效率太低。然后想起在

写 ChosenRectangle 这个类时，从 Rectangle 继承得到，其中 Rectangle 有 contains(Point p) 的方法用来判断一个点不在在矩形内。又经过查找后，找到 Area 类，可以通过 Shape 得到，并且还能进行图像的交并补等运算。于是通过 Area 对象的交并补获得了阴阳两个形状，方便判断鼠标所在位置在哪个区域内，由于 Area 类也实现了 Shape 接口，所以用 Graphics 绘制时也可直接绘制这两个区域。