实验报告

1. 问题说明

实现一个图形绘制软件，具有如下功能和非功能要求：

1.能用鼠标绘制：线段 矩形 椭圆 填充的矩形 填充的椭圆 多点折线 多边形和文字块；

2.可以用鼠标选中已经绘制的图形；

3.可以移动选中的图形；

4.可以修改选中的图形的颜色、大小、线条粗细和文字内容；

5.可以删除选中的图形；

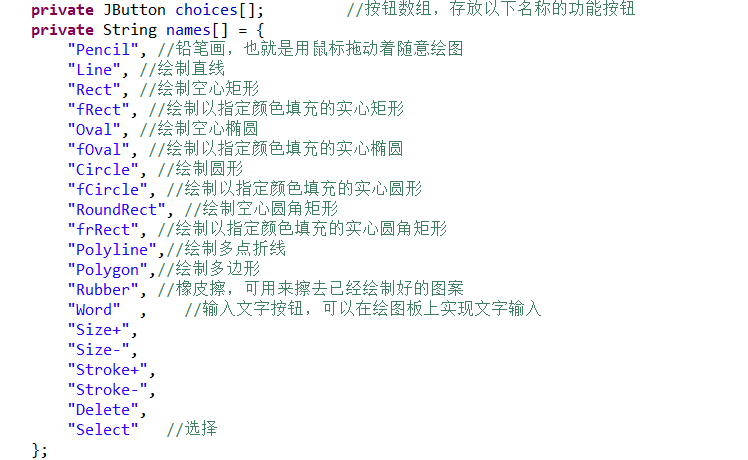
6.可以将所绘制的图形保存在文件中；

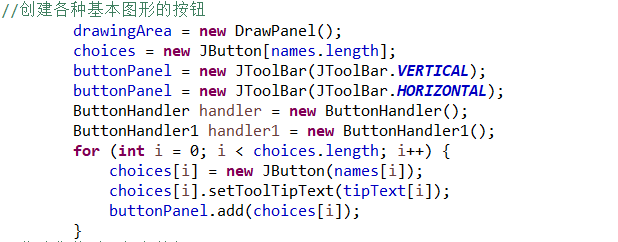
7.可以将保存的文件中的图形加载到当前的图形中；

1. 实现方式
2. 菜单栏：使用JMenuBar类制造了菜单栏对象，向其中添加了“file”、“change”、“help”三个菜单项，并在“file”中添加了“new”、“save”、“load”、“exit”功能；在“change”中添加了修改颜色、粗细、文字内容的功能；在“help”中加入了一些开发者信息。

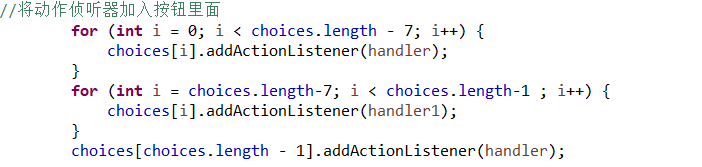


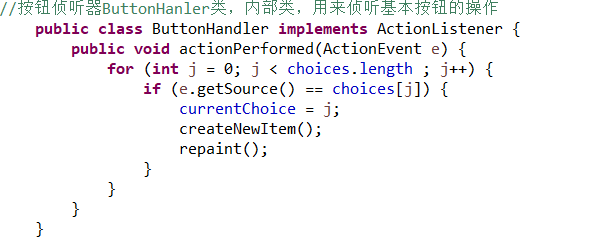
1. 工具栏：使用JToolBar类制造了工具栏对象，并在其中添加了所有可以使用的功能的按钮对象。





1. 在工具栏中的每一个对象上都添加了监听器，保证能够针对不同按钮的动作做出相应的状态变化。将所有的功能状态进行编号，并且设置currentchoice变量，用来记录当前的选择，从而对应之后的鼠标事件做出相应的反应。

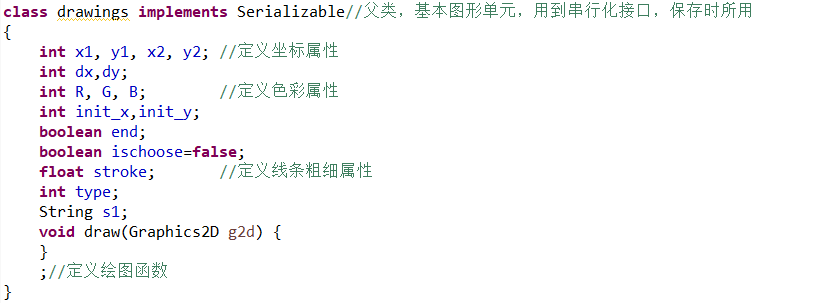


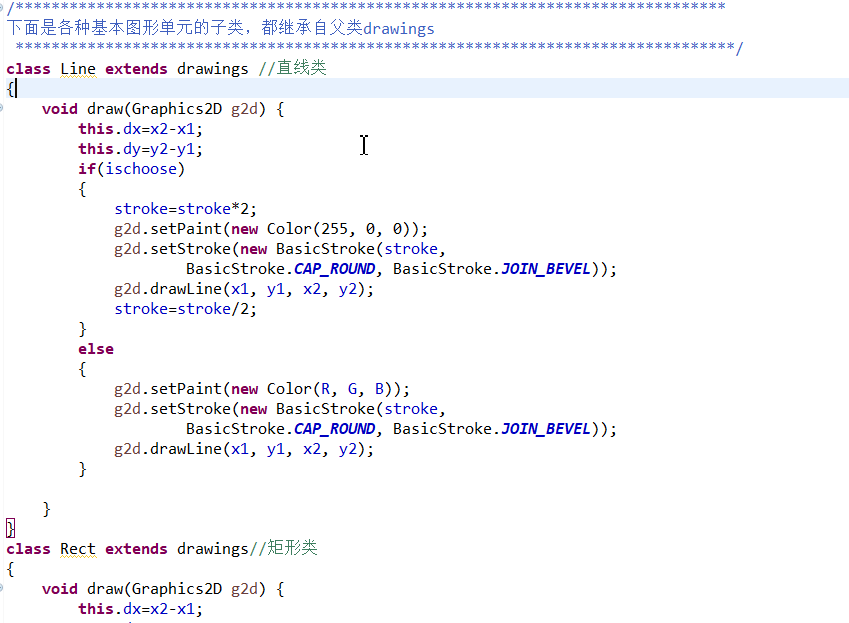


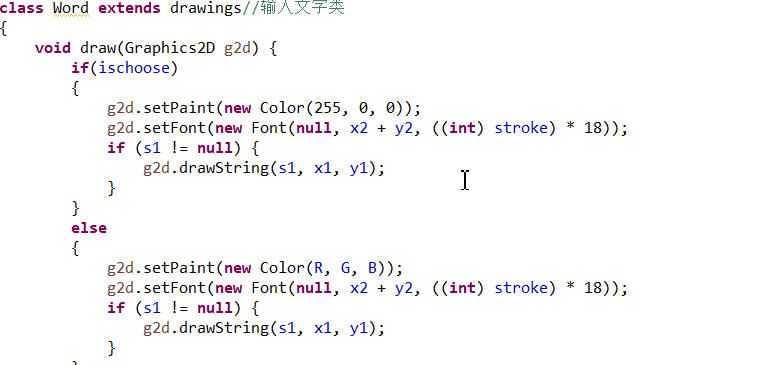
1. 鼠标在不同的状态下，相应的鼠标事件会触发不同的图形绘制过程，并且不断刷新画面使得绘制过程可以动态显示



1. 所有的图形都属于一个drawings公共父类下的一个子类，并且每一个图形子类都根据自己的相应情况实现了自己的draw（）函数，可以在调用时将这个图形绘制出来。

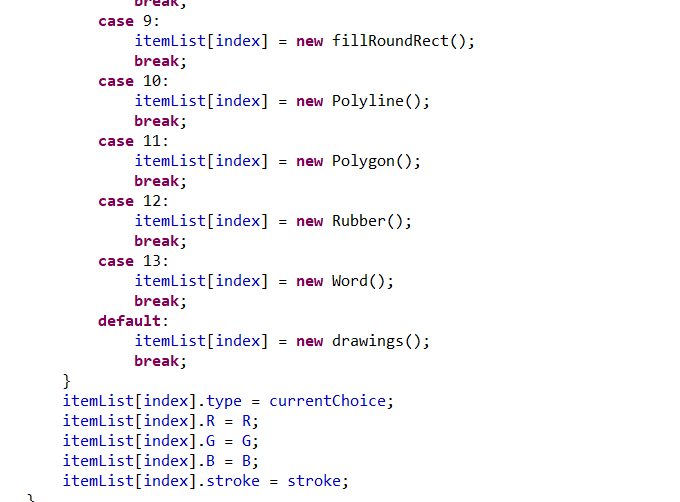




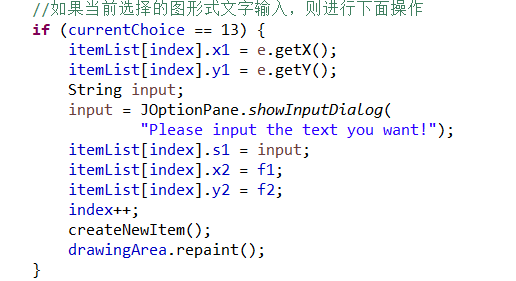


1. 定义了一个容器用于储存所有drawings类的对象，用以表示所有已经画在画布上的图形，并且之后对于图形属性进行改变时可以遍历容器找到相应的对象并对单个图形进行修改。并且每一次画面有图形变动后都遍历容器，调用所有对象的draw函数将所有图形进行一遍重新绘制。其中createNewItem函数用于根据当前不同的currentChoice制造相应不同的对象并加入到容器中。





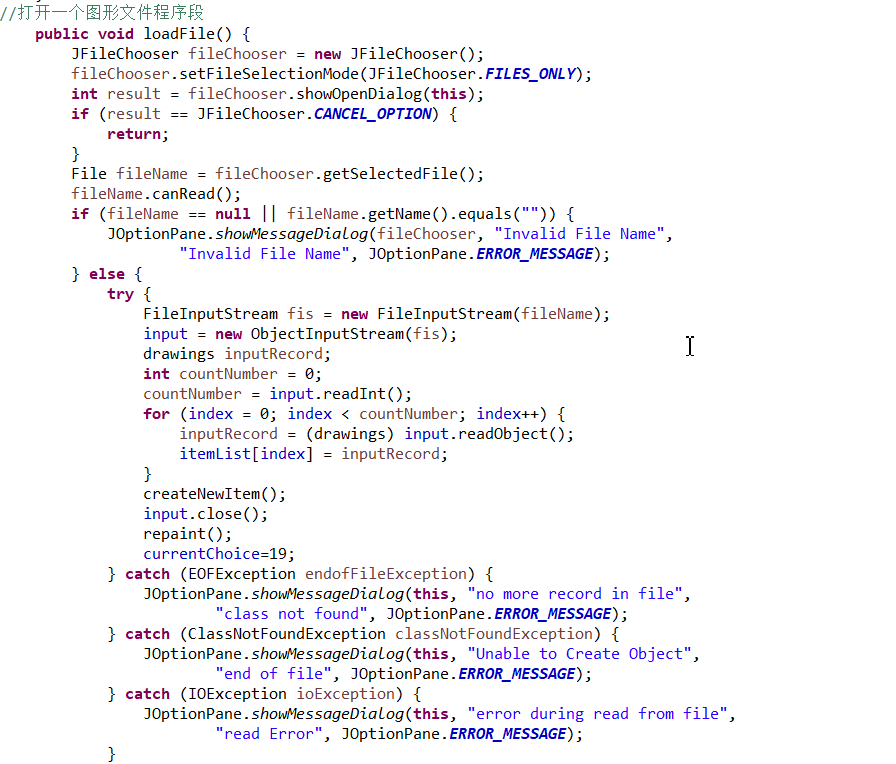
1. 添加文字：使用了java类库中的messagebox类，选择相应的按钮之后会弹出对应的对话框，将文字输入对话框中并点击确定即可。



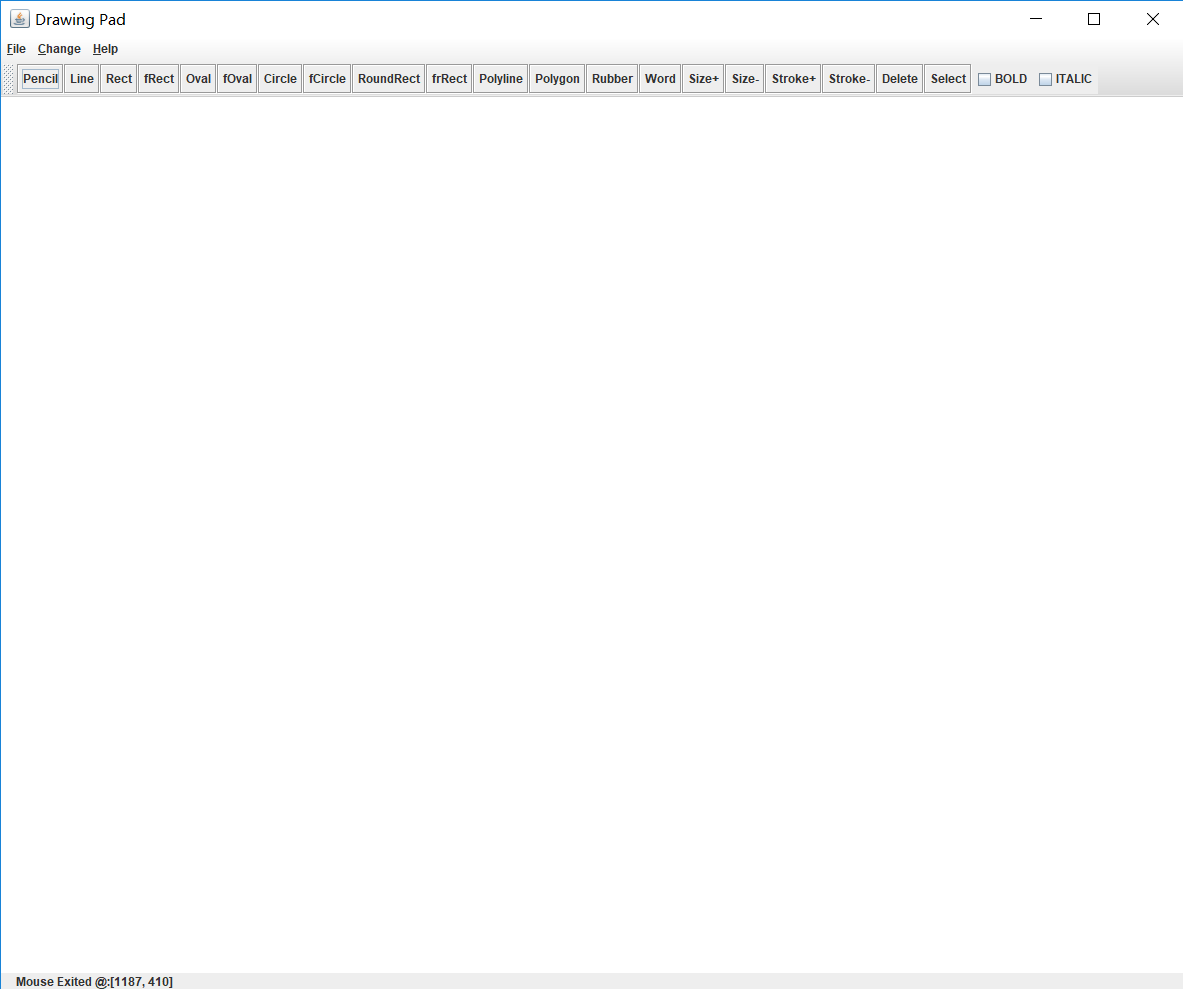
1. 保存文件：使用了对象串行化，将图像以及图形对象容器以比特流的形式保存在文件中。



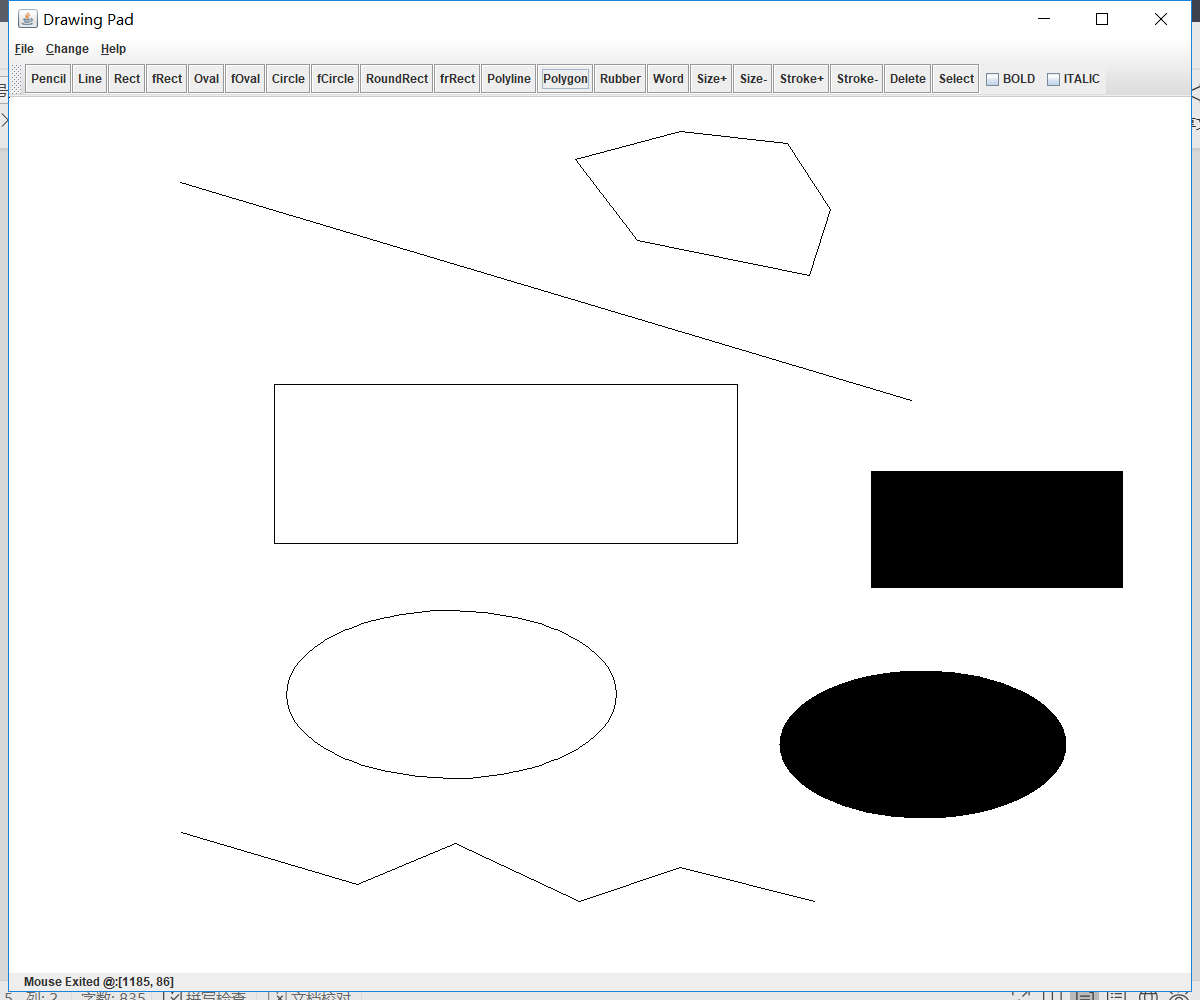
1. 读取文件：从文件中导入相应的数据流，并且重新构造图形对象数组，并使用repaint（）函数重新绘制整个画面。



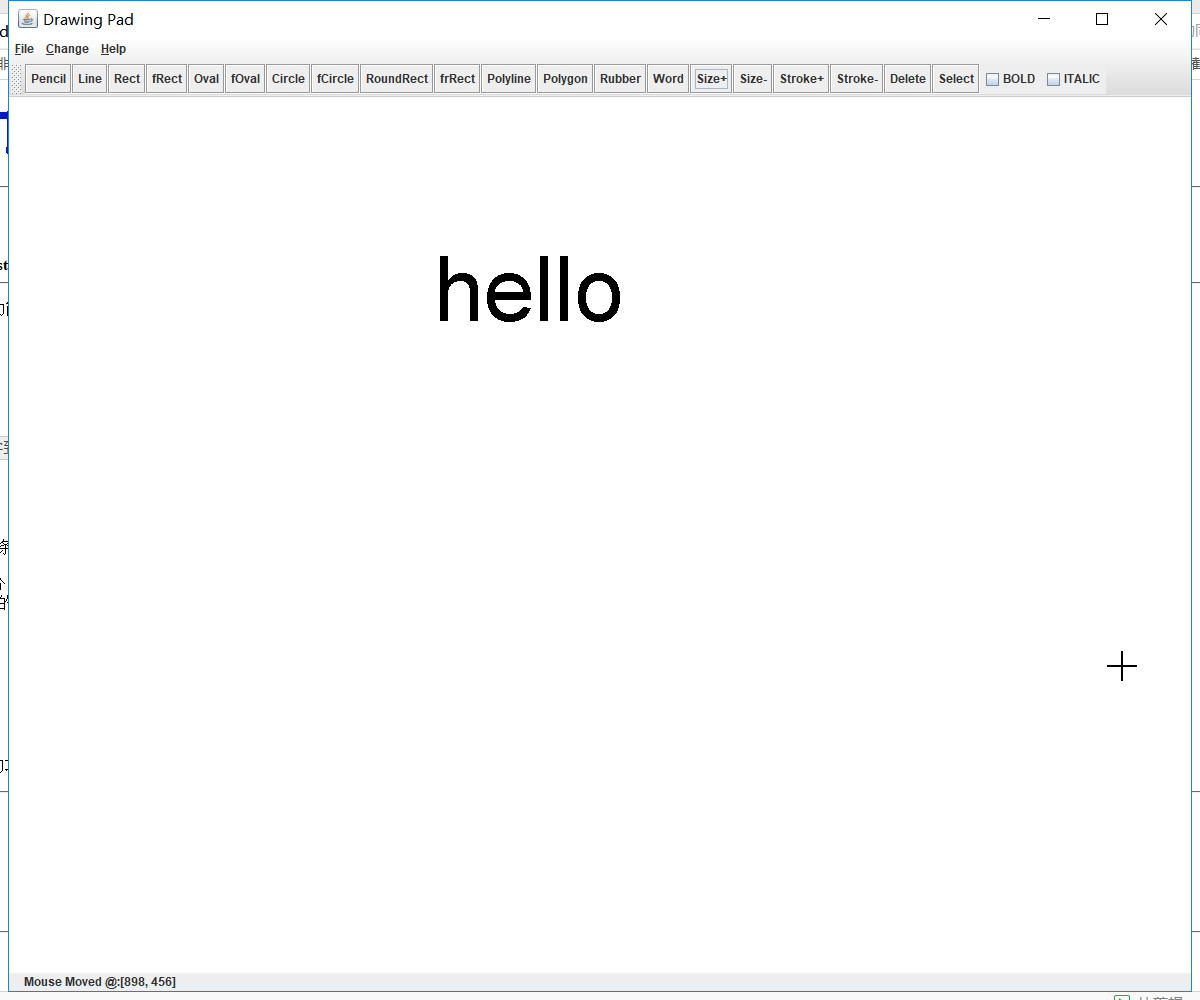
1. 实验截图
2. 整体界面



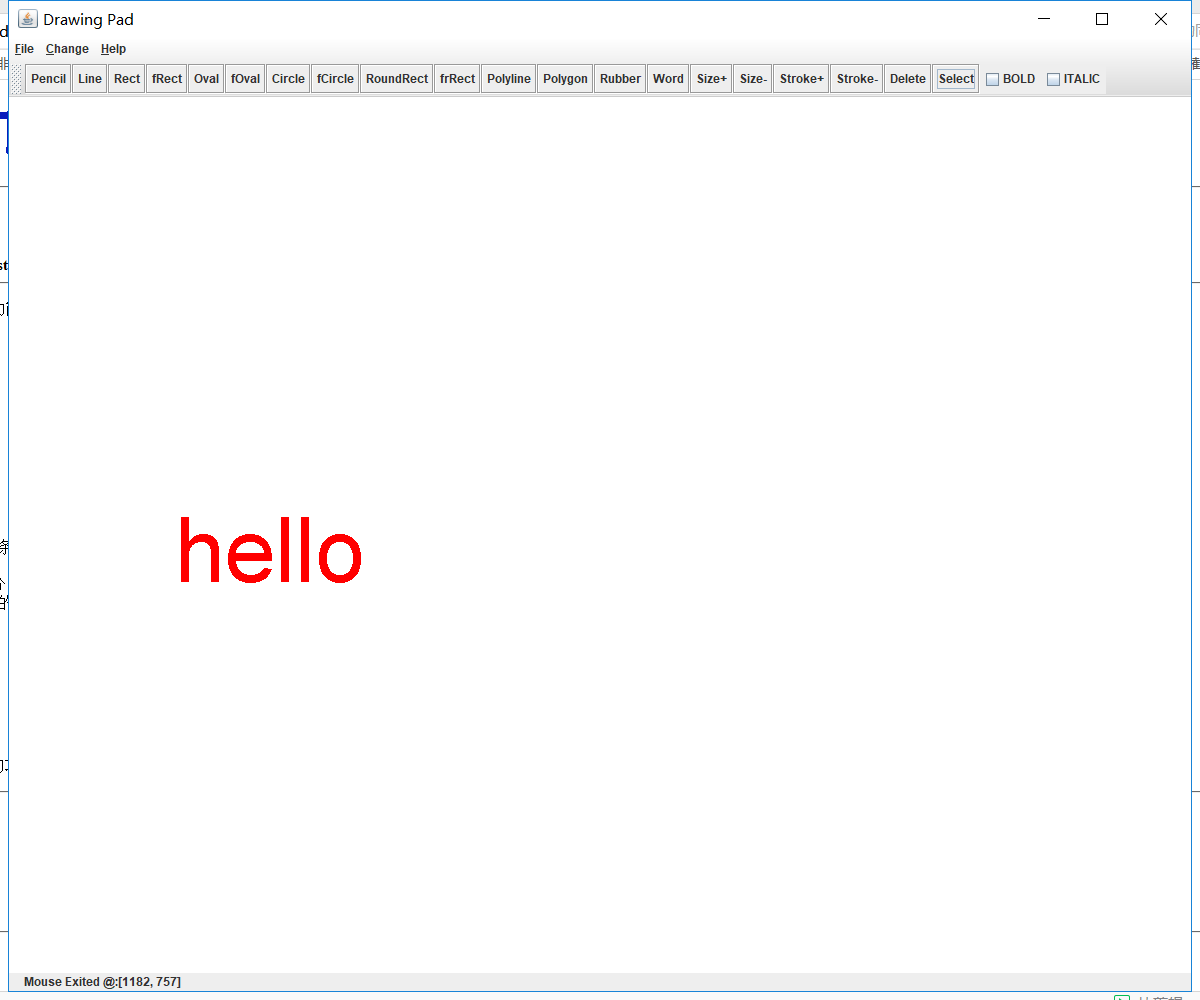
1. 基础图形的实现



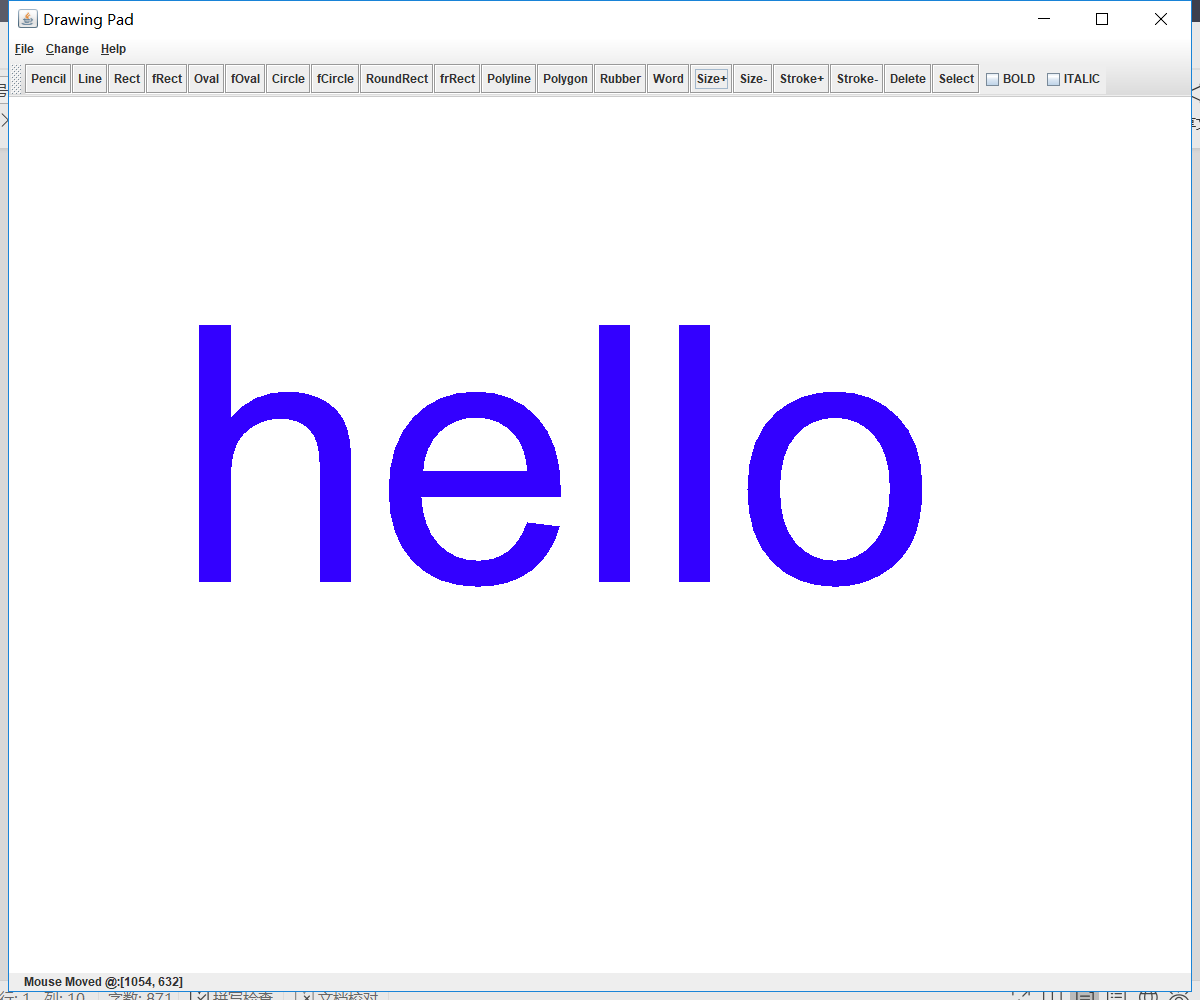
1. 添加文字块



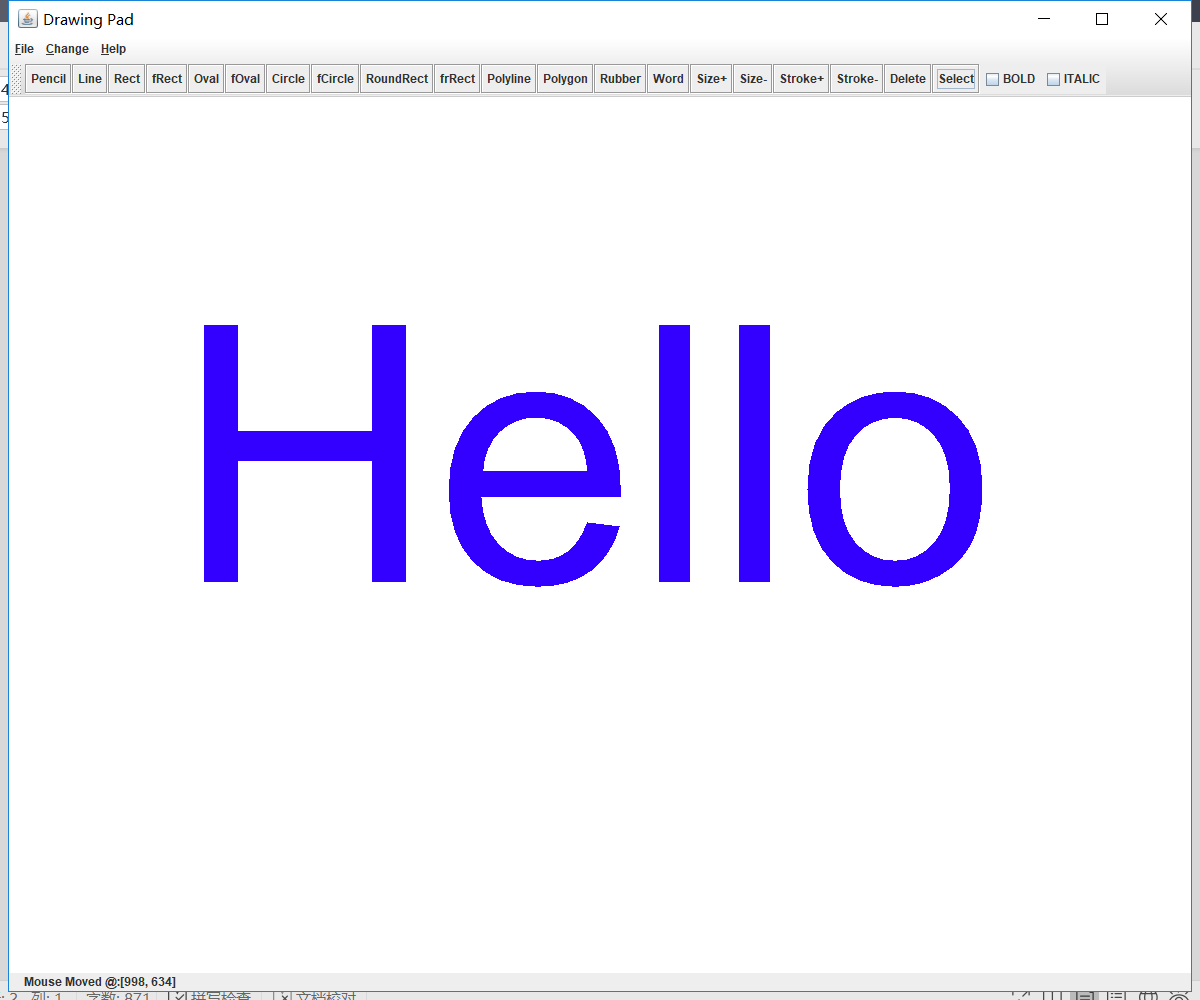
1. 选中并移动对象（选中后的对象会变红）



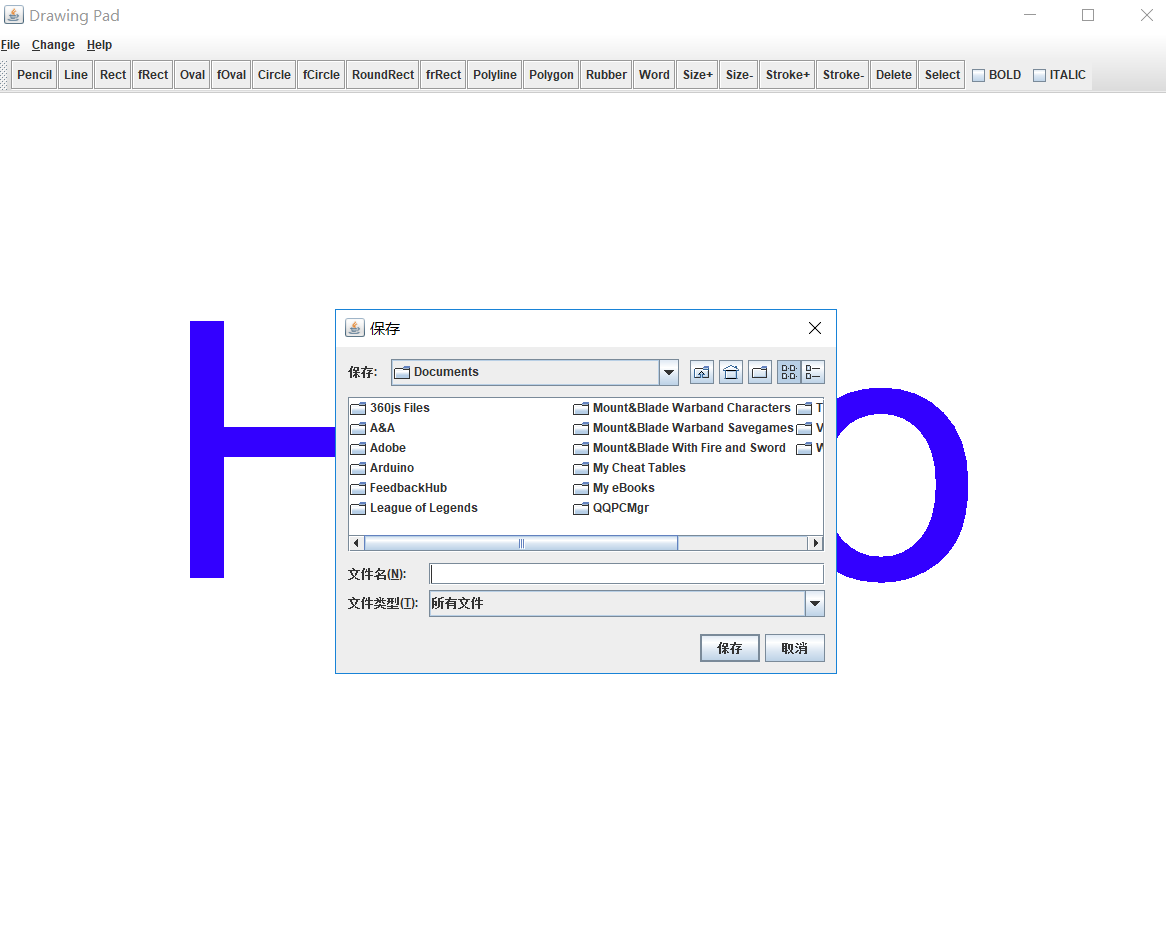
1. 修改对象大小和颜色



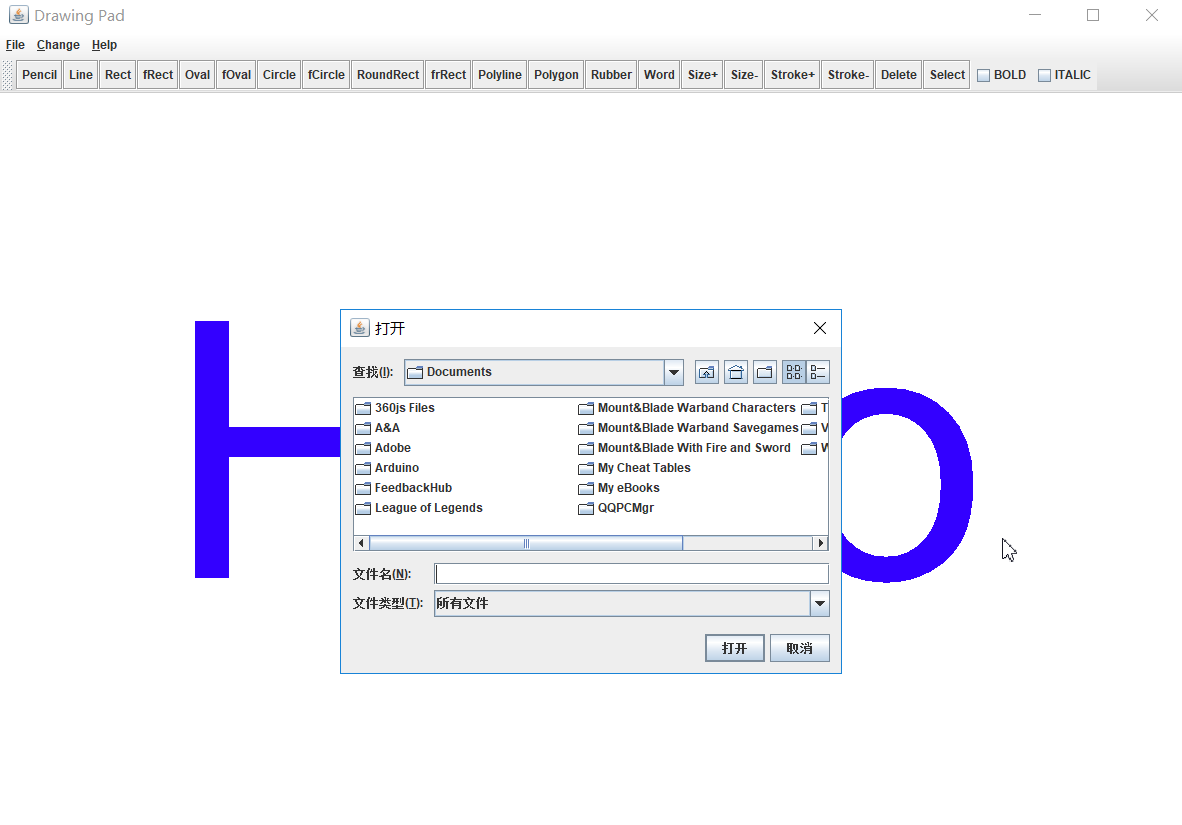
1. 修改文字内容



1. 保存文件界面



1. 读取文件界面



1. 程序源码

**package** miniCAD;

**import** java.awt.\*;

**import** java.awt.event.\*;

**import** javax.swing.\*;

**import** java.io.\*;

//定义画图的基本图形单元

**public** **class** Mini\_CAD **extends** JFrame //主类，扩展了JFrame类，用来生成主界面

{

**private** ObjectInputStream input;

**private** ObjectOutputStream output; //定义输入输出流，用来调用和保存图像文件

**private** JButton choices[]; //按钮数组，存放以下名称的功能按钮

**private** String names[] = {

"Pencil", //铅笔画，也就是用鼠标拖动着随意绘图

"Line", //绘制直线

"Rect", //绘制空心矩形

"fRect", //绘制以指定颜色填充的实心矩形

"Oval", //绘制空心椭圆

"fOval", //绘制以指定颜色填充的实心椭圆

"Circle", //绘制圆形

"fCircle", //绘制以指定颜色填充的实心圆形

"RoundRect", //绘制空心圆角矩形

"frRect", //绘制以指定颜色填充的实心圆角矩形

"Polyline",//绘制多点折线

"Polygon",//绘制多边形

"Rubber", //橡皮擦，可用来擦去已经绘制好的图案

"Word" , //输入文字按钮，可以在绘图板上实现文字输入

"Size+",

"Size-",

"Stroke+",

"Stroke-",

"Delete",

"Select" //选择

};

**private** String tipText[] = {

"Draw at will",

"Draw a straight line",

"Draw a rectangle",

"Fill a ractangle",

"Draw an oval",

"Fill an oval",

"Draw a circle",

"Fill a circle",

"Draw a round rectangle",

"Fill a round rectangle",

"Draw a polyline",

"Draw a polygon",

"Erase at will",

"Write down what u want",

"Increase size",

"Decrease size",

"Increase stroke",

"Decreae stroke",

"Delete a component",

"Select a component"

};

JToolBar buttonPanel; //定义按钮面板

**private** JLabel statusBar; //显示鼠标状态的提示条

**private** DrawPanel drawingArea; //画图区域

**private** **int** width = 1200, height = 1000; //定义画图区域初始大小

drawings[] itemList = **new** drawings[5000]; //用来存放基本图形的数组

**private** **int** currentChoice = 19; //设置默认画图状态为随笔画

**int** index = 0; //当前已经绘制的图形数目

**private** Color color = Color.***black***; //当前画笔颜色

**int** R, G, B; //用来存放当前色彩值

**int** f1, f2; //用来存放当前字体风格

**private** **float** stroke = 1.0f; //设置画笔粗细，默认值为1.0f

JCheckBox bold, italic; //定义字体风格选择框

**int** cur\_x,cur\_y;

**public** Mini\_CAD() //构造函数

{

**super**("Drawing Pad");

JMenuBar bar = **new** JMenuBar(); //定义菜单条

JMenu fileMenu = **new** JMenu("File");

fileMenu.setMnemonic('F');

//新建文件菜单条

JMenuItem newItem = **new** JMenuItem("New");

newItem.setMnemonic('N');

newItem.addActionListener(

**new** ActionListener() {

**public** **void** actionPerformed(ActionEvent e) {

newFile(); //如果被触发，则调用新建文件函数段

}

});

fileMenu.add(newItem);

//保存文件菜单项

JMenuItem saveItem = **new** JMenuItem("Save");

saveItem.setMnemonic('S');

saveItem.addActionListener(

**new** ActionListener() {

**public** **void** actionPerformed(ActionEvent e) {

saveFile(); //如果被触发，则调用保存文件函数段

}

});

fileMenu.add(saveItem);

//打开文件菜单项

JMenuItem loadItem = **new** JMenuItem("Load");

loadItem.setMnemonic('L');

loadItem.addActionListener(

**new** ActionListener() {

**public** **void** actionPerformed(ActionEvent e) {

loadFile(); //如果被触发，则调用打开文件函数段

}

});

fileMenu.add(loadItem);

fileMenu.addSeparator();

//退出菜单项

JMenuItem exitItem = **new** JMenuItem("Exit");

exitItem.setMnemonic('X');

exitItem.addActionListener(

**new** ActionListener() {

**public** **void** actionPerformed(ActionEvent e) {

System.*exit*(0); //如果被触发，则退出画图板程序

}

});

fileMenu.add(exitItem);

bar.add(fileMenu);

//设置颜色菜单条

JMenu changeMenu = **new** JMenu("Change");

changeMenu.setMnemonic('C');

//选择颜色菜单项

JMenuItem colorItem = **new** JMenuItem("Change Color");

colorItem.setMnemonic('O');

colorItem.addActionListener(

**new** ActionListener() {

**public** **void** actionPerformed(ActionEvent e) {

chooseColor(); //如果被触发，则调用选择颜色函数段

}

});

changeMenu.add(colorItem);

//bar.add(colorMenu);

//设置线条粗细菜单项

JMenuItem strokeItem = **new** JMenuItem("Change Stroke");

strokeItem.setMnemonic('K');

strokeItem.addActionListener(

**new** ActionListener() {

**public** **void** actionPerformed(ActionEvent e) {

setStroke();

}

});

changeMenu.add(strokeItem);

JMenuItem textItem = **new** JMenuItem("Change text");

textItem.setMnemonic('T');

textItem.addActionListener(

**new** ActionListener() {

**public** **void** actionPerformed(ActionEvent e) {

setText();

}

});

changeMenu.add(textItem);

bar.add(changeMenu);

//设置提示菜单条

JMenu helpMenu = **new** JMenu("Help");

helpMenu.setMnemonic('H');

//设置提示菜单项

JMenuItem aboutItem = **new** JMenuItem("About this Drawing Pad!");

aboutItem.setMnemonic('A');

aboutItem.addActionListener(

**new** ActionListener() {

**public** **void** actionPerformed(ActionEvent e) {

JOptionPane.*showMessageDialog*(**null**,

"This is a mini CAD pad!\nCopyright (c) 2018 Denny Lu ",

" Info ",

JOptionPane.***INFORMATION\_MESSAGE***);

}

});

helpMenu.add(aboutItem);

bar.add(helpMenu);

//创建各种基本图形的按钮

drawingArea = **new** DrawPanel();

choices = **new** JButton[names.length];

buttonPanel = **new** JToolBar(JToolBar.***VERTICAL***);

buttonPanel = **new** JToolBar(JToolBar.***HORIZONTAL***);

ButtonHandler handler = **new** ButtonHandler();

ButtonHandler1 handler1 = **new** ButtonHandler1();

**for** (**int** i = 0; i < choices.length; i++) {

choices[i] = **new** JButton(names[i]);

choices[i].setToolTipText(tipText[i]);

buttonPanel.add(choices[i]);

}

//将动作侦听器加入按钮里面

**for** (**int** i = 0; i < choices.length - 7; i++) {

choices[i].addActionListener(handler);

}

**for** (**int** i = choices.length-7; i < choices.length-1 ; i++) {

choices[i].addActionListener(handler1);

}

choices[choices.length - 1].addActionListener(handler);

bold = **new** JCheckBox("BOLD");

italic = **new** JCheckBox("ITALIC");

checkBoxHandler cHandler = **new** checkBoxHandler();

bold.addItemListener(cHandler);

italic.addItemListener(cHandler);

buttonPanel.add(bold);

buttonPanel.add(italic);

Container c = getContentPane();

**super**.setJMenuBar(bar);

c.add(buttonPanel, BorderLayout.***NORTH***);

c.add(drawingArea, BorderLayout.***CENTER***);

statusBar = **new** JLabel();

c.add(statusBar, BorderLayout.***SOUTH***);

statusBar.setText(" Welcome To The Little CAD Pad!!! :)");

createNewItem();

setSize(width, height);

**this**.setVisible(**true**);

}

//按钮侦听器ButtonHanler类，内部类，用来侦听基本按钮的操作

**public** **class** ButtonHandler **implements** ActionListener {

**public** **void** actionPerformed(ActionEvent e) {

**for** (**int** j = 0; j < choices.length ; j++) {

**if** (e.getSource() == choices[j]) {

currentChoice = j;

createNewItem();

repaint();

}

}

}

}

//按钮侦听器ButtonHanler1类，用来侦听颜色选择、画笔粗细设置、文字输入按钮的操作

**public** **class** ButtonHandler1 **implements** ActionListener {

**public** **void** actionPerformed(ActionEvent e) {

**if** (e.getSource() == choices[choices.length - 2]) {

**int** i;

**for**(i=0; i<index; i++)

{

**if**(itemList[i].ischoose)

itemList[i]=**new** drawings();

}

repaint();

}

**if** (e.getSource() == choices[choices.length - 3])

{

**int** i;

**for**(i=0; i<index; i++)

{

**if**((itemList[i].ischoose)&&(itemList[i].type!=13))

{

itemList[i].stroke=(**float**) (itemList[i].stroke\*0.95);

}

}

repaint();

}

**if** (e.getSource() == choices[choices.length - 4])

{

**int** i;

**for**(i=0; i<index; i++)

{

**if**((itemList[i].ischoose)&&(itemList[i].type!=13))

{

itemList[i].stroke=(**float**) (itemList[i].stroke\*1.05);

}

}

repaint();

}

**if** (e.getSource() == choices[choices.length - 5])

{

**int** i;

**for**(i=0; i<index; i++)

{

**if**(itemList[i].ischoose)

{

**if**(itemList[i].type!=13)

{

itemList[i].dx=(**int**) (itemList[i].dx\*0.95);

itemList[i].dy=(**int**) (itemList[i].dy\*0.95);

itemList[i].x2=itemList[i].x1+itemList[i].dx;

itemList[i].y2=itemList[i].y1+itemList[i].dy;

**if**((itemList[i+1].ischoose)&&(i<index-1))

{

itemList[i+1].x1=itemList[i].x2;

itemList[i+1].y1=itemList[i].y2;

}

}

**else**

{

itemList[i].stroke=(**float**) (itemList[i].stroke\*0.75);

}

}

}

repaint();

}

**if** (e.getSource() == choices[choices.length - 6])

{

**int** i;

**for**(i=0; i<index; i++)

{

**if**(itemList[i].ischoose)

{

**if**(itemList[i].type!=13)

{

itemList[i].dx=(**int**) (itemList[i].dx\*1.05);

itemList[i].dy=(**int**) (itemList[i].dy\*1.05);

itemList[i].x2=itemList[i].x1+itemList[i].dx;

itemList[i].y2=itemList[i].y1+itemList[i].dy;

**if**((itemList[i+1].ischoose)&&(i<index-1))

{

itemList[i+1].x1=itemList[i].x2;

itemList[i+1].y1=itemList[i].y2;

}

}

**else**

{

itemList[i].stroke=(**float**) (itemList[i].stroke+1);

}

}

}

repaint();

}

**if** (e.getSource() == choices[choices.length - 7]) {

JOptionPane.*showMessageDialog*(**null**,

"Please hit the drawing pad to choose the word input position",

"Hint", JOptionPane.***INFORMATION\_MESSAGE***);

currentChoice = 13;

createNewItem();

repaint();

}

}

}

//鼠标事件mouseA类，继承了MouseAdapter，用来完成鼠标相应事件操作

**class** mouseA **extends** MouseAdapter {

**public** **void** mousePressed(MouseEvent e) {

statusBar.setText(" Mouse Pressed @:[" + e.getX() +

", " + e.getY() + "]");//设置状态提示

**if**((currentChoice!=10)&&(currentChoice!=11)&&(currentChoice!=19))

{

itemList[index].x1 = itemList[index].x2 = e.getX();

itemList[index].y1 = itemList[index].y2 = e.getY();

}

//如果当前选择的图形是随笔画或者橡皮擦，则进行下面的操作

**if** (currentChoice == 0 || currentChoice == 12) {

itemList[index].x1 = itemList[index].x2 = e.getX();

itemList[index].y1 = itemList[index].y2 = e.getY();

index++;

createNewItem();

}

//如果当前选择的图形式文字输入，则进行下面操作

**if** (currentChoice == 13) {

itemList[index].x1 = e.getX();

itemList[index].y1 = e.getY();

String input;

input = JOptionPane.*showInputDialog*(

"Please input the text you want!");

itemList[index].s1 = input;

itemList[index].x2 = f1;

itemList[index].y2 = f2;

index++;

createNewItem();

drawingArea.repaint();

}

**if**(currentChoice==19)

{

cur\_x=e.getX();

cur\_y=e.getY();

}

}

**public** **void** mouseReleased(MouseEvent e) {

statusBar.setText(" Mouse Released @:[" + e.getX() +

", " + e.getY() + "]");

**if**((currentChoice!=10)&&(currentChoice!=11)&&(currentChoice!=19))

{

**if** (currentChoice == 0 || currentChoice == 12) {

itemList[index].x1 = e.getX();

itemList[index].y1 = e.getY();

}

itemList[index].x2 = e.getX();

itemList[index].y2 = e.getY();

repaint();

index++;

createNewItem();

}

}

@Override

**public** **void** mouseClicked(MouseEvent e)

{

**if**(currentChoice==10)

{

**if**(e.getButton()==MouseEvent.***BUTTON3***)

{

itemList[index].x1 = e.getX();

itemList[index].y1 = e.getY();

itemList[index].x2 = e.getX();

itemList[index].y2 = e.getY();

index++;

createNewItem();

}

**else** **if**(e.getButton()==MouseEvent.***BUTTON1***)

{

itemList[index-1].x2 = e.getX();

itemList[index-1].y2 = e.getY();

itemList[index].x1 = e.getX();

itemList[index].y1 = e.getY();

itemList[index].x2 = e.getX();

itemList[index].y2 = e.getY();

repaint();

index++;

createNewItem();

}

}

**if**(currentChoice==11)

{

**if**(e.getButton()==MouseEvent.***BUTTON3***)

{

itemList[index].x1 = e.getX();

itemList[index].y1 = e.getY();

itemList[index].x2 = e.getX();

itemList[index].y2 = e.getY();

itemList[index].init\_x = e.getX();

itemList[index].init\_y = e.getY();

itemList[index].end = **true**;

index++;

createNewItem();

}

**else** **if**(e.getButton()==MouseEvent.***BUTTON1***)

{

**if**(index>=1)

{

itemList[index-1].x2 = e.getX();

itemList[index-1].y2 = e.getY();

itemList[index-1].end = **false**;

itemList[index].init\_x=itemList[index-1].init\_x;

itemList[index].init\_y=itemList[index-1].init\_y;

itemList[index].x1 = e.getX();

itemList[index].y1 = e.getY();

itemList[index].x2 = e.getX();

itemList[index].y2 = e.getY();

itemList[index].end = **true**;

repaint();

index++;

createNewItem();

}

}

}

**if**(currentChoice==19)

{

**int** dis,choice;

**int** i,j,x,y;

x=e.getX();

y=e.getY();

**if**(e.getButton()==MouseEvent.***BUTTON3***)

{

**for**(j=0; j<index; j++)

itemList[j].ischoose=**false**;

repaint();

}

**else**

{

dis=(itemList[0].x1-x)\*(itemList[0].x1-x)+(itemList[0].y1-y)\*(itemList[0].y1-y);

choice=0;

**for**(i=0; i<index; i++)

{

**if**((itemList[i].x1-x)\*(itemList[i].x1-x)+(itemList[i].y1-y)\*(itemList[i].y1-y)<dis)

{

dis=(itemList[i].x1-x)\*(itemList[i].x1-x)+(itemList[i].y1-y)\*(itemList[i].y1-y);

choice=i;

}

}

**for**(j=0; j<index; j++)

itemList[j].ischoose=**false**;

itemList[choice].ischoose=**true**;

**int** forward=choice;

**int** backward=choice;

**while**((forward>=1)&&(itemList[forward-1].x2==itemList[forward].x1)&&(itemList[forward-1].y2==itemList[forward].y1))

{

itemList[forward-1].ischoose=**true**;

forward--;

}

**while**((backward<=index-1)&&(itemList[backward+1].x1==itemList[backward].x2)&&(itemList[backward+1].y1==itemList[backward].y2))

{

itemList[backward+1].ischoose=**true**;

backward++;

}

repaint();

}

}

}

**public** **void** mouseEntered(MouseEvent e) {

statusBar.setText(" Mouse Entered @:[" + e.getX() +

", " + e.getY() + "]");

}

**public** **void** mouseExited(MouseEvent e) {

statusBar.setText(" Mouse Exited @:[" + e.getX() +

", " + e.getY() + "]");

}

}

//鼠标事件mouseB类继承了MouseMotionAdapter，用来完成鼠标拖动和鼠标移动时的相应操作

**class** mouseB **extends** MouseMotionAdapter {

**public** **void** mouseDragged(MouseEvent e) {

statusBar.setText(" Mouse Dragged @:[" + e.getX() +

", " + e.getY() + "]");

**if**((currentChoice!=10)&&(currentChoice!=11)&&(currentChoice!=19))

{

**if** (currentChoice == 0 || currentChoice == 12) {

itemList[index - 1].x1 = itemList[index].x2 = itemList[index].x1 = e.getX();

itemList[index - 1].y1 = itemList[index].y2 = itemList[index].y1 = e.getY();

index++;

createNewItem();

} **else** {

itemList[index].x2 = e.getX();

itemList[index].y2 = e.getY();

}

repaint();

}

**if**(currentChoice==19)

{

**int** i;

**for**(i=0; i<index; i++)

**if**(itemList[i].ischoose)

{

**if**(itemList[i].type==13)

{

itemList[i].x1+=(e.getX()-cur\_x);

itemList[i].y1+=(e.getY()-cur\_y);

}

**else**

{

itemList[i].x1+=(e.getX()-cur\_x);

itemList[i].y1+=(e.getY()-cur\_y);

itemList[i].x2+=(e.getX()-cur\_x);

itemList[i].y2+=(e.getY()-cur\_y);

itemList[i].init\_x+=(e.getX()-cur\_x);

itemList[i].init\_y+=(e.getY()-cur\_y);

}

}

repaint();

cur\_x=e.getX();

cur\_y=e.getY();

}

}

**public** **void** mouseMoved(MouseEvent e) {

statusBar.setText(" Mouse Moved @:[" + e.getX() +

", " + e.getY() + "]");

}

}

//选择字体风格时候用到的事件侦听器类，加入到字体风格的选择框中

**private** **class** checkBoxHandler **implements** ItemListener {

**public** **void** itemStateChanged(ItemEvent e) {

**if** (e.getSource() == bold) {

**if** (e.getStateChange() == ItemEvent.***SELECTED***) {

f1 = Font.***BOLD***;

} **else** {

f1 = Font.***PLAIN***;

}

**int** i;

**for**(i=0; i<index; i++)

**if**((itemList[i].ischoose)&&(itemList[i].type==13))

{

itemList[i].x2=f1;

**break**;

}

repaint();

}

**if** (e.getSource() == italic) {

**if** (e.getStateChange() == ItemEvent.***SELECTED***) {

f2 = Font.***ITALIC***;

} **else** {

f2 = Font.***PLAIN***;

}

**int** i;

**for**(i=0; i<index; i++)

**if**((itemList[i].ischoose)&&(itemList[i].type==13))

{

itemList[i].y2=f2;

**break**;

}

repaint();

}

}

}

//画图面板类，用来画图

**class** DrawPanel **extends** JPanel {

**public** DrawPanel() {

setCursor(Cursor.*getPredefinedCursor*(Cursor.***CROSSHAIR\_CURSOR***));

setBackground(Color.***white***);

addMouseListener(**new** mouseA());

addMouseMotionListener(**new** mouseB());

}

@Override

**public** **void** paintComponent(Graphics g) {

**super**.paintComponent(g);

Graphics2D g2d = (Graphics2D) g; //定义画笔

**int** j = 0;

**while** (j <= index) {

draw(g2d, itemList[j]);

j++;

}

}

**void** draw(Graphics2D g2d, drawings i) {

i.draw(g2d);//将画笔传入到各个子类中，用来完成各自的绘图

}

}

//新建一个画图基本单元对象的程序段

**void** createNewItem() {

**if** (currentChoice == 13)//进行相应的游标设置

{

drawingArea.setCursor(Cursor.*getPredefinedCursor*(Cursor.***TEXT\_CURSOR***));

} **else** {

drawingArea.setCursor(Cursor.*getPredefinedCursor*(Cursor.***CROSSHAIR\_CURSOR***));

}

**switch** (currentChoice) {

**case** 0:

itemList[index] = **new** Pencil();

**break**;

**case** 1:

itemList[index] = **new** Line();

**break**;

**case** 2:

itemList[index] = **new** Rect();

**break**;

**case** 3:

itemList[index] = **new** fillRect();

**break**;

**case** 4:

itemList[index] = **new** Oval();

**break**;

**case** 5:

itemList[index] = **new** fillOval();

**break**;

**case** 6:

itemList[index] = **new** Circle();

**break**;

**case** 7:

itemList[index] = **new** fillCircle();

**break**;

**case** 8:

itemList[index] = **new** RoundRect();

**break**;

**case** 9:

itemList[index] = **new** fillRoundRect();

**break**;

**case** 10:

itemList[index] = **new** Polyline();

**break**;

**case** 11:

itemList[index] = **new** Polygon();

**break**;

**case** 12:

itemList[index] = **new** Rubber();

**break**;

**case** 13:

itemList[index] = **new** Word();

**break**;

**default**:

itemList[index] = **new** drawings();

**break**;

}

itemList[index].type = currentChoice;

itemList[index].R = R;

itemList[index].G = G;

itemList[index].B = B;

itemList[index].stroke = stroke;

}

//选择当前颜色程序段

**public** **void** chooseColor() {

color = JColorChooser.*showDialog*(Mini\_CAD.**this**,

"Choose a color", color);

**if**(color!=**null**)

{

R = color.getRed();

G = color.getGreen();

B = color.getBlue();

}

**int** i;

**for**(i=0; i<index; i++)

**if**(itemList[i].ischoose)

{

itemList[i].R=R;

itemList[i].G=G;

itemList[i].B=B;

itemList[i].ischoose=**false**;

}

repaint();

itemList[index].R = R;

itemList[index].G = G;

itemList[index].B = B;

}

//选择当前线条粗细程序段

**public** **void** setStroke() {

String input;

input = JOptionPane.*showInputDialog*(

"Please input a float stroke value! ( >0 )");

stroke = Float.*parseFloat*(input);

**int** i;

**for**(i=0; i<index; i++)

**if**((itemList[i].ischoose)&&(itemList[i].type!=13))

{

itemList[i].stroke=stroke;

itemList[i].ischoose=**false**;

}

repaint();

itemList[index].stroke = stroke;

}

**public** **void** setText() {

String input;

**int** i;

**for**(i=0; i<index; i++)

**if**((itemList[i].ischoose)&&(itemList[i].type==13))

{

input = JOptionPane.*showInputDialog*(

"Please input the text!");

itemList[i].s1=input;

itemList[i].ischoose=**false**;

**break**;

}

**if**(i>=index)

JOptionPane.*showMessageDialog*(**null**,

"Please select a valid text component!",

"Hint", JOptionPane.***INFORMATION\_MESSAGE***);

repaint();

}

//保存图形文件程序段

**public** **void** saveFile() {

JFileChooser fileChooser = **new** JFileChooser();

fileChooser.setFileSelectionMode(JFileChooser.***FILES\_ONLY***);

**int** result = fileChooser.showSaveDialog(**this**);

**if** (result == JFileChooser.***CANCEL\_OPTION***) {

**return**;

}

File fileName = fileChooser.getSelectedFile();

fileName.canWrite();

**if** (fileName == **null** || fileName.getName().equals("")) {

JOptionPane.*showMessageDialog*(fileChooser, "Invalid File Name",

"Invalid File Name", JOptionPane.***ERROR\_MESSAGE***);

} **else** {

**try** {

fileName.delete();

FileOutputStream fos = **new** FileOutputStream(fileName);

output = **new** ObjectOutputStream(fos);

//drawings record;

output.writeInt(index);

**for** (**int** i = 0; i < index; i++) {

drawings p = itemList[i];

output.writeObject(p);

output.flush(); //将所有图形信息强制转换成父类线性化存储到文件中

}

output.close();

fos.close();

} **catch** (IOException ioe) {

ioe.printStackTrace();

}

}

}

//打开一个图形文件程序段

**public** **void** loadFile() {

JFileChooser fileChooser = **new** JFileChooser();

fileChooser.setFileSelectionMode(JFileChooser.***FILES\_ONLY***);

**int** result = fileChooser.showOpenDialog(**this**);

**if** (result == JFileChooser.***CANCEL\_OPTION***) {

**return**;

}

File fileName = fileChooser.getSelectedFile();

fileName.canRead();

**if** (fileName == **null** || fileName.getName().equals("")) {

JOptionPane.*showMessageDialog*(fileChooser, "Invalid File Name",

"Invalid File Name", JOptionPane.***ERROR\_MESSAGE***);

} **else** {

**try** {

FileInputStream fis = **new** FileInputStream(fileName);

input = **new** ObjectInputStream(fis);

drawings inputRecord;

**int** countNumber = 0;

countNumber = input.readInt();

**for** (index = 0; index < countNumber; index++) {

inputRecord = (drawings) input.readObject();

itemList[index] = inputRecord;

}

createNewItem();

input.close();

repaint();

currentChoice=19;

} **catch** (EOFException endofFileException) {

JOptionPane.*showMessageDialog*(**this**, "no more record in file",

"class not found", JOptionPane.***ERROR\_MESSAGE***);

} **catch** (ClassNotFoundException classNotFoundException) {

JOptionPane.*showMessageDialog*(**this**, "Unable to Create Object",

"end of file", JOptionPane.***ERROR\_MESSAGE***);

} **catch** (IOException ioException) {

JOptionPane.*showMessageDialog*(**this**, "error during read from file",

"read Error", JOptionPane.***ERROR\_MESSAGE***);

}

}

}

//新建一个文件程序段

**public** **void** newFile() {

index = 0;

currentChoice = 19;

color = Color.***black***;

stroke = 1.0f;

createNewItem();

repaint();//将有关值设置为初始状态，并且重画

}

//主函数段

**public** **static** **void** main(String args[]) {

Mini\_CAD newPad = **new** Mini\_CAD();

newPad.setLocationRelativeTo(**null**);

newPad.setDefaultCloseOperation(JFrame.***EXIT\_ON\_CLOSE***);

}

}

**class** drawings **implements** Serializable//父类，基本图形单元，用到串行化接口，保存时所用

{

**int** x1, y1, x2, y2; //定义坐标属性

**int** dx,dy;

**int** R, G, B; //定义色彩属性

**int** init\_x,init\_y;

**boolean** end;

**boolean** ischoose=**false**;

**float** stroke; //定义线条粗细属性

**int** type;

String s1;

**void** draw(Graphics2D g2d) {

}

;//定义绘图函数

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

下面是各种基本图形单元的子类，都继承自父类drawings

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

**class** Line **extends** drawings //直线类

{

**void** draw(Graphics2D g2d) {

**this**.dx=x2-x1;

**this**.dy=y2-y1;

**if**(ischoose)

{

stroke=stroke\*2;

g2d.setPaint(**new** Color(255, 0, 0));

g2d.setStroke(**new** BasicStroke(stroke,

BasicStroke.***CAP\_ROUND***, BasicStroke.***JOIN\_BEVEL***));

g2d.drawLine(x1, y1, x2, y2);

stroke=stroke/2;

}

**else**

{

g2d.setPaint(**new** Color(R, G, B));

g2d.setStroke(**new** BasicStroke(stroke,

BasicStroke.***CAP\_ROUND***, BasicStroke.***JOIN\_BEVEL***));

g2d.drawLine(x1, y1, x2, y2);

}

}

}

**class** Rect **extends** drawings//矩形类

{

**void** draw(Graphics2D g2d) {

**this**.dx=x2-x1;

**this**.dy=y2-y1;

**if**(ischoose)

{

stroke=stroke\*2;

g2d.setPaint(**new** Color(255, 0, 0));

g2d.setStroke(**new** BasicStroke(stroke));

g2d.drawRect(Math.*min*(x1, x2), Math.*min*(y1, y2),

Math.*abs*(x1 - x2), Math.*abs*(y1 - y2));

stroke=stroke/2;

}

**else**

{

g2d.setPaint(**new** Color(R, G, B));

g2d.setStroke(**new** BasicStroke(stroke));

g2d.drawRect(Math.*min*(x1, x2), Math.*min*(y1, y2),

Math.*abs*(x1 - x2), Math.*abs*(y1 - y2));

}

}

}

**class** fillRect **extends** drawings//实心矩形类

{

**void** draw(Graphics2D g2d) {

**this**.dx=x2-x1;

**this**.dy=y2-y1;

**if**(ischoose)

{

stroke=stroke\*2;

g2d.setPaint(**new** Color(255, 0, 0));

g2d.setStroke(**new** BasicStroke(stroke));

g2d.fillRect(Math.*min*(x1, x2), Math.*min*(y1, y2),

Math.*abs*(x1 - x2), Math.*abs*(y1 - y2));

stroke=stroke/2;

}

**else**

{

g2d.setPaint(**new** Color(R, G, B));

g2d.setStroke(**new** BasicStroke(stroke));

g2d.fillRect(Math.*min*(x1, x2), Math.*min*(y1, y2),

Math.*abs*(x1 - x2), Math.*abs*(y1 - y2));

}

}

}

**class** Oval **extends** drawings//椭圆类

{

**void** draw(Graphics2D g2d) {

**this**.dx=x2-x1;

**this**.dy=y2-y1;

**if**(ischoose)

{

stroke=stroke\*2;

g2d.setPaint(**new** Color(255, 0, 0));

g2d.setStroke(**new** BasicStroke(stroke));

g2d.drawOval(Math.*min*(x1, x2), Math.*min*(y1, y2),

Math.*abs*(x1 - x2), Math.*abs*(y1 - y2));

stroke=stroke/2;

}

**else**

{

g2d.setPaint(**new** Color(R, G, B));

g2d.setStroke(**new** BasicStroke(stroke));

g2d.drawOval(Math.*min*(x1, x2), Math.*min*(y1, y2),

Math.*abs*(x1 - x2), Math.*abs*(y1 - y2));

}

}

}

**class** fillOval **extends** drawings//实心椭圆

{

**void** draw(Graphics2D g2d) {

**this**.dx=x2-x1;

**this**.dy=y2-y1;

**if**(ischoose)

{

stroke=stroke\*2;

g2d.setPaint(**new** Color(255, 0, 0));

g2d.setStroke(**new** BasicStroke(stroke));

g2d.fillOval(Math.*min*(x1, x2), Math.*min*(y1, y2),

Math.*abs*(x1 - x2), Math.*abs*(y1 - y2));

stroke=stroke/2;

}

**else**

{

g2d.setPaint(**new** Color(R, G, B));

g2d.setStroke(**new** BasicStroke(stroke));

g2d.fillOval(Math.*min*(x1, x2), Math.*min*(y1, y2),

Math.*abs*(x1 - x2), Math.*abs*(y1 - y2));

}

}

}

**class** Circle **extends** drawings//圆类

{

**void** draw(Graphics2D g2d) {

**this**.dx=x2-x1;

**this**.dy=y2-y1;

**if**(ischoose)

{

stroke=stroke\*2;

g2d.setPaint(**new** Color(255, 0, 0));

g2d.setStroke(**new** BasicStroke(stroke));

g2d.drawOval(Math.*min*(x1, x2), Math.*min*(y1, y2),

Math.*max*(Math.*abs*(x1 - x2), Math.*abs*(y1 - y2)),

Math.*max*(Math.*abs*(x1 - x2), Math.*abs*(y1 - y2)));

stroke=stroke/2;

}

**else**

{

g2d.setPaint(**new** Color(R, G, B));

g2d.setStroke(**new** BasicStroke(stroke));

g2d.drawOval(Math.*min*(x1, x2), Math.*min*(y1, y2),

Math.*max*(Math.*abs*(x1 - x2), Math.*abs*(y1 - y2)),

Math.*max*(Math.*abs*(x1 - x2), Math.*abs*(y1 - y2)));

}

}

}

**class** fillCircle **extends** drawings//实心圆

{

**void** draw(Graphics2D g2d) {

**this**.dx=x2-x1;

**this**.dy=y2-y1;

**if**(ischoose)

{

stroke=stroke\*2;

g2d.setPaint(**new** Color(255, 0, 0));

g2d.setStroke(**new** BasicStroke(stroke));

g2d.fillOval(Math.*min*(x1, x2), Math.*min*(y1, y2),

Math.*max*(Math.*abs*(x1 - x2), Math.*abs*(y1 - y2)),

Math.*max*(Math.*abs*(x1 - x2), Math.*abs*(y1 - y2)));

stroke=stroke/2;

}

**else**

{

g2d.setPaint(**new** Color(R, G, B));

g2d.setStroke(**new** BasicStroke(stroke));

g2d.fillOval(Math.*min*(x1, x2), Math.*min*(y1, y2),

Math.*max*(Math.*abs*(x1 - x2), Math.*abs*(y1 - y2)),

Math.*max*(Math.*abs*(x1 - x2), Math.*abs*(y1 - y2)));

}

}

}

**class** RoundRect **extends** drawings//圆角矩形类

{

**void** draw(Graphics2D g2d) {

**this**.dx=x2-x1;

**this**.dy=y2-y1;

**if**(ischoose)

{

stroke=stroke\*2;

g2d.setPaint(**new** Color(255, 0, 0));

g2d.setStroke(**new** BasicStroke(stroke));

g2d.drawRoundRect(Math.*min*(x1, x2), Math.*min*(y1, y2),

Math.*abs*(x1 - x2), Math.*abs*(y1 - y2),

50, 35);

stroke=stroke/2;

}

**else**

{

g2d.setPaint(**new** Color(R, G, B));

g2d.setStroke(**new** BasicStroke(stroke));

g2d.drawRoundRect(Math.*min*(x1, x2), Math.*min*(y1, y2),

Math.*abs*(x1 - x2), Math.*abs*(y1 - y2),

50, 35);

}

}

}

**class** fillRoundRect **extends** drawings//实心圆角矩形类

{

**void** draw(Graphics2D g2d) {

**this**.dx=x2-x1;

**this**.dy=y2-y1;

**if**(ischoose)

{

stroke=stroke\*2;

g2d.setPaint(**new** Color(255, 0, 0));

g2d.setStroke(**new** BasicStroke(stroke));

g2d.fillRoundRect(Math.*min*(x1, x2), Math.*min*(y1, y2),

Math.*abs*(x1 - x2), Math.*abs*(y1 - y2),

50, 35);

stroke=stroke/2;

}

**else**

{

g2d.setPaint(**new** Color(R, G, B));

g2d.setStroke(**new** BasicStroke(stroke));

g2d.fillRoundRect(Math.*min*(x1, x2), Math.*min*(y1, y2),

Math.*abs*(x1 - x2), Math.*abs*(y1 - y2),

50, 35);

}

}

}

**class** Pencil **extends** drawings//随笔画类

{

**void** draw(Graphics2D g2d) {

**this**.dx=x2-x1;

**this**.dy=y2-y1;

g2d.setPaint(**new** Color(R, G, B));

g2d.setStroke(**new** BasicStroke(stroke,

BasicStroke.***CAP\_ROUND***, BasicStroke.***JOIN\_BEVEL***));

g2d.drawLine(x1, y1, x2, y2);

}

}

**class** Polyline **extends** drawings//多点折线类

{

**void** draw(Graphics2D g2d) {

**this**.dx=x2-x1;

**this**.dy=y2-y1;

**if**(ischoose)

{

stroke=stroke\*2;

g2d.setPaint(**new** Color(255, 0, 0));

g2d.setStroke(**new** BasicStroke(stroke,

BasicStroke.***CAP\_ROUND***, BasicStroke.***JOIN\_BEVEL***));

g2d.drawLine(x1, y1, x2, y2);

stroke=stroke/2;

}

**else**

{

g2d.setPaint(**new** Color(R, G, B));

g2d.setStroke(**new** BasicStroke(stroke,

BasicStroke.***CAP\_ROUND***, BasicStroke.***JOIN\_BEVEL***));

g2d.drawLine(x1, y1, x2, y2);

}

}

}

**class** Polygon **extends** drawings//多边形类

{

**void** draw(Graphics2D g2d) {

**this**.dx=x2-x1;

**this**.dy=y2-y1;

**if**(ischoose)

{

stroke=stroke\*2;

g2d.setPaint(**new** Color(255, 0, 0));

g2d.setStroke(**new** BasicStroke(stroke,

BasicStroke.***CAP\_ROUND***, BasicStroke.***JOIN\_BEVEL***));

g2d.drawLine(x1, y1, x2, y2);

**if**(end)

g2d.drawLine(x2, y2, init\_x, init\_y);

stroke=stroke/2;

}

**else**

{

g2d.setPaint(**new** Color(R, G, B));

g2d.setStroke(**new** BasicStroke(stroke,

BasicStroke.***CAP\_ROUND***, BasicStroke.***JOIN\_BEVEL***));

g2d.drawLine(x1, y1, x2, y2);

**if**(end)

g2d.drawLine(x2, y2, init\_x, init\_y);

}

}

}

**class** Rubber **extends** drawings//橡皮擦类

{

**void** draw(Graphics2D g2d) {

**this**.dx=x2-x1;

**this**.dy=y2-y1;

g2d.setPaint(**new** Color(255, 255, 255));

g2d.setStroke(**new** BasicStroke(stroke + 4,

BasicStroke.***CAP\_ROUND***, BasicStroke.***JOIN\_BEVEL***));

g2d.drawLine(x1, y1, x2, y2);

}

}

**class** Word **extends** drawings//输入文字类

{

**void** draw(Graphics2D g2d) {

**if**(ischoose)

{

g2d.setPaint(**new** Color(255, 0, 0));

g2d.setFont(**new** Font(**null**, x2 + y2, ((**int**) stroke) \* 18));

**if** (s1 != **null**) {

g2d.drawString(s1, x1, y1);

}

}

**else**

{

g2d.setPaint(**new** Color(R, G, B));

g2d.setFont(**new** Font(**null**, x2 + y2, ((**int**) stroke) \* 18));

**if** (s1 != **null**) {

g2d.drawString(s1, x1, y1);

}

}

}

}