

浙江大学

本科实验报告

课程名称:	Java 应用技术
实验名称:	Mini CAD
姓 名:	赵天辞
学 院:	计算机学院
系:	软件工程
专 业:	软件工程
学 号:	3180105160

2020 年 10 月 29 日

浙江大学实验报告

实验名称: Mini CAD 实验类型: 编程实验

一、 实验目的

学习使用 Java 原生 GUI 库开发桌面应用程序, 掌握 Java GUI 的事件触发和处理机制
掌握 Java 的对象序列化机制, 实现对象的保存和读取

二、 实验内容

开发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

三、 开发环境

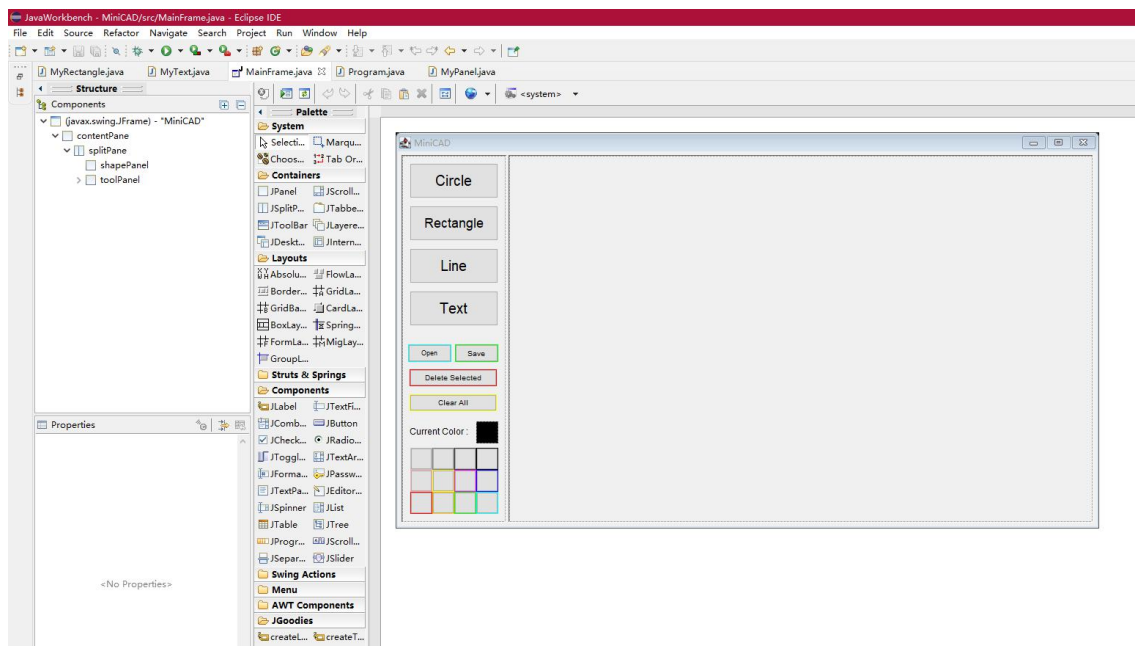
Java-SE 15, Eclipse, Visual Studio Code

四、 实验原理

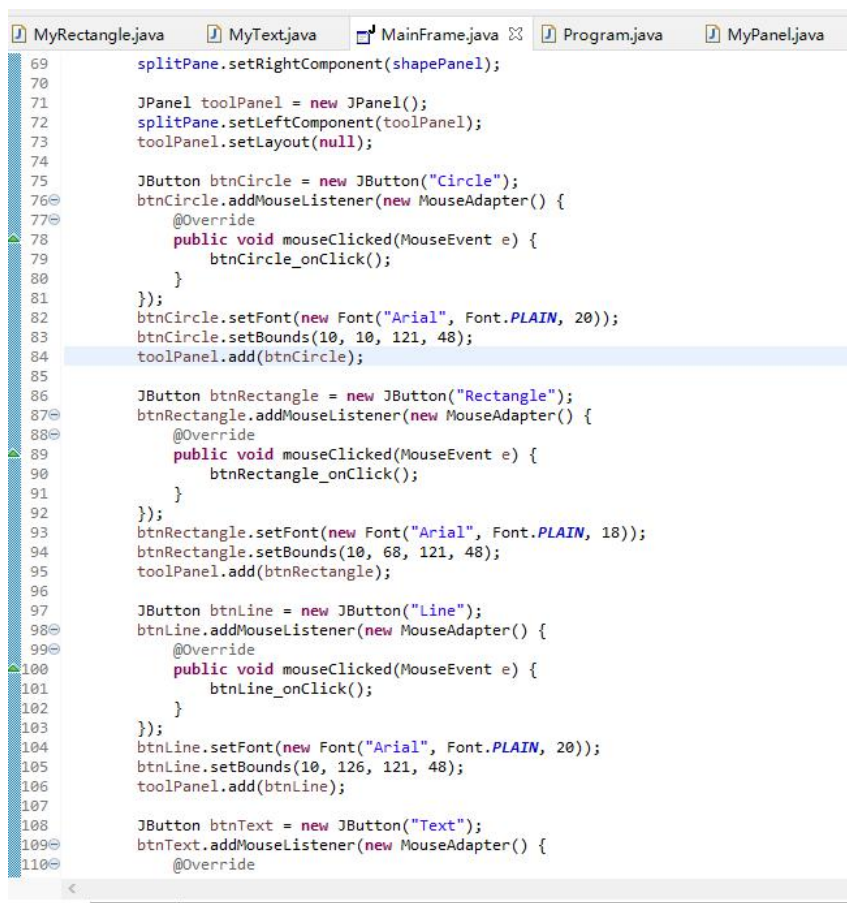
4.1 主窗口 MainFrame 的建立

4.1.1 使用 WindowBuilder 插件可视化设计并建立主窗口 GUI

本程序使用 WindowBuilder 插件来进行窗口的可视化设计和相关代码的自动生成, 使用 WindowBuilder 如下图所示



使用该插件可快速地建立所需的窗口与控件，快速调整布局，并且该插件会自动生成相应代码；WindowBuilder 生成的代码如下图所示：



```
69      splitPane.setRightComponent(shapePanel);
70
71      JPanel toolPanel = new JPanel();
72      splitPane.setLeftComponent(toolPanel);
73      toolPanel.setLayout(null);
74
75      JButton btnCircle = new JButton("Circle");
76      btnCircle.addMouseListener(new MouseAdapter() {
77          @Override
78          public void mouseClicked(MouseEvent e) {
79              btnCircle_onClick();
80          }
81      });
82      btnCircle.setFont(new Font("Arial", Font.PLAIN, 20));
83      btnCircle.setBounds(10, 10, 121, 48);
84      toolPanel.add(btnCircle);
85
86      JButton btnRectangle = new JButton("Rectangle");
87      btnRectangle.addMouseListener(new MouseAdapter() {
88          @Override
89          public void mouseClicked(MouseEvent e) {
90              btnRectangle_onClick();
91          }
92      });
93      btnRectangle.setFont(new Font("Arial", Font.PLAIN, 18));
94      btnRectangle.setBounds(10, 68, 121, 48);
95      toolPanel.add(btnRectangle);
96
97      JButton btnLine = new JButton("Line");
98      btnLine.addMouseListener(new MouseAdapter() {
99          @Override
100         public void mouseClicked(MouseEvent e) {
101             btnLine_onClick();
102         }
103     });
104     btnLine.setFont(new Font("Arial", Font.PLAIN, 20));
105     btnLine.setBounds(10, 126, 121, 48);
106     toolPanel.add(btnLine);
107
108     JButton btnText = new JButton("Text");
109     btnText.addMouseListener(new MouseAdapter() {
110         @Override
```

4.1.2 绑定控件事件

为所有的控件按钮添加匿名类对象作为鼠标事件监听器，并绑定相应的点击处理函数，如下图所示：

```
btnCircle.addMouseListener(new MouseAdapter() {
    @Override
    public void mouseClicked(MouseEvent e) {
        btnCircle_onClick();
    }
});
```

4.1.3 窗口的生成

在主函数中实例化 MainFrame 类对象并设置其为可见以创建主窗口，代码如下：

```

public class Program {
    public static void main(String[] args) {
        EventQueue.invokeLater(new Runnable() {
            public void run() {
                try {
                    var frame = new MainFrame();
                    frame.setVisible(true);
                } catch (Exception e) {
                    e.printStackTrace();
                }
            }
        });
    }
}

```

4.2 自定义图形控件

4.2.1 图形类基类 MyShape

(1) 概述

抽象类 MyShape 是自定义图形控件基类，继承自 JComponent，封装了图形控件对鼠标事件的处理和自身图形的绘制；

MyShape 类定义了以下属性：

```

private Color color;           当前图形的颜色
private boolean isSelected;    是否被选中
private boolean isPressed;     鼠标是否在图形上被按下
private boolean isInside;      鼠标光标是否在图形内
private boolean isDragging;    是否被拖动

```

MyShape 类定义了两个抽象方法：

```

protected abstract Shape getShape();
protected abstract boolean checkMousePos(MouseEvent e);

```

getShape() 方法返回当前图形的形状（仅边界）

checkMousePos() 方法用来判定当前鼠标光标位置是否在图形内部

所有图形控件子类必须实现这两个抽象方法

(2) 图形绘制

覆写了 paintComponent(Graphics g) 方法，进行该控件图形的绘制，过程如下：

首先调用 getShape() 方法获取外形，根据当前状态设置画笔颜色，绘制；然后进行填充

(3) 鼠标移动事件

定义了 onMouseMoved(MouseEvent e) 方法，处理鼠标移动事件，过程如下：

调用 `checkMousePos()` 方法检查鼠标光标位置是否在图形内；若在图形内，更改属性 `isInside` 为 `true`，否则更改为 `false`

(4) 鼠标按下事件

定义了 `onMousePressed(MouseEvent e)` 方法，处理鼠标按下事件，过程如下：

判断鼠标按键，若为鼠标左键，则继续：

判断光标是否在图形内，若是，则继续：

更改 `isPressed` 属性为 `true`

(5) 鼠标松开事件（图形选中）

定义了 `onMouseReleased(MouseEvent e)` 方法，处理鼠标松开事件，过程如下：

判断鼠标按键，若为鼠标左键，则继续：

判断 `isPressed` 属性是否为 `true`，若是，则继续：

若光标在图形内且不在被拖动，该图形被点击

设置 `isDragging` 为 `false`

设置 `isPressed` 为 `false`

(6) 鼠标拖动事件（图形拖动）

定义了 `onMouseDragged(MouseEvent e)` 方法，处理鼠标拖动事件，过程如下：

若当前图形被选中且被按下，则继续：

设置 `isDragging` 为 `true`

移动至光标当前位置

(7) 鼠标滚轮事件（图形缩放）

定义了 `onMouseWheelMoved(MouseWheelEvent e)` 方法，处理鼠标滚轮事件，过程如下：

若当前图形被选中：

根据滚轮滚动方向对图形进行放大、缩小

(8) 颜色更改

定义了共有方法 `setColor(Color color)`，调用以更改图形颜色，过程如下：

更改当前 `color` 属性，重绘

4.2.2 矩形 `MyRectangle`

(1) 概述

MyRectangle 类为矩形控件类，继承自 MyShape

(2) 构造函数

定义构造函数：

```
public MyRectangle(int x, int y, Color color, int width, int height)
```

x、y 为矩形左上角坐标，color 为颜色，width、height 为长宽

(3) 图形绘制

实现父类的 getShape() 方法，返回对应矩形

(4) 光标检测

实现父类的 checkMousePos() 方法，进行矩形的坐标检查

4.2.3 圆形 MyCircle

(1) 概述

MyCircle 类为圆形控件类，继承自 MyShape

(3) 构造函数

定义构造函数：

```
public MyCircle(Point origin, Color color, int radius)
```

origin 为原点坐标，color 为颜色，radius 为半径

(3) 图形绘制

实现父类的 getShape() 方法，返回对应圆形

(4) 光标检测

实现父类的 checkMousePos() 方法，进行圆形的坐标检查

4.2.4 直线 MyLine

(1) 概述

MyLine 类为直线控件类，继承自 MyShape

MyLine 类定义了以下属性：

```
private float lineWidth;    线宽
```

(4) 构造函数

定义构造函数：

```
public MyLine(Point startPoint, Point endPoint, Color color)
```

startPoint 为起始坐标，endPoint 为终点坐标，color 为颜色

(3) 图形绘制

实现父类的 getShape() 方法，返回对应线段

(4) 光标检测

实现父类的 checkMousePos() 方法，进行线段的坐标检查

(5) 覆写鼠标点击事件

覆写了父类的 onMouseClicked() 方法，过程如下：

调用父类的 onMouseClicked() 方法

若按键为鼠标中键：

增加线宽

若按键为鼠标右键：

减少线宽

4.2.5 文本 MyText

(1) 概述

MyText 类为文本控件类，继承自 MyShape

MyText 类定义了以下属性：

```
private Font currentFont; 当前字体
```

```
private String content; 文本内容
```

(6) 构造函数

定义构造函数：

```
public MyText(Point origin, String content, Font font, Color color)
```

origin 为左上角坐标，content 为文本内容，font 为字体，color 为颜色

(3) 图形绘制

实现父类的 getShape() 方法，返回对应文本

(4) 光标检测

实现父类的 checkMousePos() 方法，同矩形

(7) 覆写鼠标滚轮事件

覆写了父类的 onMouseEvent() 方法，过程如下：

根据滚轮滚动方向（增加/减小）当前字体的字号

4.3 保存和恢复

利用 Java 的对象可序列化机制，可以非常容易地实现场景的保存和恢复，具体实现如下：

保存时，序列化图形控件容器 shapePanel（其包含所有创建的图形控件）并将其保存至文件；

恢复时，读取文件内容并将其反序列化得到保存的 shapePanel，将其替换现有的容器

五、 实验结果与使用说明

5.1 主窗口



左侧为工具栏区：

Circle, Rectangle, Line, Text 按钮对应相应的图形；

Open, Save 按钮为打开文件和保存；

Delete Selected 按钮删除选中的图形；

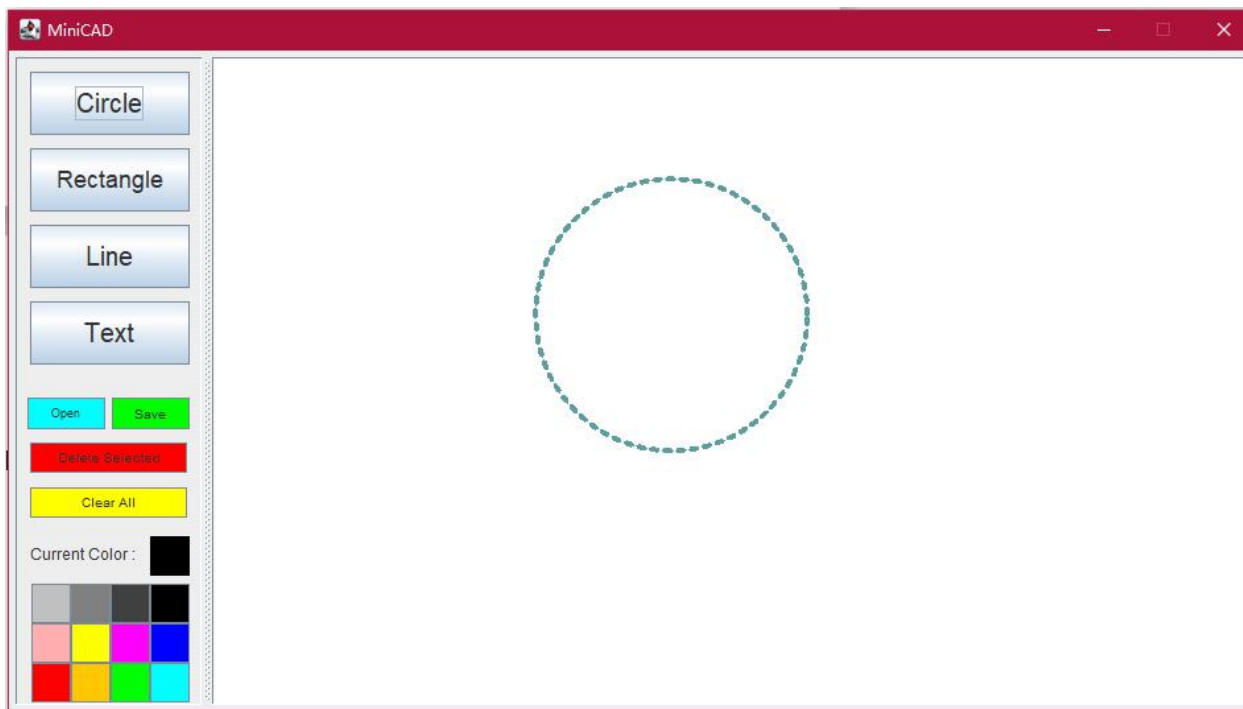
Clear All 按钮清空当前场景；

点击不同的颜色以更改当前颜色

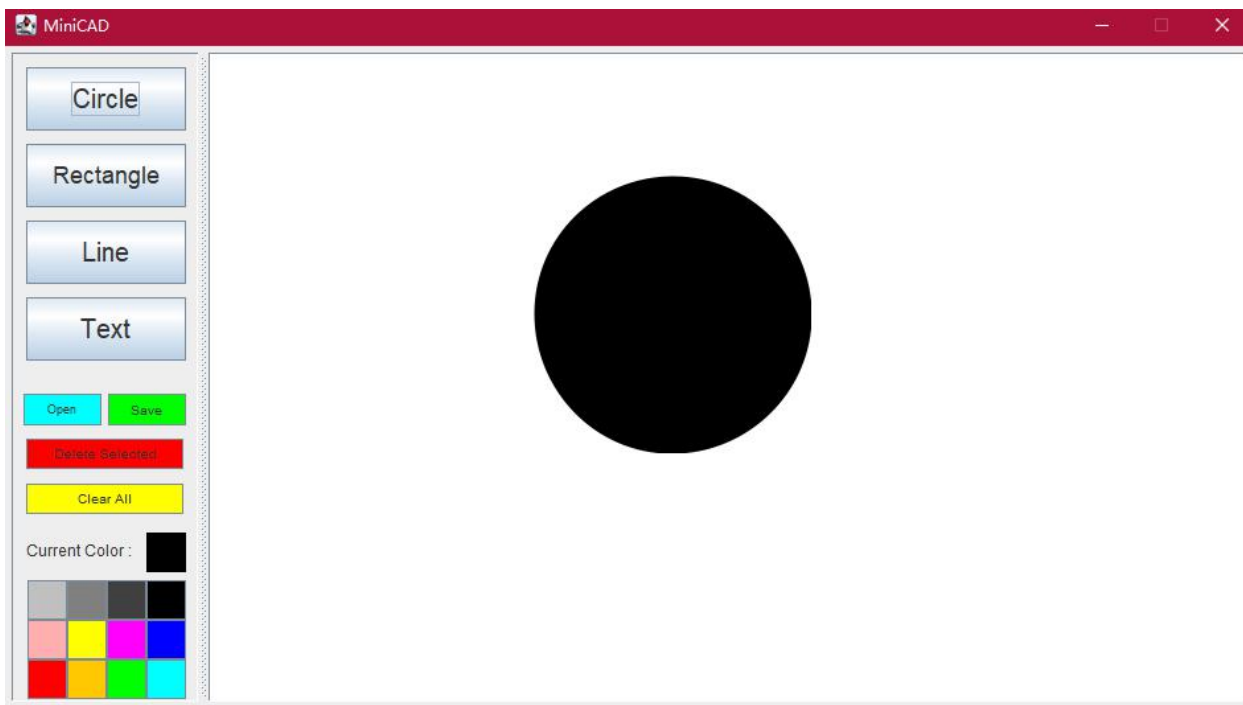
右侧为场景区（绘图区）

5.2 添加圆形

点击 Circle 按钮，点击绘图区，移动鼠标以预览添加的圆形

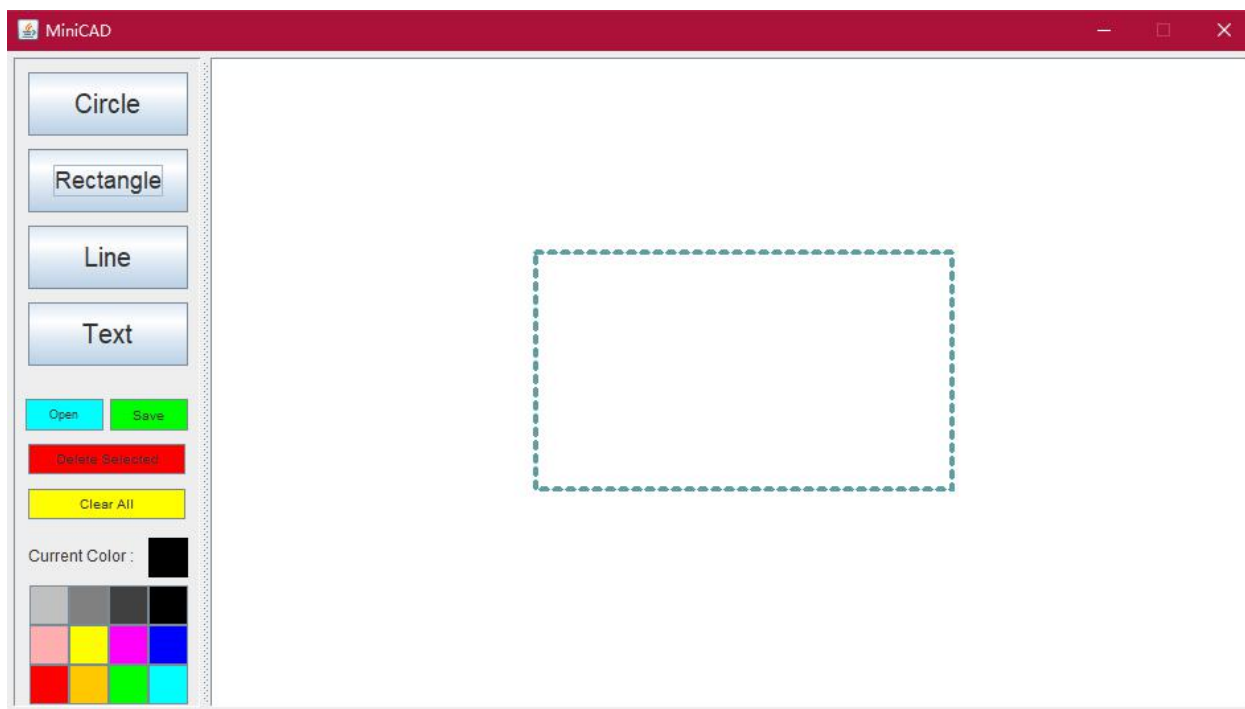


再次点击绘图区以确定，图形被添加



5.3 添加矩形

点击 Rectangle 按钮，点击绘图区，移动鼠标以预览添加的矩形

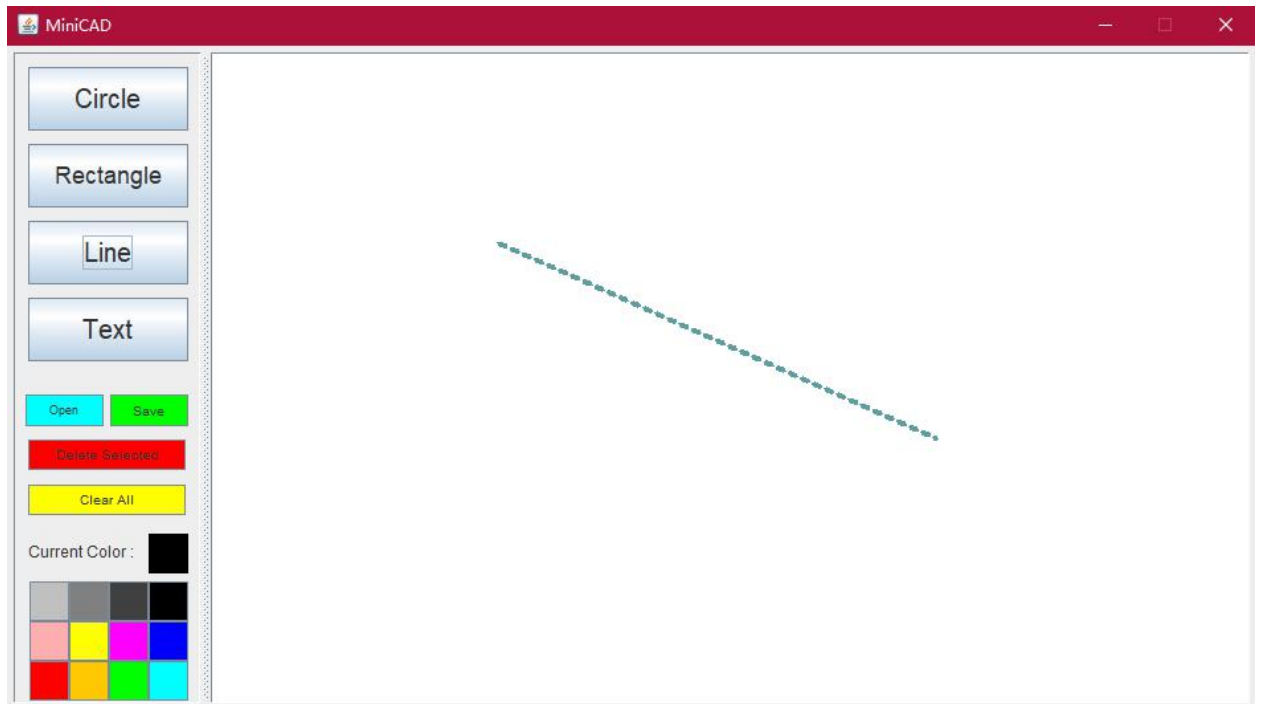


再次点击绘图区以确定，图形被添加

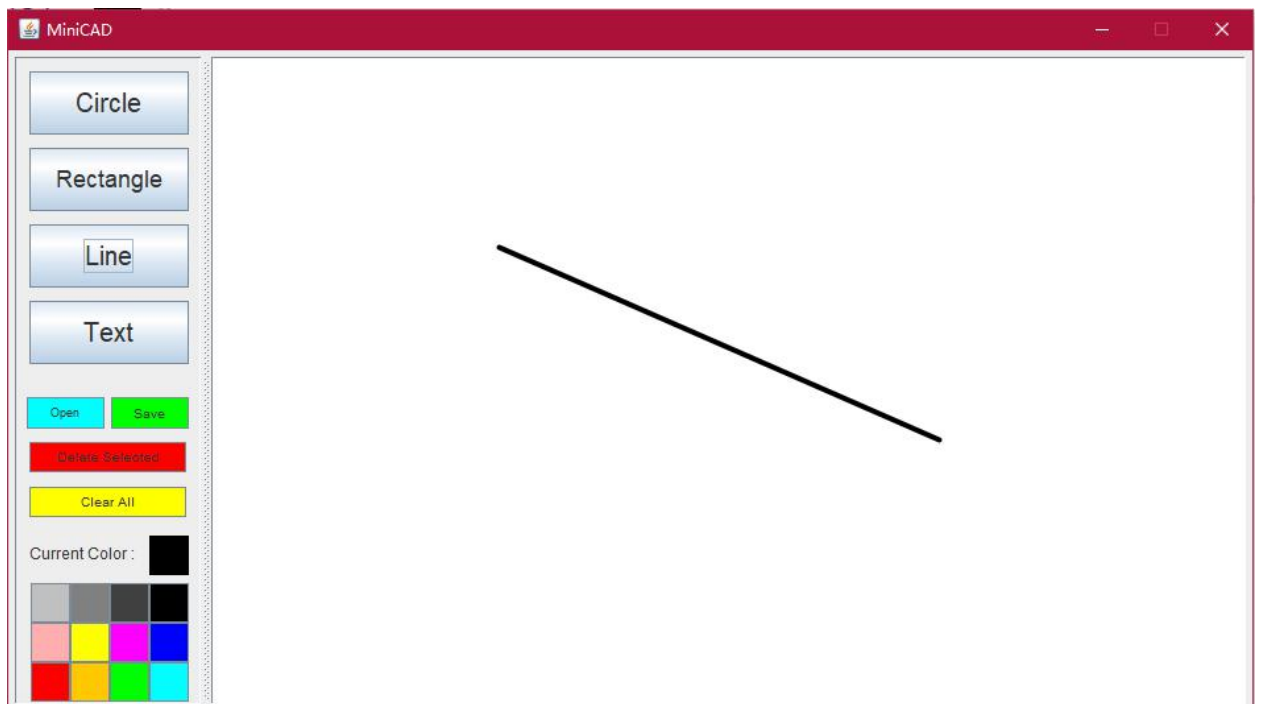


5.4 添加直线

点击 Line 按钮，点击绘图区，移动鼠标以预览添加的直线

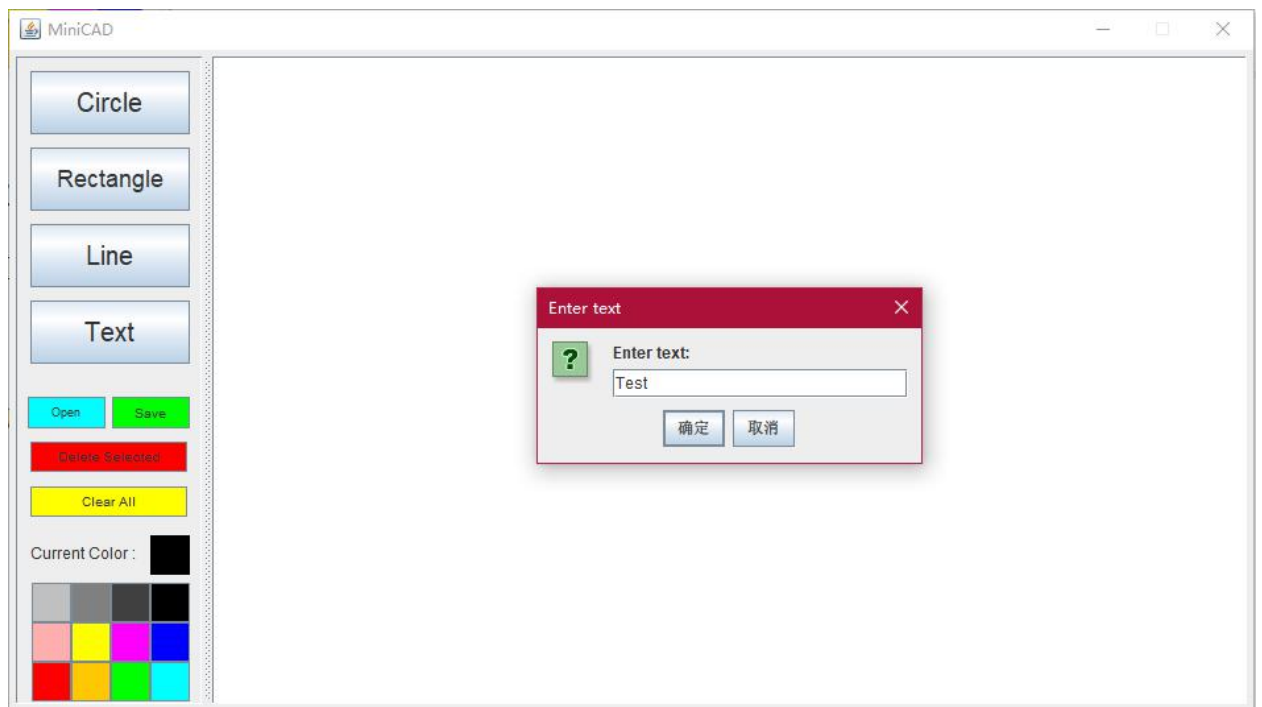


再次点击绘图区以确定，图形被添加

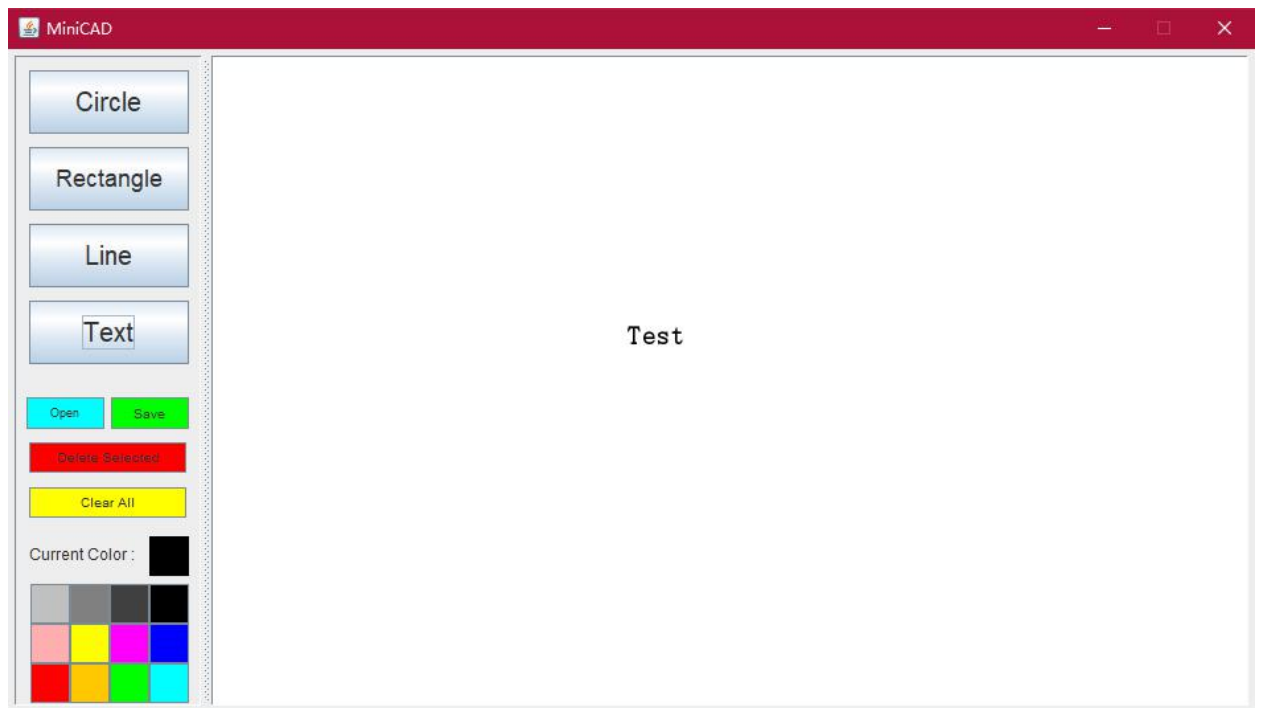


5.5 添加文本

点击 Text 按钮，点击绘图区，在弹出的对话框内输入要添加的文本内容

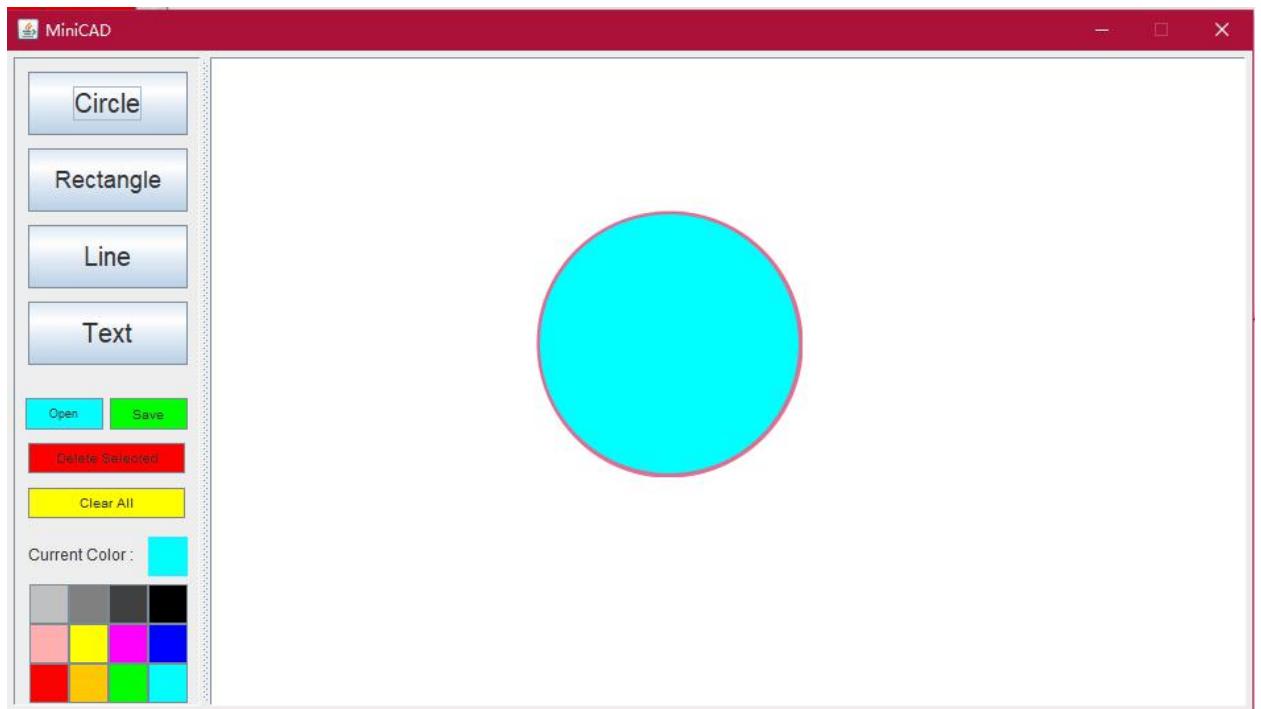


点击确定，图形被添加

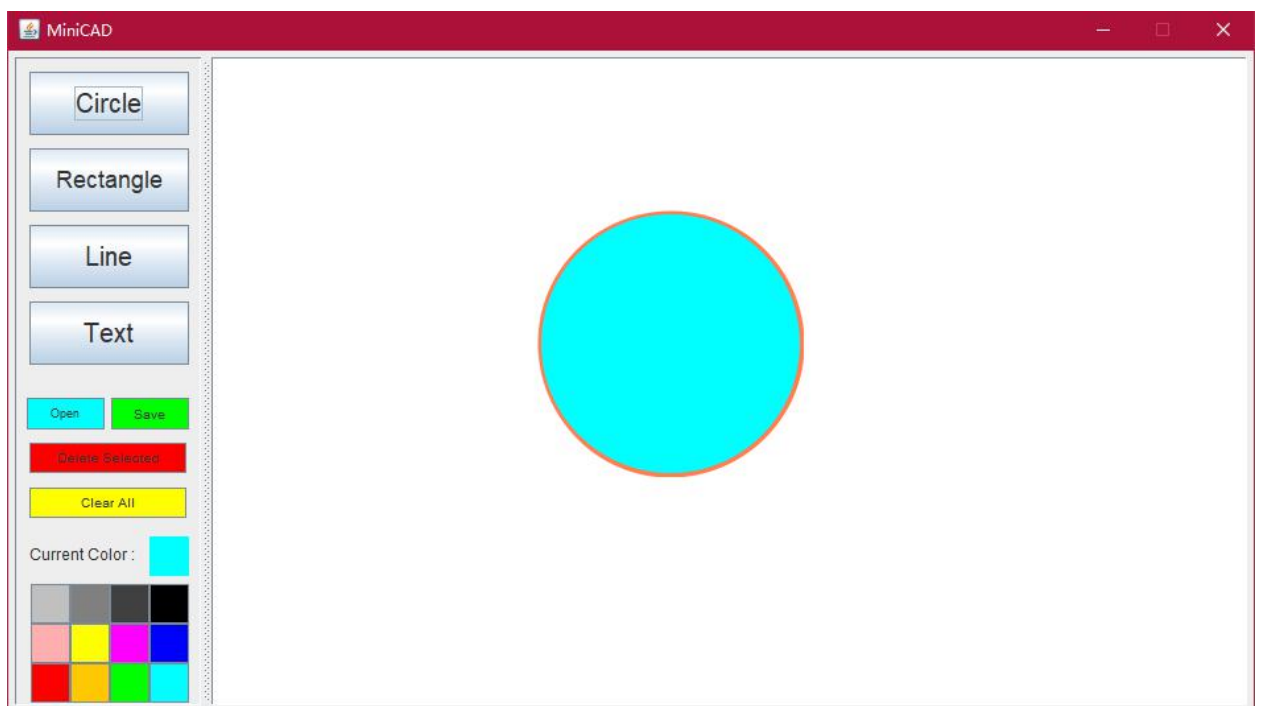


5.6 选中图形

鼠标光标靠近图形，图形边框会高亮

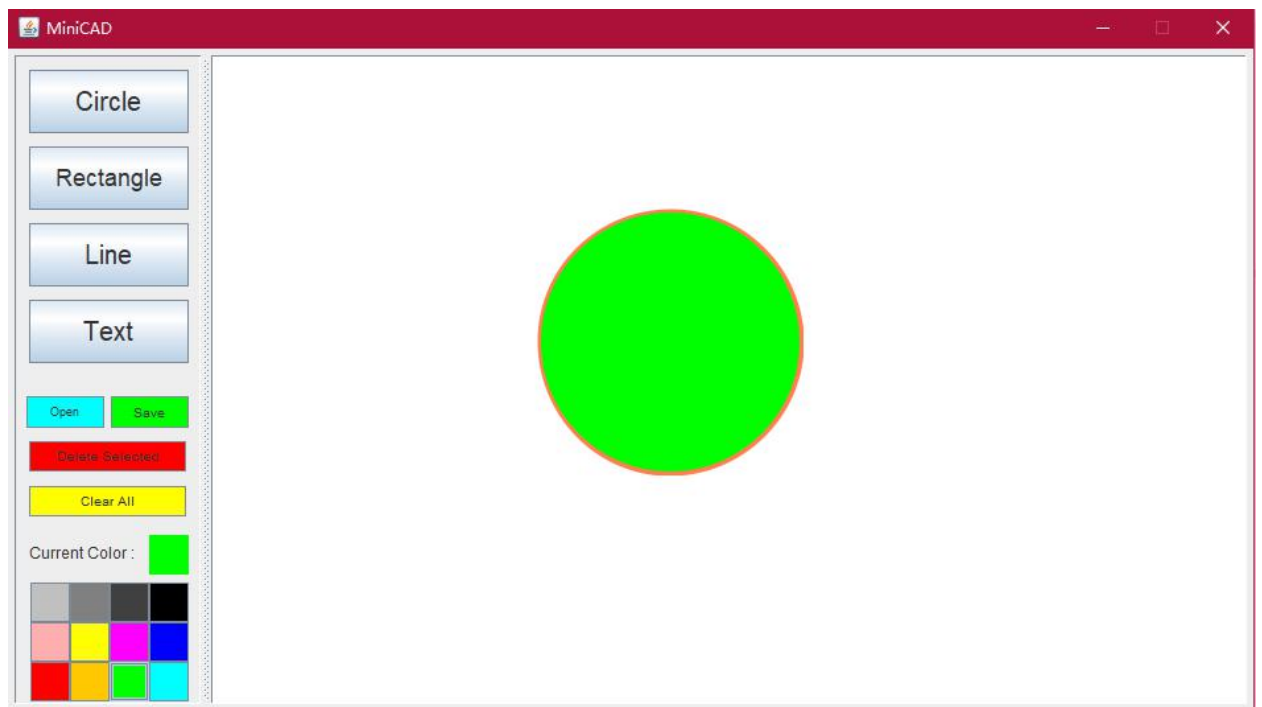


点击图形，图形被选中，边框颜色改变



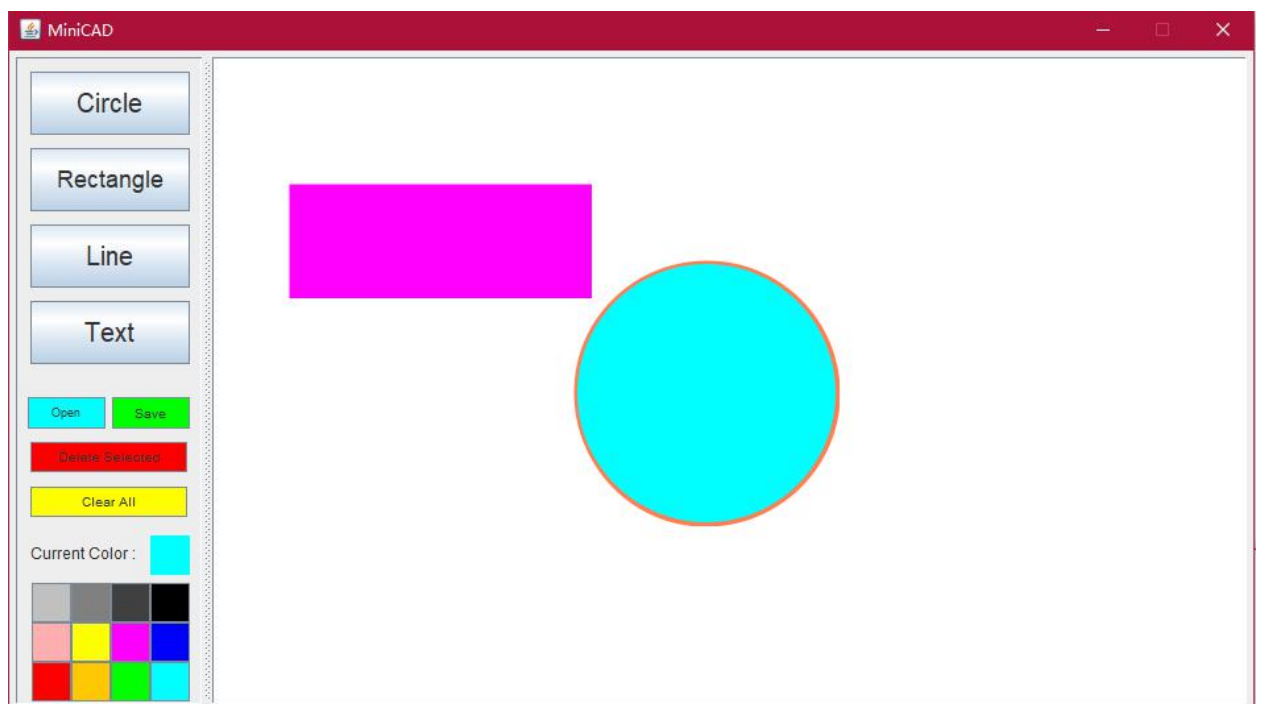
5.7 更改颜色

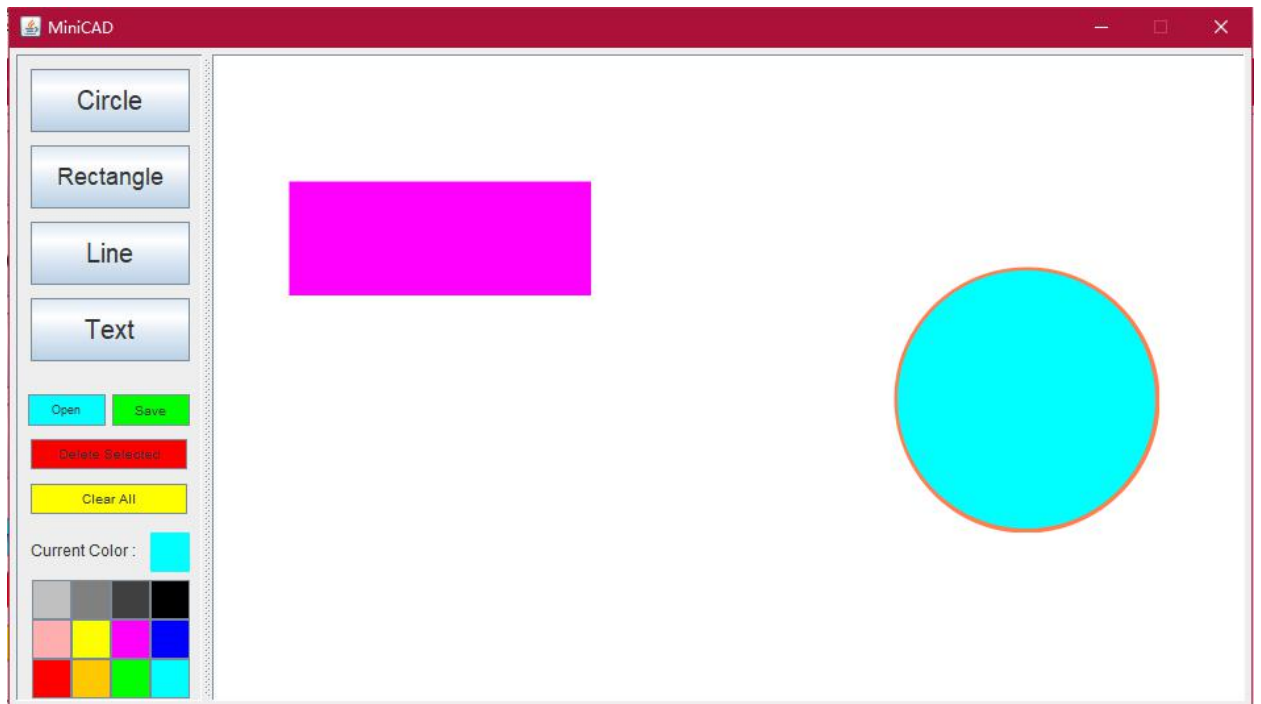
选中要更改颜色的图形，点击工具栏下方的图片，图形颜色更改



5.8 拖动

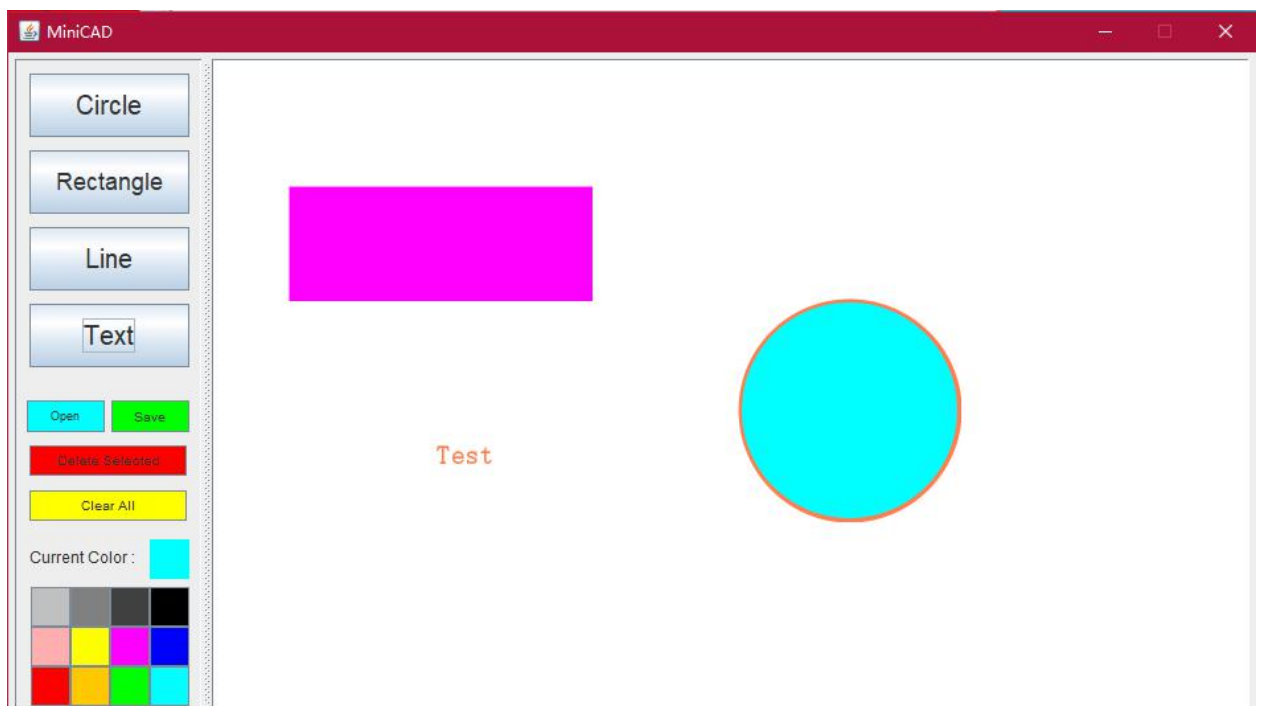
拖动选中的图形，图形被移动

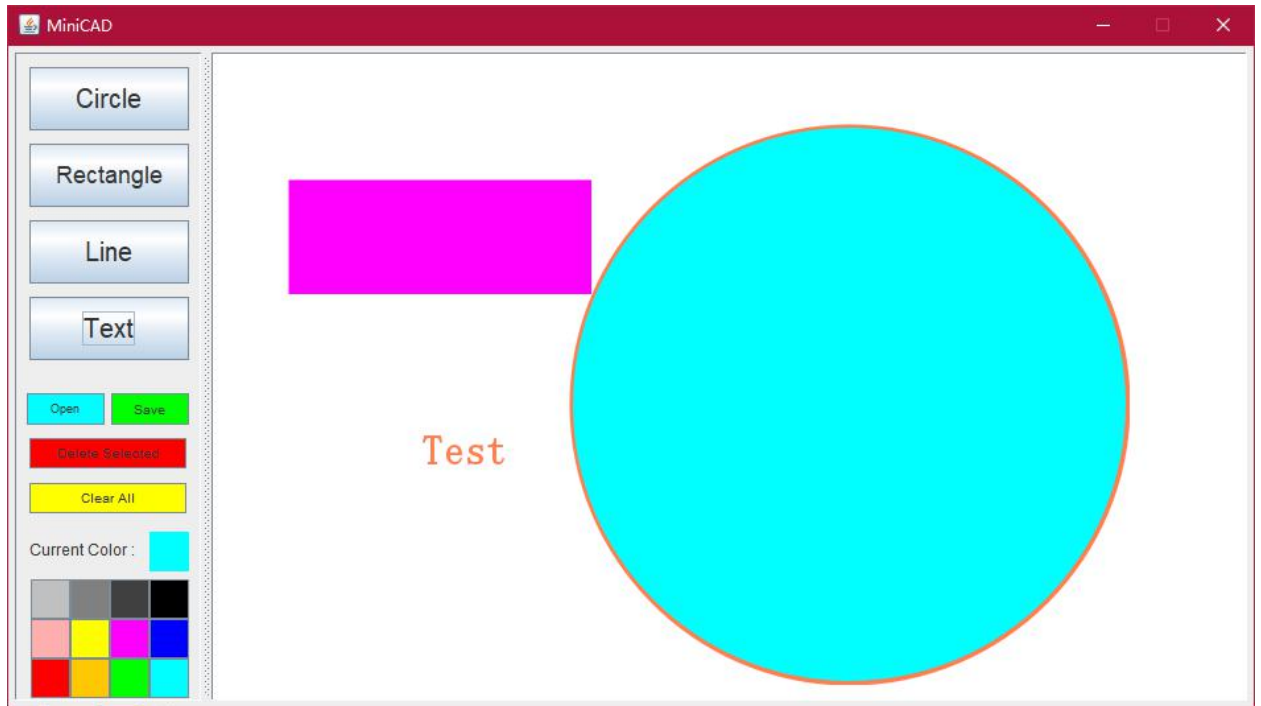




5.9 缩放

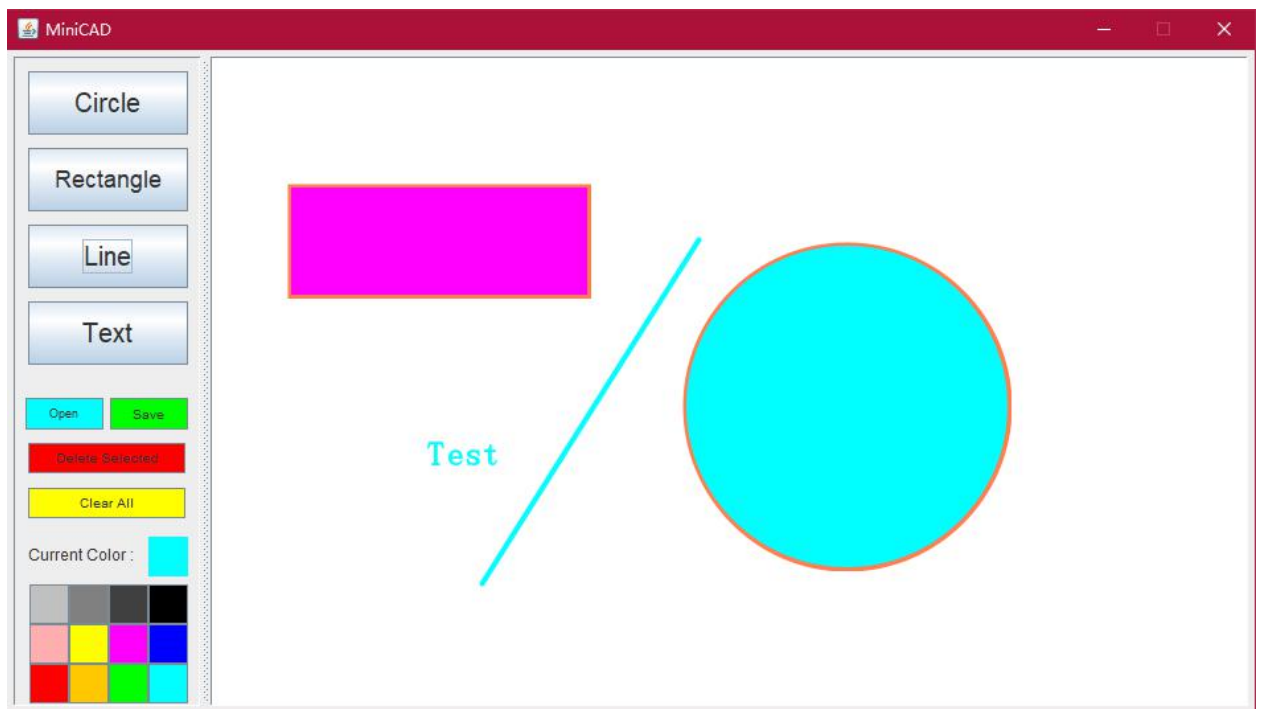
选中需要缩放的图形，鼠标滚轮向前放大，向后缩小

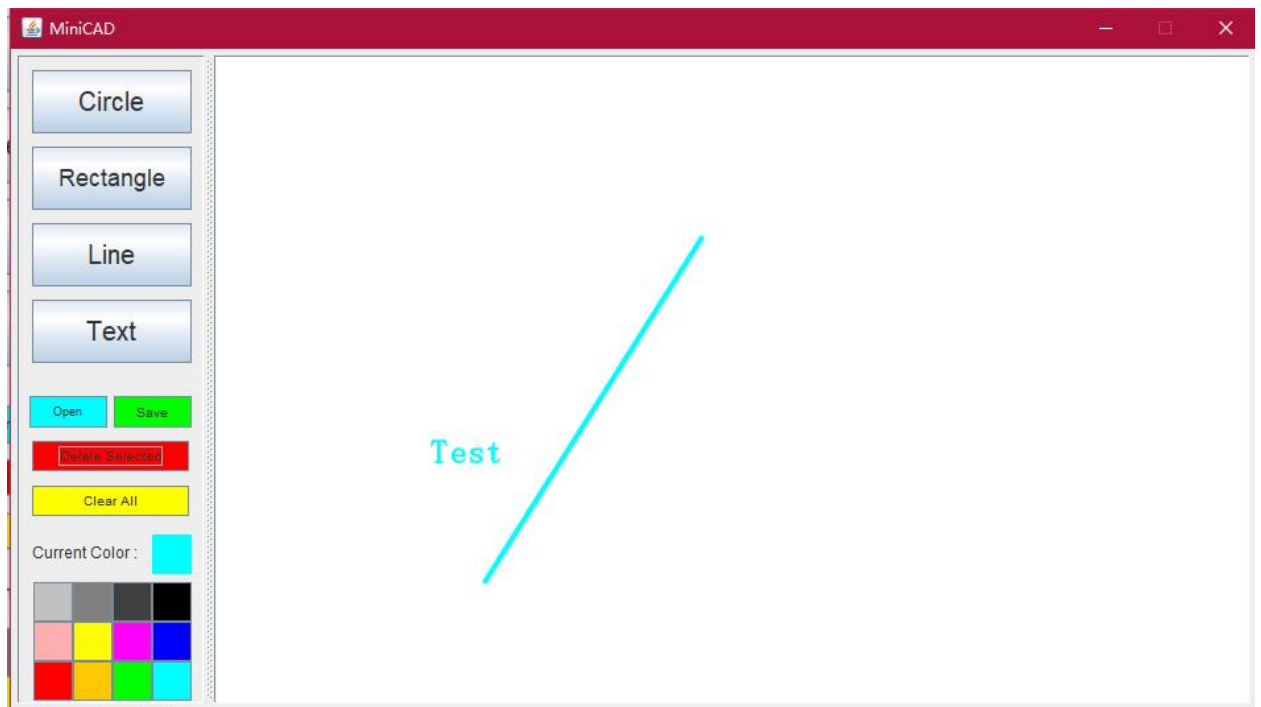




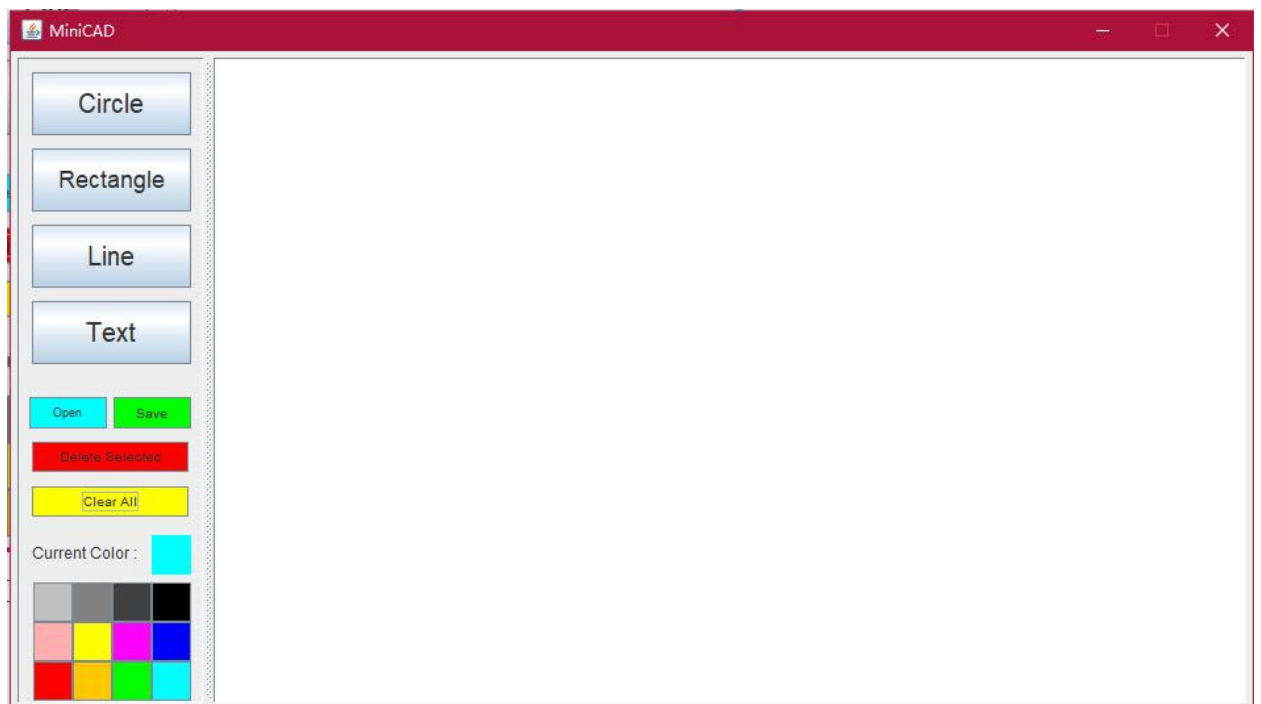
5.10 删除选中项与清空场景

点击工具栏按钮 Delete Selected 删除选中



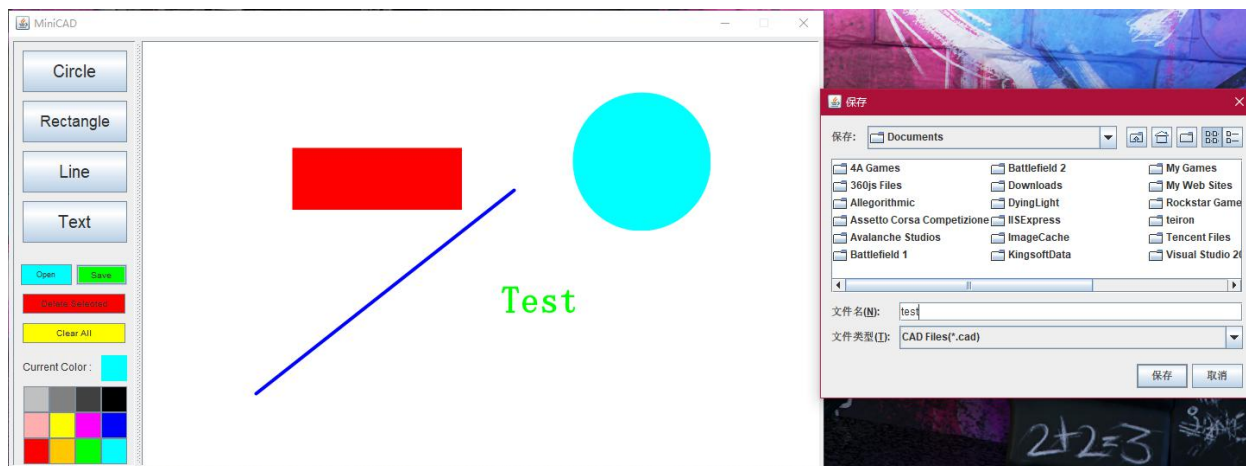


点击工具栏 Clear All 清空场景

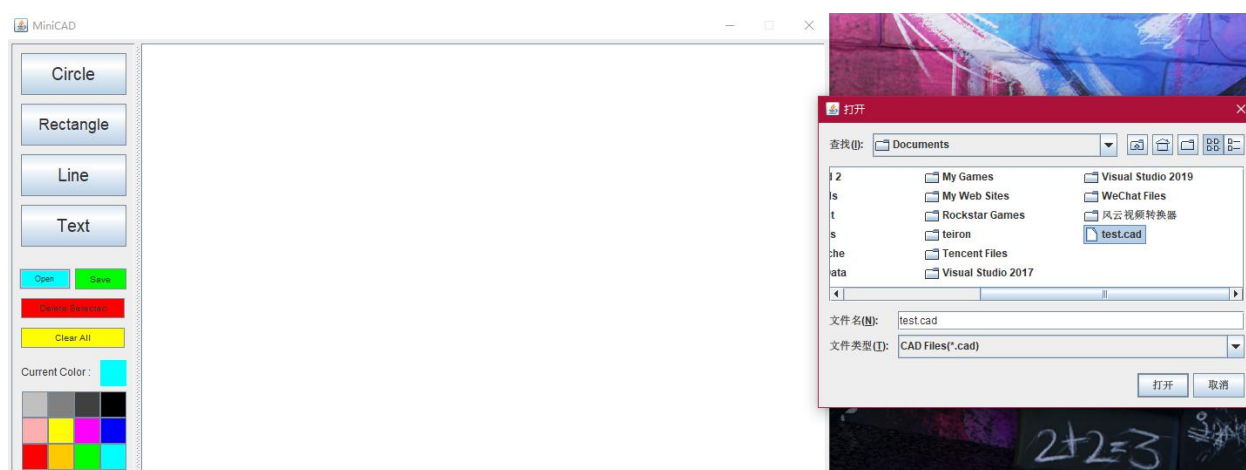


5.11 保存与恢复

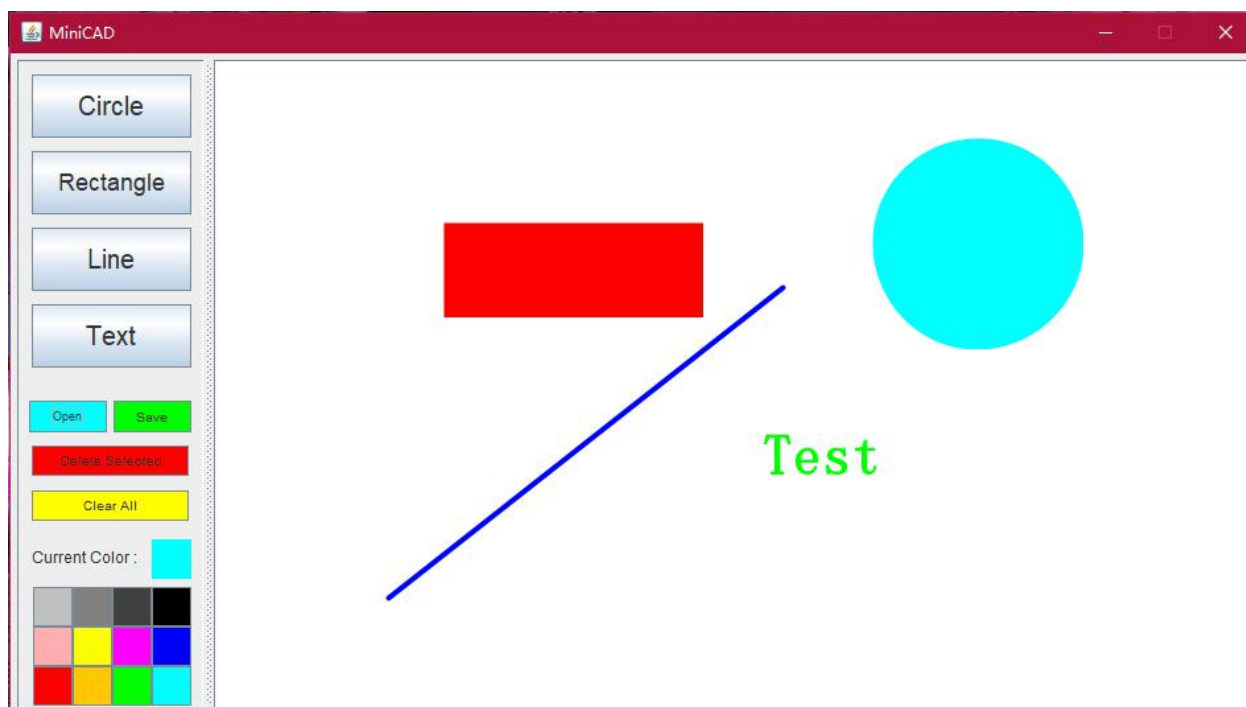
点击 Save 按钮，在弹出的对话框选择保存的位置与文件名，点击保存



点击 Open 按钮，在对话框中选择保存的*.cad 文件，点击打开



保存的场景被加载



六、 实验结果分析

程序实现了该实现要求的功能，并能正确运行

七、 实验心得

通过本次实验，我学习和使用了 Java 原生 GUI 库开发桌面应用程序，掌握了 Java GUI 的事件触发和处理机制，并且掌握 Java 的对象序列化机制，实现了对对象的保存和读取