Java实验报告

软件学院 软件工程4班 2113850 李鹏

——实验题目：Java GUI实现画笔小程序

设计思路：

1. 界面设计：需要一个能够显示绘画区域的画布和一些绘图工具，比如选择不同的形状、颜色和填充模式等。在画布上绘制图形时，可以使用鼠标或者触摸屏对图形进行控制和编辑。同时，还需要提供保存和打开文件的功能，用户可以将绘制的图形保存成文件并且以后可以再次打开进行编辑。
2. 绘画功能：需要实现基本的绘画功能，包括直线、矩形、椭圆等形状的绘制，并且支持文本的输入和编辑。为了方便用户进行编辑，可以在绘制过程中实时显示绘制结果，并且支持多次撤销和重做操作。
3. 颜色和填充模式：用户可以根据自己的需求选择绘制图形的背景色和前端颜色，并且可以选择填充模式或者非填充模式。在选择颜色时，可以使用调色板来提供更方便的选择方式。
4. 图形保存和打开：当用户完成绘制时，可以将绘制的图形以文件形式保存在本地，以便于日后打开，同时也可以打开之前保存的文件进行编辑。为了实现这个功能，可以使用对象输入和输出流来进行序列化和反序列化，以便于将图形数据保存为文件并且从文件中读取并显示图形数据。

这些是实现提供直线、矩形、椭圆等形状和文字绘制，以及背景色和前端颜色可选，图形的绘制可以选择填充模式或者非填充模式，图形可以保存，并且可以打开已保存的图形文件的思路，可以根据需求和技术选择相应的方法和技术。

首先定义一些主要的变量和类：

private final JToggleButton lineBtn = new JToggleButton("直线");

private final JToggleButton rectBtn = new JToggleButton("矩形");

private final JToggleButton ovalBtn = new JToggleButton("椭圆");

private final JToggleButton textBtn = new JToggleButton("文字");

private final JToggleButton fillBtn = new JToggleButton("填充");

private final JButton clearBtn = new JButton("清空");

private final JComboBox<String> bgBox = new JComboBox<>();

private final JComboBox<String> fgBox = new JComboBox<>();

private final JTextArea inputArea = new JTextArea(5, 15);

private final JLabel statusLabel = new JLabel("状态栏");

private final DrawingPanel drawingPanel = new DrawingPanel();

// 定义一个 ArrayList 用于存储绘制的图形

private final ArrayList<Shape> shapes = new ArrayList<>();

// 定义一个枚举类用于表示不同的操作类型

private enum OperationType {

DRAW, MOVE, RESIZE;

}

// 定义一个内部类用于绘制图形

private class DrawingPanel extends JPanel {

// 当前操作类型

private OperationType operationType = OperationType.DRAW;

// 当前选择的图形类型

private ShapeType shapeType = ShapeType.LINE;

// 当前选择的颜色

private Color bgColor = Color.WHITE;

private Color fgColor = Color.BLACK;

// 当前是否选择填充模式

private boolean isFillMode = false;

// 当前绘制的图形对象

private Shape currentShape = null;

// 鼠标操作相关的变量

private int startX, startY, endX, endY;

private boolean isDragging = false;

// 用于记录当前选中的图形和鼠标的偏移量

private Shape selectedShape = null;

private int dx = 0, dy = 0;

}

初始化界面，包括添加组件、设置布局等：

public void init() {

// 添加按钮组件

ButtonGroup shapeGroup = new ButtonGroup();

shapeGroup.add(lineBtn);

shapeGroup.add(rectBtn);

shapeGroup.add(ovalBtn);

shapeGroup.add(textBtn);

JPanel shapePanel = new JPanel();

shapePanel.setBorder(BorderFactory.createTitledBorder("形状"));

shapePanel.add(lineBtn);

shapePanel.add(rectBtn);

shapePanel.add(ovalBtn);

shapePanel.add(textBtn);

fillBtn.addActionListener(this::onFillBtnClicked);

clearBtn.addActionListener(this::onClearBtnClicked);

JPanel optPanel = new JPanel();

optPanel.setLayout(new FlowLayout(FlowLayout.LEFT));

optPanel.setBorder(BorderFactory.createTitledBorder("操作"));

optPanel.add(fillBtn);

optPanel.add(clearBtn);

// 添加颜色选择器

for (Map.Entry<String, Color> entry : COLORS.entrySet()) {

bgBox.addItem(entry.getKey());

fgBox.addItem(entry.getKey());

}

bgBox.addActionListener(this::onBgBoxSelected);

fgBox.addActionListener(this::onFgBoxSelected);

JPanel colorPanel = new JPanel();

colorPanel.setBorder(BorderFactory.createTitledBorder("颜色"));

colorPanel.add(new JLabel("背景："));

colorPanel.add(bgBox);

colorPanel.add(new JLabel("前景："));

colorPanel.add(fgBox);

// 添加输入框和状态栏

JScrollPane inputScrollPane = new JScrollPane(inputArea,

ScrollPaneConstants.VERTICAL\_SCROLLBAR\_ALWAYS,

ScrollPaneConstants.HORIZONTAL\_SCROLLBAR\_NEVER);

inputScrollPane.setBorder(BorderFactory.createTitledBorder("文本"));

JPanel statusPanel = new JPanel();

statusPanel.setBorder(BorderFactory.createEtchedBorder());

statusPanel.setLayout(new BorderLayout());

statusPanel.add(statusLabel, BorderLayout.WEST);

// 添加绘图面板

drawingPanel.addMouseListener(new MouseAdapter() {...});

drawingPanel.addMouseMotionListener(new MouseAdapter() {...});

drawingPanel.setBorder(BorderFactory.createEtchedBorder());

// 添加组件到容器中

Container container = getContentPane();

container.setLayout(new BorderLayout());

container.add(shapePanel, BorderLayout.NORTH);

container.add(optPanel, BorderLayout.CENTER);

container.add(colorPanel, BorderLayout.SOUTH);

container.add(inputScrollPane, BorderLayout.EAST);

container.add(statusPanel, BorderLayout.SOUTH);

container.add(drawingPanel, BorderLayout.CENTER);

// 设置窗口属性

setTitle("画笔小程序");

setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

pack();

setVisible(true);

}

各个按钮添加事件处理函数:

private void onFillBtnClicked(ActionEvent e) {

JToggleButton btn = (JToggleButton) e.getSource();

drawingPanel.isFillMode = btn.isSelected();

}

private void onClearBtnClicked(ActionEvent e) {

drawingPanel.shapes.clear();

drawingPanel.currentShape = null;

drawingPanel.repaint();

}

private void onBgBoxSelected(ActionEvent e) {

JComboBox<?> box = (JComboBox<?>) e.getSource();

String colorName = (String) box.getSelectedItem();

drawingPanel.bgColor = COLORS.get(colorName);

drawingPanel.repaint();

}

private void onFgBoxSelected(ActionEvent e) {

JComboBox<?> box = (JComboBox<?>) e.getSource();

String colorName = (String) box.getSelectedItem();

drawingPanel.fgColor = COLORS.get(colorName);

drawingPanel.repaint();

}

private void onShapeButtonClicked(ShapeType shapeType) {

drawingPanel.operationType = OperationType.DRAW;

drawingPanel.shapeType = shapeType;

drawingPanel.selectedShape = null;

}

处理鼠标事件是本程序的核心。在鼠标按下时记录起始坐标，在拖拽时不断更新终点坐标，并根据当前操作类型进行相应的操作。其中，对于每个添加到画板上的图形，都可以通过鼠标拖拽进行移动和调整大小的操作。

private class MouseEventHandler extends MouseAdapter {

@Override

public void mousePressed(MouseEvent e) {

if (e.getButton() == MouseEvent.BUTTON1) {

drawingPanel.isDragging = true;

drawingPanel.startX = e.getX();

drawingPanel.startY = e.getY();

switch (drawingPanel.operationType) {

case DRAW:

Shape shape = createShape(drawingPanel.shapeType, drawingPanel.startX, drawingPanel.startY, 0, 0);

drawingPanel.currentShape = shape;

break;

case MOVE:

for (Shape shape : drawingPanel.shapes) {

if (shape.contains(e.getPoint())) {

drawingPanel.selectedShape = shape;

drawingPanel.dx = e.getX() - shape.x;

drawingPanel.dy = e.getY() - shape.y;

break;

}

}

break;

case RESIZE:

for (Shape shape : drawingPanel.shapes) {

if (shape.contains(e.getPoint())) {

drawingPanel.selectedShape = shape;

drawingPanel.dx = e.getX() - (shape.x + shape.width);

drawingPanel.dy = e.getY() - (shape.y + shape.height);

break;

}

}

break;

}

statusLabel.setText(String.format("起点：(%d,%d)", e.getX(), e.getY()));

}

}

@Override

public void mouseDragged(MouseEvent e) {

if (drawingPanel.isDragging) {

drawingPanel.endX = e.getX();

drawingPanel.endY = e.getY();

switch (drawingPanel.operationType) {

case DRAW:

adjustShape(drawingPanel.currentShape, drawingPanel.startX, drawingPanel.startY, drawingPanel.endX, drawingPanel.endY);

break;

case MOVE:

if (drawingPanel.selectedShape != null) {

drawingPanel.selectedShape.x = e.getX() - drawingPanel.dx;

drawingPanel.selectedShape.y = e.getY() - drawingPanel.dy;

}

break;

case RESIZE:

if (drawingPanel.selectedShape != null) {

drawingPanel.selectedShape.width = e.getX() - drawingPanel.dx - drawingPanel.selectedShape.x;

drawingPanel.selectedShape.height = e.getY() - drawingPanel.dy - drawingPanel.selectedShape.y;

}

break;

}

statusLabel.setText(String.format("起点：(%d,%d)，终点：(%d,%d)", drawingPanel.startX, drawingPanel.startY, e.getX(), e.getY()));

drawingPanel.repaint();

}

}

@Override

public void mouseReleased(MouseEvent e) {

if (e.getButton() == MouseEvent.BUTTON1) {

drawingPanel.isDragging = false;

switch (drawingPanel.operationType) {

case DRAW:

drawingPanel.shapes.add(drawingPanel.currentShape);

drawingPanel.currentShape = null;

break;

case MOVE:

case RESIZE:

drawingPanel.selectedShape = null;

break;

}

statusLabel.setText("");

}

}

private Shape createShape(ShapeType type, int x, int y, int w, int h) {

switch (type) {

case LINE:

return new Line2D.Double(x, y, x, y);

case RECTANGLE:

return new Rectangle(x, y, w, h);

case OVAL:

return new Ellipse2D.Double(x, y, w, h);

case TEXT:

return new TextShape(inputArea.getText(), x, y, fgBox.getSelectedIndex());

default:

throw new IllegalArgumentException("Unsupported shape type: " + type);

}

}

private void adjustShape(Shape shape, int startX, int startY, int endX, int endY) {

if (shape instanceof Line2D) {

((Line2D) shape).setLine(startX, startY, endX, endY);

} else if (shape instanceof Rectangle) {

int x = Math.min(startX, endX);

int y = Math.min(startY, endY);

int w = Math.abs(endX - startX);

int h = Math.abs(endY - startY);

((Rectangle) shape).setBounds(x, y, w, h);

} else if (shape instanceof Ellipse2D) {

int x = Math.min(startX, endX);

int y = Math.min(startY, endY);

int w = Math.abs(endX - startX);

int h = Math.abs(endY - startY);

((Ellipse2D) shape).setFrame(x, y, w, h);

}

}

}

菜单栏、文件对话框等组件，可以打开和保存图形文件。同时，还可以通过菜单栏设置当前操作类型。

private void onOpenMenuItemClicked() {

try (ObjectInputStream in = new ObjectInputStream(new FileInputStream("shapes.dat"))) {

ArrayList<Shape> shapes = (ArrayList<Shape>) in.readObject();

drawingPanel.shapes.clear();

drawingPanel.shapes.addAll(shapes);

drawingPanel.repaint();

statusLabel.setText("打开文件成功！");

} catch (FileNotFoundException e) {

statusLabel.setText("文件不存在！");

} catch (IOException | ClassNotFoundException e) {

statusLabel.setText("打开文件出错！");

}

}

private void onSaveMenuItemClicked() {

try (ObjectOutputStream out = new ObjectOutputStream(new FileOutputStream("shapes.dat"))) {

out.writeObject(drawingPanel.shapes);

statusLabel.setText("保存文件成功！");

} catch (IOException e) {

statusLabel.setText("保存文件出错！");

}

}

private void onDrawMenuItemClicked() {

drawingPanel.operationType = OperationType.DRAW;

statusLabel.setText("切换到绘制模式！");

}

private void onMoveMenuItemClicked() {

drawingPanel.operationType = OperationType.MOVE;

statusLabel.setText("切换到移动模式！");

}

private void onResizeMenuItemClicked() {

drawingPanel.operationType = OperationType.RESIZE;

statusLabel.setText("切换到调整大小模式！");

}

private void onExitMenuItemClicked() {

dispose();

System.exit(0);

}