MiniCAD 实验报告

author： 3200105712 李飞扬

1 实验内容和要求

用Java的awt和swing做一个简单的绘图工具，以CAD的方式操作，能放置直线、矩形、圆和文字，能选中图形，修改参数，如颜色等，能拖动图形和调整大小，可以保存和恢复。功能请参考视频演示。

具体支持的功能有：

- 按下鼠标左键拖动，直到松开鼠标左键绘制新的图形。
- 可以绘制直线、矩形、圆形和字符串四种基本图形。用户可以输入字符串。
- 删除画板上的图形。
- 拖动图形。选中画板上的某个图形，用鼠标拖动图形位置。
- 改变图形粗细。
- 改变图形大小。
- 改变图形颜色。选中画板上的某个图形，点击颜色按钮，该图形颜色变为颜色按钮上的颜色。
- 将画板上的图形导出到文件。
- 将文件内容导入到画板上。

2 实验环境

- MacOS Ventura 13.0.1
- IDE：IntelliJ IDEA 2022.2.3 aarch64
- JDK：Java 18 Amazon Corretto version 18

3 技术概述

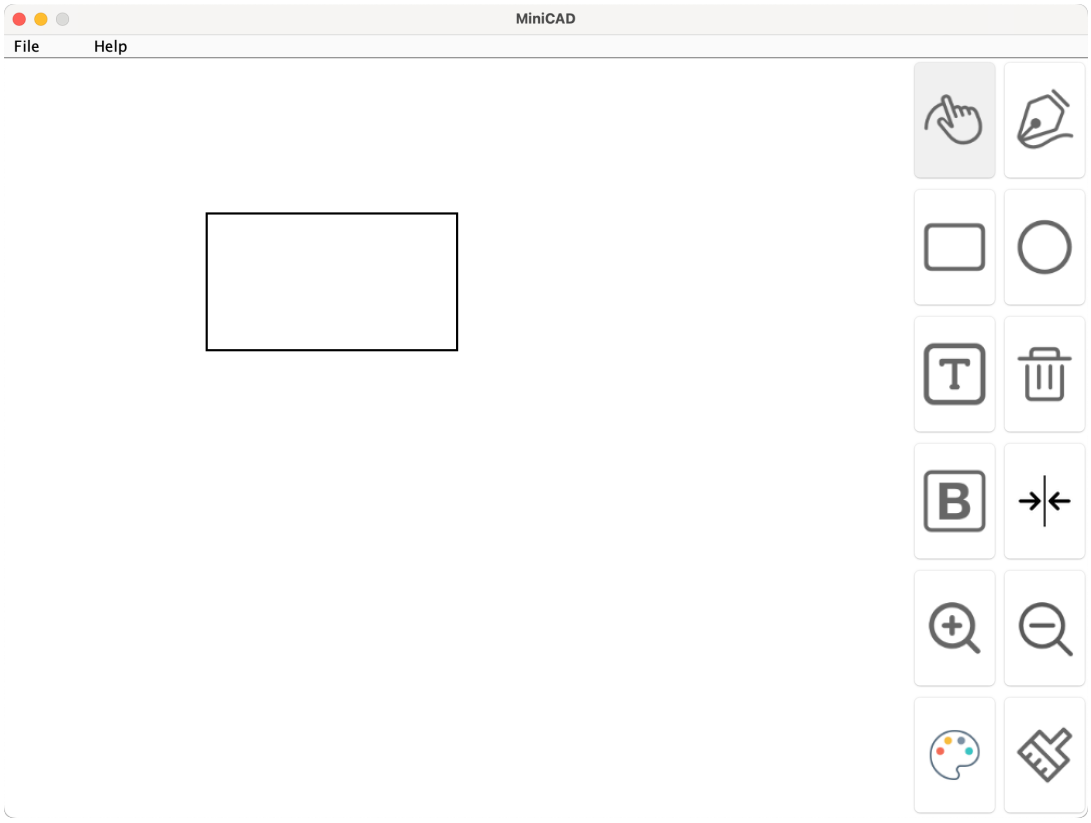
本程序采用 Model–View–Controller（模型–视图–控制器）模式即 MVC 模式来架构代码。

- Model（模型） – 模型层是最底下的一层，代表一个存取数据的对象。 它也可以带有逻辑，在数据变化时更新控制器。
- View（视图） – 视图层是最上面的一层，代表模型包含的数据的可视化。
- Controller（控制器） – 控制层是中间一层，作用于模型和视图上。它控制数据流向模型对象，并在数据变化时更新视图。它使视图与模型分离开。

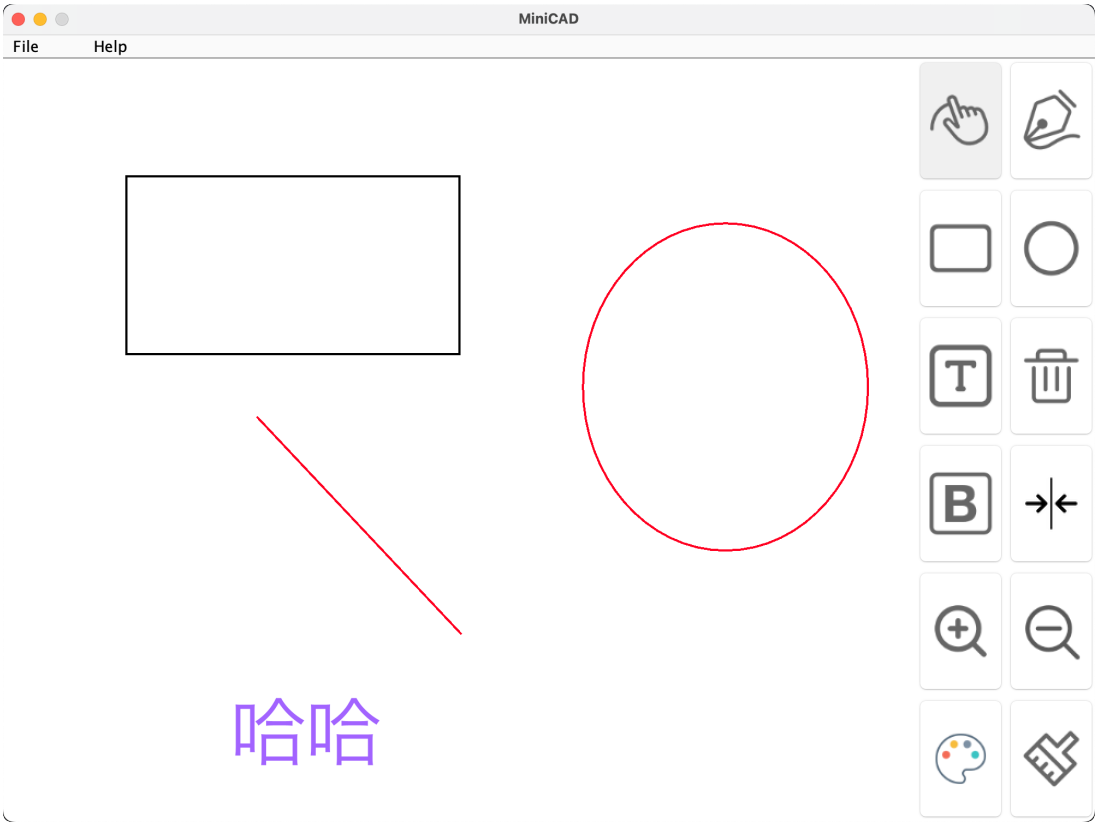
这三层是紧密联系在一起的，但又是互相独立的，每一层内部的变化不影响其他层。每 一层都对外提供接口（Interface），供上面一层调用。这样一来，软件就可以实现模块化，修改外观或者变更数据都不用修改其他层，大大方便了维护和升级。

4 成果与功能展示

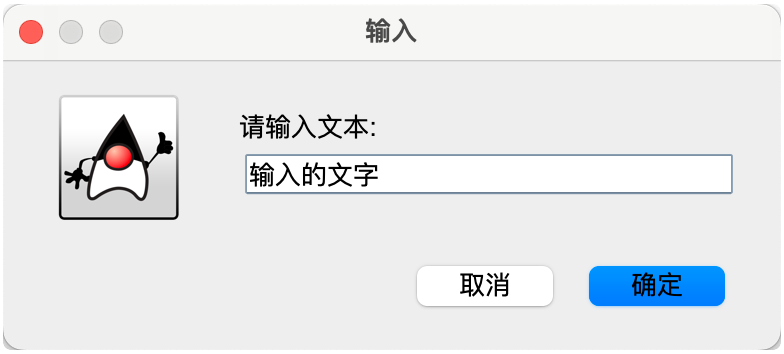
- 成果如图所示：主体分为三部分，分别是菜单栏，按钮操作栏和主画板。



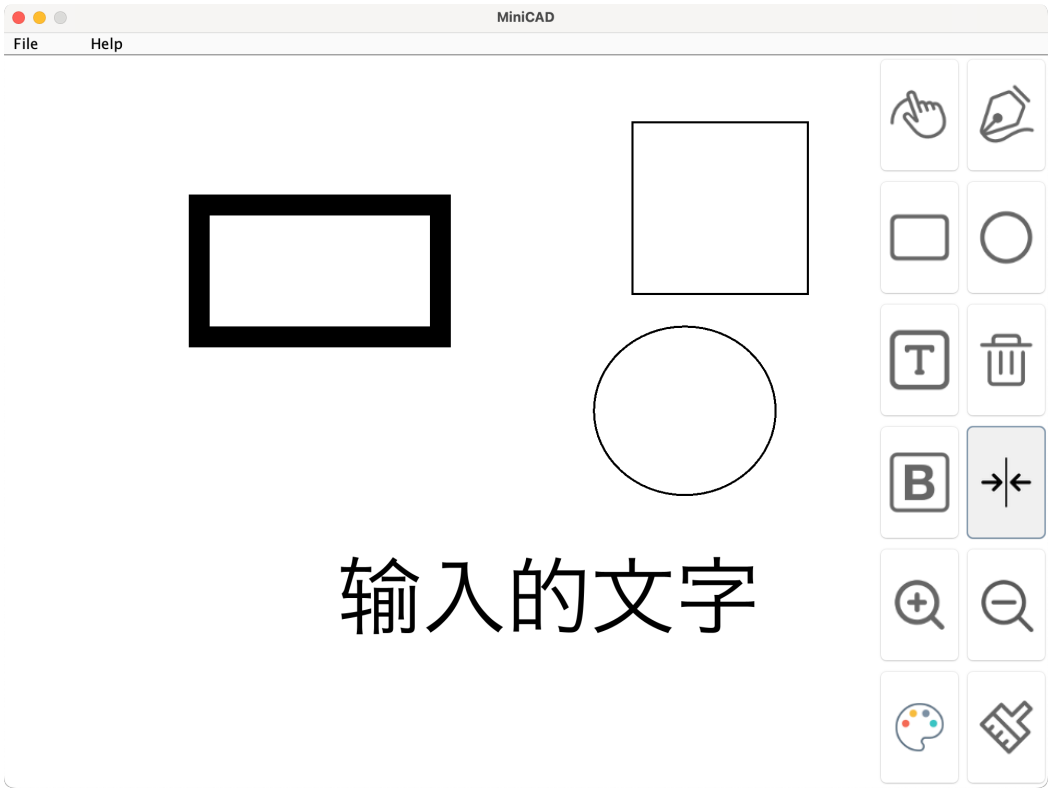
- 点击选定按钮，可以选中一个图形。选中后可以拖动鼠标来移动对应的图形。
- 点击画笔按钮，可以拖动鼠标来画直线。
- 点击矩形和圆形按钮，可以拖动鼠标来画矩形和圆。



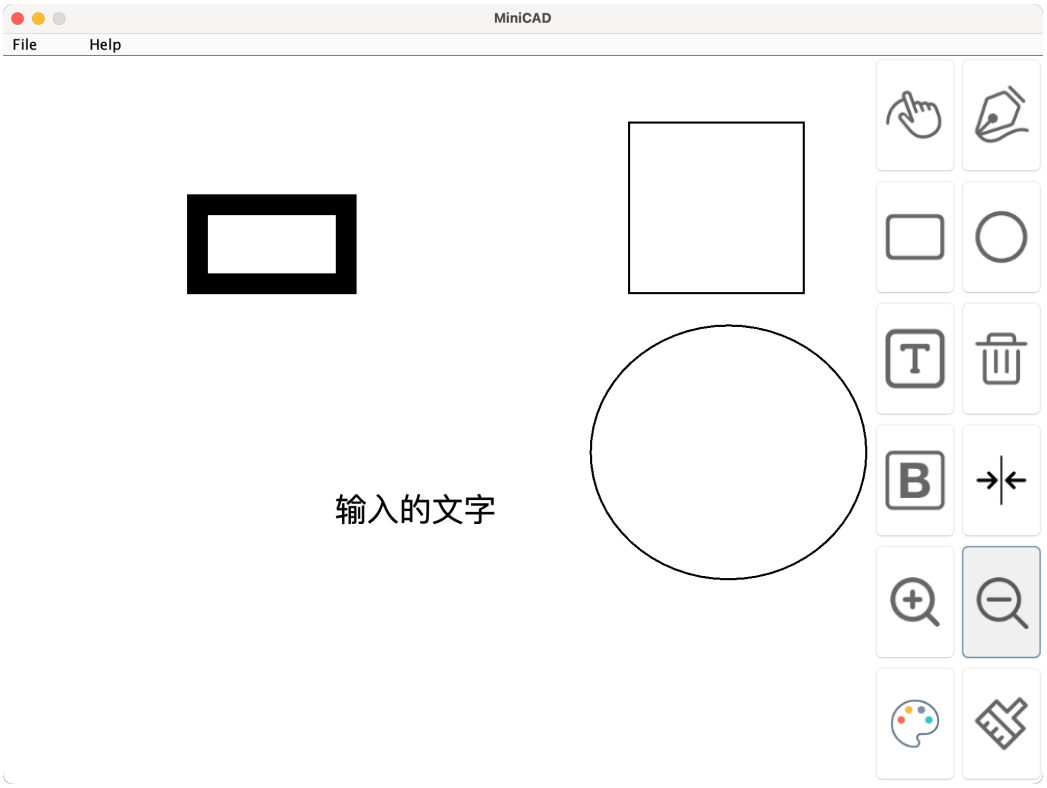
- 点击文字按钮，可以输入文字，并拖动鼠标来改变大小：



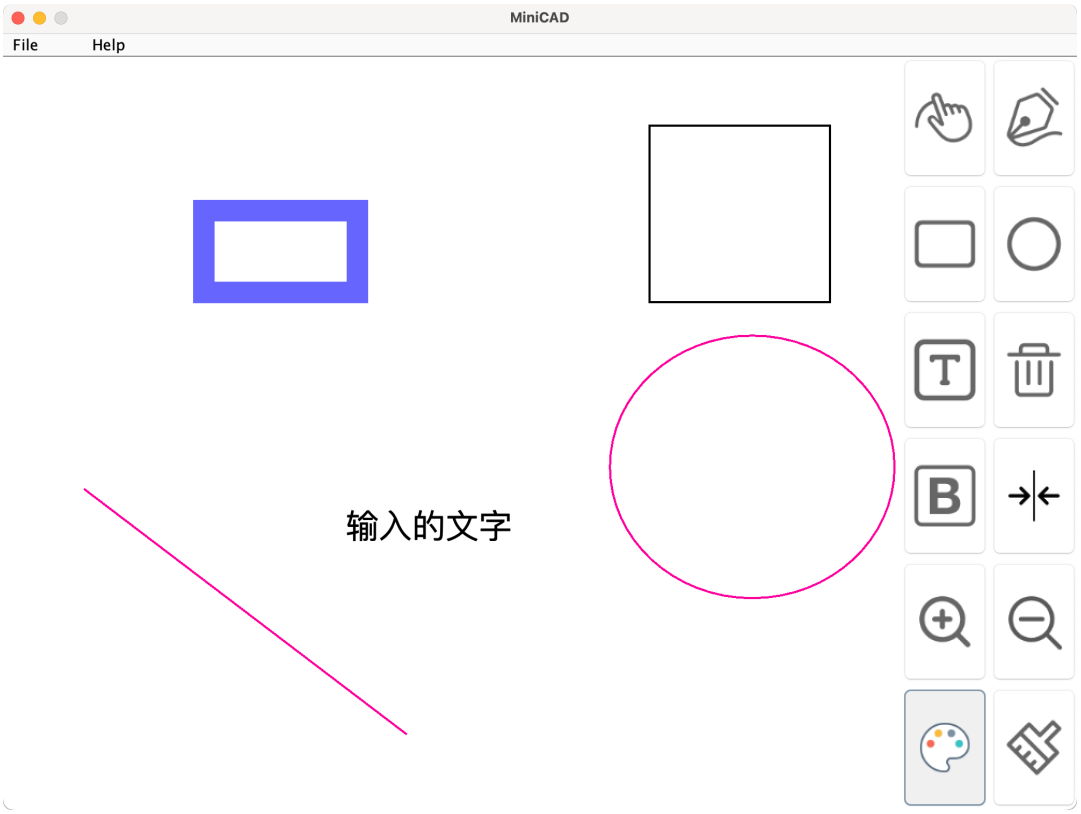
- 在选中后，点击删除按钮可以删除一个图形。
- 在选中后，可以将一个图形加粗或变细。



- 在选中后，可以点击放大或缩小按钮，来变大和变小文字或者图形。



- 可以点击颜色按钮来改变当前的画笔颜色。如果已经选中了一个图形和文字，还可以直接改变它的颜色。



- 可以点击清屏按钮清除所有图形。
- 可以点击菜单栏，选择保存或者加载一个 cad 文件。
- 可以点击菜单栏的 Help 选项查看信息。

6 技术实现

6.1 Model数据结构

本项目的数据结构较为简单。只需要保存所有的图形即可。我设计了 `Shape` 主类，其他的各种图形都是 `Shape` 的继承子类。

Java | 复制代码

```java
public class model {
    public static ArrayList<Shape> shapes = new ArrayList<Shape>(); //存放所有图形
    public static void saveFile() {
        JFileChooser chooser=new JFileChooser();
        FileNameExtensionFilter filter = new FileNameExtensionFilter(
            "Cad画板文件(*.cad)", "cad");
        chooser.setFileFilter(filter);
        chooser.showSaveDialog(null);
        chooser.setDialogTitle("保存文件");
        File file = chooser.getSelectedFile();
        if(file != null) {
            try {
                ObjectOutputStream out = new ObjectOutputStream(new FileOutputStream(file));
                out.writeObject(shapes);
                JOptionPane.showMessageDialog(null,"保存成功!");
                out.close();
            }
            catch (IOException e) {
                e.printStackTrace();
                JOptionPane.showMessageDialog(null,"保存失败!");
            }
        }

    }
    public static void loadFile() {
        int confirmDialog = JOptionPane.showConfirmDialog(null, "是否保存文件", "提示信息", 0);
        if(confirmDialog == 0) {
            saveFile();
        }
        try {
            JFileChooser chooser = new JFileChooser();
            cadFileFilter filter = new cadFileFilter();
            chooser.setFileFilter(filter);
            chooser.addChoosableFileFilter(filter);
            chooser.setDialogTitle("请选择保存的cad文件打开：");
            chooser.showOpenDialog(null);
            File file = chooser.getSelectedFile();
            if(file==null){
                JOptionPane.showMessageDialog(null, "未选择文件!");
            }
            else {

```

```
                ObjectInputStream in = new ObjectInputStream(new FileInput
Stream(file));
                shapes = ((ArrayList<Shape>)in.readObject());
                view.drawArea.repaint();
                in.close();
            }
        }
        catch (Exception e) {
            e.printStackTrace();
            JOptionPane.showMessageDialog(null,"打开失败!");
        }
    }
}
```

`Shape` 类的设计如下：

- 提供抽象方法 `draw` 和 `isSelected` 由子类实现。并提供点到直线距离、点到点距离的静态方法：

Java | 复制代码

```java
public abstract class Shape implements Serializable{

    @Serial
    private static final long serialVersionUID = 1L;
    public ArrayList<Point> points = new ArrayList<>(); //存储左上角和右下角顶点
    public Color color;
    public float thick;
    Shape(Point a, Point b) {
        points.add(a);
        points.add(b);
        color = Color.BLACK;
        thick = 2.0f;
    }
    public abstract void draw(Graphics g);
    public abstract boolean isSelected(Point p);
    static double pointDis(Point a, Point b) {
        return (a.x - b.x)*(a.x - b.x) + (a.y - b.y)*(a.y - b.y);
    }
    static double pointToLine(Point A, Point B, Point P) {
        double r = ((P.x - A.x)*(B.x - A.x) + (P.y - A.y)*(B.y - A.y)) / pointDis(A,B);
        if (r <= 0) return Math.sqrt(pointDis(A, P));
        else if (r >= 1) return Math.sqrt(pointDis(B, P));
        else {
            double AC = r * Math.sqrt(pointDis(A, B));
            return Math.sqrt(pointDis(A, P)-AC * AC);
        }
    }
}
```

6.2 View UI 设计

- 主体窗口加入菜单栏、按钮栏和画布。

Java | 复制代码

```java
package View;

import javax.swing.*;
import java.awt.*;

public class view extends JFrame {
    public static canvas drawArea = new canvas();
    public view() {
        super("MiniCAD");
        setSize(1000, 750);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

        View.menuBar menuBar = new menuBar();
        setJMenuBar(menuBar.menuBar);

        Panel myButtonBar = new buttonBar();
        add(myButtonBar, BorderLayout.EAST);

        drawArea.setPreferredSize(null);
        add(drawArea);

        init();
    }
    private void init() {
//        pack();
        setVisible(true);
        setResizable(false);
        setBackground(Color.WHITE);
        setLocationRelativeTo(null);
    }
}
```

- 按钮栏是主体的 UI 部分：
  - 整体布局采用了 `GridLayout` 。

Java | 复制代码

```java
public class buttonBar extends Panel {
    ImageIcon lineIcon = new ImageIcon("resource/line.png");
    public JToggleButton lineButton = new JToggleButton(lineIcon);
    ImageIcon rectangleIcon = new ImageIcon("resource/rectangle.png");
    public JToggleButton rectangleButton = new JToggleButton(rectangleIcon
);
    ImageIcon circleIcon = new ImageIcon("resource/circle.png");
    public JToggleButton circleButton = new JToggleButton(circleIcon);
    ImageIcon textIcon = new ImageIcon("resource/text.png");
    public JToggleButton textButton = new JToggleButton(textIcon);
    ImageIcon boldIcon = new ImageIcon("resource/bold.png");
    public JToggleButton boldButton = new JToggleButton(boldIcon);
    ImageIcon colorIcon = new ImageIcon("resource/color.png");
    public JToggleButton colorButton = new JToggleButton(colorIcon);
    ImageIcon bigIcon = new ImageIcon("resource/big.png");
    public JToggleButton bigButton = new JToggleButton(bigIcon);
    ImageIcon smallIcon = new ImageIcon("resource/small.png");
    public JToggleButton smallButton = new JToggleButton(smallIcon);
    ImageIcon selectIcon = new ImageIcon("resource/select.png");
    public JToggleButton selectButton = new JToggleButton(selectIcon);
    ImageIcon cleanIcon = new ImageIcon("resource/clean.png");
    public JToggleButton cleanButton = new JToggleButton(cleanIcon);
    ImageIcon deleteIcon = new ImageIcon("resource/delete.png");
    public JToggleButton deleteButton = new JToggleButton(deleteIcon);
    ImageIcon thinIcon = new ImageIcon("resource/thin.png");
    public JToggleButton thinButton = new JToggleButton(thinIcon);

    public static Map<JToggleButton, String> map = new HashMap<>();

    buttonBar() {
        setLayout(new GridLayout(6,2,3,3));

        add(selectButton);
        add(lineButton);
        add(rectangleButton);
        add(circleButton);
        add(textButton);
        add(deleteButton);
        add(boldButton);
        add(thinButton);
        add(bigButton);
        add(smallButton);
        add(colorButton);
        add(cleanButton);

```

```
            map.put(lineButton, "line");
            map.put(rectangleButton, "rectangle");
            map.put(circleButton, "circle");
            map.put(textButton, "text");
            map.put(selectButton, "select");
            map.put(boldButton, "bold");
            map.put(bigButton, "big");
            map.put(smallButton, "small");
            map.put(colorButton, "color");
            map.put(cleanButton, "clean");
            map.put(deleteButton, "delete");
            map.put(thinButton, "thin");

            lineButton.addActionListener(new buttonStatusAutoChange());
            rectangleButton.addActionListener(new buttonStatusAutoChange());
            circleButton.addActionListener(new buttonStatusAutoChange());
            textButton.addActionListener(new buttonStatusAutoChange());
            selectButton.addActionListener(new buttonStatusAutoChange());
            boldButton.addActionListener(new buttonStatusAutoChange());
            bigButton.addActionListener(new buttonStatusAutoChange());
            smallButton.addActionListener(new buttonStatusAutoChange());
            colorButton.addActionListener(new buttonStatusAutoChange());
            cleanButton.addActionListener(new buttonStatusAutoChange());
            deleteButton.addActionListener(new buttonStatusAutoChange());
            thinButton.addActionListener(new buttonStatusAutoChange());
        }
    }
```

6.3 Controller 控制器逻辑

- 控制器逻辑分为几个部分：
  - 文件类型过滤器，用于在读取本地文件时过滤下 `.cad` 文件

Java | 复制代码

```java
public class cadFileFilter extends javax.swing.filechooser.FileFilter {
    public boolean accept(File f) {
        if (f.isDirectory()) return true;
        return f.getName().endsWith(".cad");  //选择以.cad为后缀的文件
    }
    public String getDescription() {
        return "Cad画板文件(*.cad)";
    }
}
```

- 按钮状态改变控制器，保证每时只能有一个按钮处于按下状态，同时根据按下的按钮调整当前的画布状态：

Java | 复制代码

```java
public static class buttonStatusAutoChange implements ActionListener {
    @Override
    public void actionPerformed(ActionEvent e) {
        JToggleButton toggleBtn = (JToggleButton) e.getSource();
        boolean status = !toggleBtn.isSelected();
        String string = buttonBar.map.get(toggleBtn);
        if(!status) {
            for (Map.Entry<JToggleButton, String> entry : View.buttonBar.map.entrySet()) {
                String mapKey = entry.getValue();
                if(!string.equals(mapKey)) {
                    entry.getKey().setSelected(false);
                }
            }
            statusNow = string;
//                System.out.println(statusNow);
        } else {
            toggleBtn.setSelected(true);
//                System.out.println(statusNow);
        }
        if(statusNow.equals("line") || statusNow.equals("rectangle") || statusNow.equals("circle")) selectedShape = null;
        View.view.drawArea.setCursor(Cursor.getPredefinedCursor(Cursor.DEFAULT_CURSOR));
        switch (statusNow) {
            case "color" -> {
                selectedColor = JColorChooser.showDialog(view, "颜色选择器", Color.WHITE);
                if (selectedShape != null) {
                    selectedShape.color = selectedColor;
                    View.view.drawArea.repaint();
                }
            }
            case "text" -> {
                textString = JOptionPane.showInputDialog("请输入文本: ");
                statusNow = "text";
            }
            case "clean" -> {
                Model.model.shapes.clear();
                View.view.drawArea.repaint();
            }
            case "bold" -> {
                if (selectedShape != null) {
                    selectedShape.thick += 2.0f;
                    View.view.drawArea.repaint();
```

```
                }
            }
            case "big" -> {
                if (selectedShape != null) {
                    Point A = selectedShape.points.get(0);
                    Point B = selectedShape.points.get(1);
                    selectedShape.points.set(1, new Point((int)(1.05 * B.x - 0.05 * A.x), (int)(1.05 * B.y - 0.05 * A.y)));
                    View.view.drawArea.repaint();
                }
            }
            case "small" -> {
                if (selectedShape != null) {
                    Point A = selectedShape.points.get(0);
                    Point B = selectedShape.points.get(1);
                    selectedShape.points.set(1, new Point((int)(0.05 * A.x + 0.95 * B.x), (int)(0.05 * A.y + 0.95 * B.y)));
                    View.view.drawArea.repaint();
                }
            }
            case "thin" -> {
                if (selectedShape != null) {
                    if(selectedShape.thick > 2.0f) selectedShape.thick -= 1.5f;
                    View.view.drawArea.repaint();
                }
            }
            case "delete" -> {
                if (selectedShape != null) {
                    Model.model.shapes.remove(selectedShape);
                    View.view.drawArea.repaint();
                }
            }
        }

    }
}
```

- 鼠标动作监听器：处理画笔相关逻辑

Java | 复制代码

```java
public static class mouseListener implements MouseListener, MouseMotionListener {

    Shape newShape;
    @Override
    public void mouseClicked(MouseEvent e) {

    }

    @Override
    public void mousePressed(MouseEvent e) {
        if(statusNow.equals("idle")) return;
        if(statusNow.equals("select")) {
            selectedShape = null;
            View.view.drawArea.setCursor(Cursor.getPredefinedCursor(Cursor.DEFAULT_CURSOR));
            for (Shape a: Model.model.shapes) {
                if(a.isSelected(e.getPoint())) {
                    selectedShape = a;
                    startPoint = a.points.get(0);
                    endPoint = a.points.get(1);
                    dragPoint = e.getPoint();
                    View.view.drawArea.setCursor(Cursor.getPredefinedCursor(Cursor.HAND_CURSOR));
                    break;
                }
            }
            return ;
        }
        Point p = e.getPoint();
        switch (statusNow) {
            case "line":
                newShape = new Line(p, p, selectedColor);
                Model.model.shapes.add(newShape);
                View.view.drawArea.repaint();
                break;
            case "rectangle":
                newShape = new Rectangle(p, p, selectedColor);
                Model.model.shapes.add(newShape);
                View.view.drawArea.repaint();
                break;
            case "circle":
                newShape = new Circle(p, p, selectedColor);
                Model.model.shapes.add(newShape);
                View.view.drawArea.repaint();
```

```
                break;
            case "text":
                newShape = new Text(p, p, selectedColor, textString);
                Model.model.shapes.add(newShape);
                View.view.drawArea.repaint();
                break;
        }
        View.view.drawArea.repaint();
    }

    @Override
    public void mouseReleased(MouseEvent e) {

    }

    @Override
    public void mouseEntered(MouseEvent e) {

    }

    @Override
    public void mouseExited(MouseEvent e) {

    }

    @Override
    public void mouseDragged(MouseEvent e) {
        if(statusNow.equals("rectangle") || statusNow.equals("circle") || statusNow.equals("line") || statusNow.equals("text")){
            Model.model.shapes.get(Model.model.shapes.size() - 1).points.set(1, e.getPoint());
            View.view.drawArea.repaint();
        } else if (statusNow.equals("select") && selectedShape != null) {
            Point dragToPoint = e.getPoint();
            int dx = dragToPoint.x - dragPoint.x, dy = dragToPoint.y - dragPoint.y;
            selectedShape.points.set(0, new Point(startPoint.x + dx, startPoint.y + dy));
            selectedShape.points.set(1, new Point(endPoint.x + dx, endPoint.y + dy));
            View.view.drawArea.repaint();
        }
    }
    @Override
    public void mouseMoved(MouseEvent e) {

    }
}
```

- 其他监听器：如点击信息时弹出窗口，点击保存时进行保存，点击加载时进行加载等。

Java 复制代码

```java
public static class saveListener implements ActionListener {
    @Override
    public void actionPerformed(ActionEvent e) {
        Model.model.saveFile();
    }
}
public static class loadListener implements ActionListener {
    @Override
    public void actionPerformed(ActionEvent e) {
        Model.model.loadFile();
    }
}
public static class messageListener implements ActionListener {
    @Override
    public void actionPerformed(ActionEvent e) {
        JOptionPane.showMessageDialog(null,"* @author: Li Feiyang" + System.getProperty("line.separator") +
                "* @student number: 3200105712" + System.getProperty("line.separator") +
                "* @created: 2022-11-16" + System.getProperty("line.separator") +
                "* @purpose: Project2 MiniCAD of Java Application Technology of ZJU" + System.getProperty("line.separator") +
                "*  Copyright 2022 All rights reserved.");
    }
}
```