Java GUI 编程

GUI 编程简介

• 组件

- 窗口
- 弹窗
- 面板
- 文本框
- 列表框
- 按钮
- 图片
- 监听事件
- 鼠标
- 键盘事件
- 外挂
- 破解工具

• 简介

GUI 核心技术：

- Swing
- AWT

Java GUI 不流行的原因：

1. 界面不美观
2. 需要 jre 环境

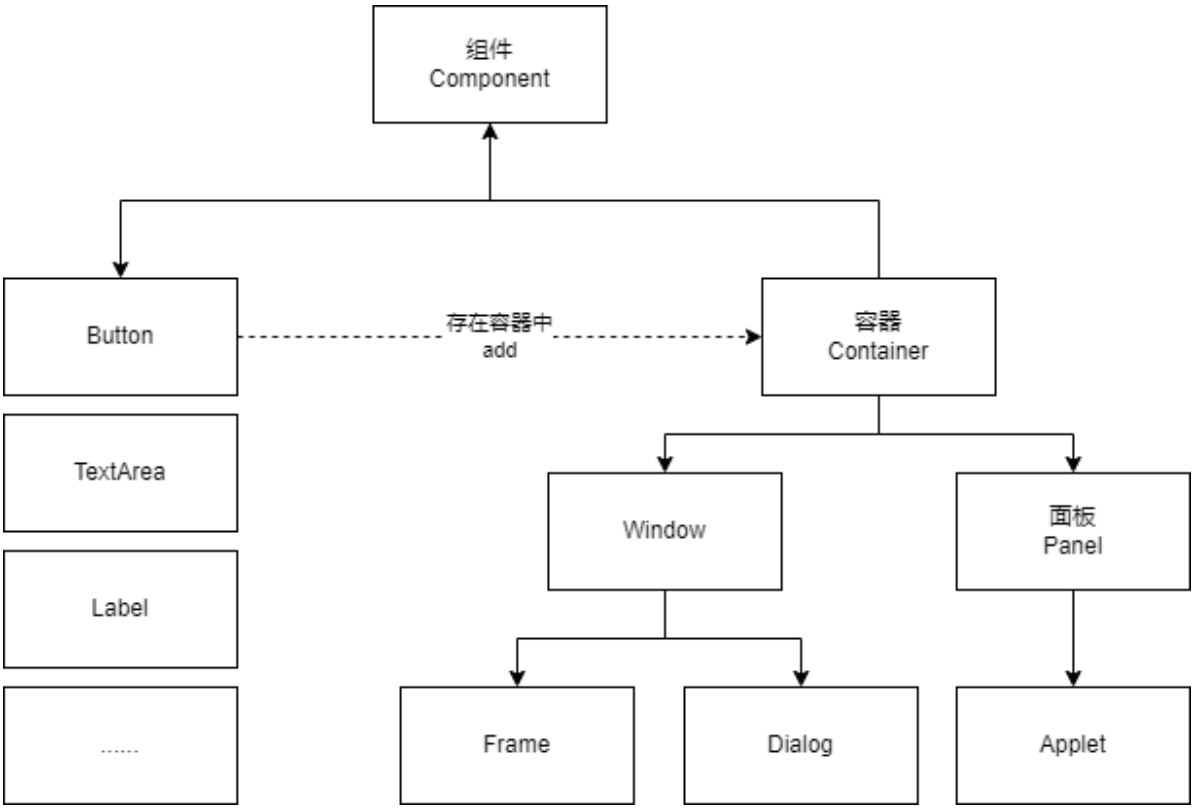
• 为什么要学习Java GUI?

1. 可以写出自己心中想要的一些小工具
2. 工作的时候，可能需要维护到 Swing 界面（概率极小！）
3. 了解 MVC 架构，了解监听

AWT

• AWT 介绍

1. 包含了很多的类和接口
2. GUI：图形用户界面编程
3. 元素：窗口、按钮、文本框
4. java.awt



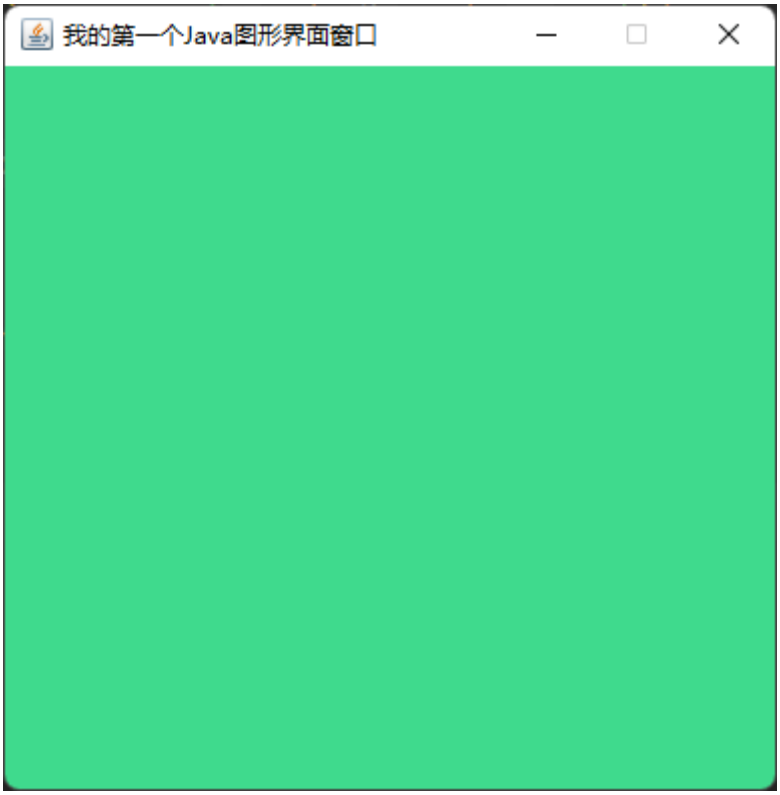
• 组件和容器

- 第一个窗口

```java
import java.awt.*;

public class TestFrame {
    public static void main(String[] args) {
        Frame frame = new Frame("我的第一个Java图形界面窗口");

        // 设置可见性
```

```
        frame.setVisible(true);

        // 设置窗口大小
        frame.setSize(400, 400);

        // 设置背景颜色
        frame.setBackground(new Color(63, 218, 141));

        // 弹出的初始位置
        frame.setLocation(200, 200);

        // 设置大小固定
        frame.setResizable(false);
    }
}
```

效果：



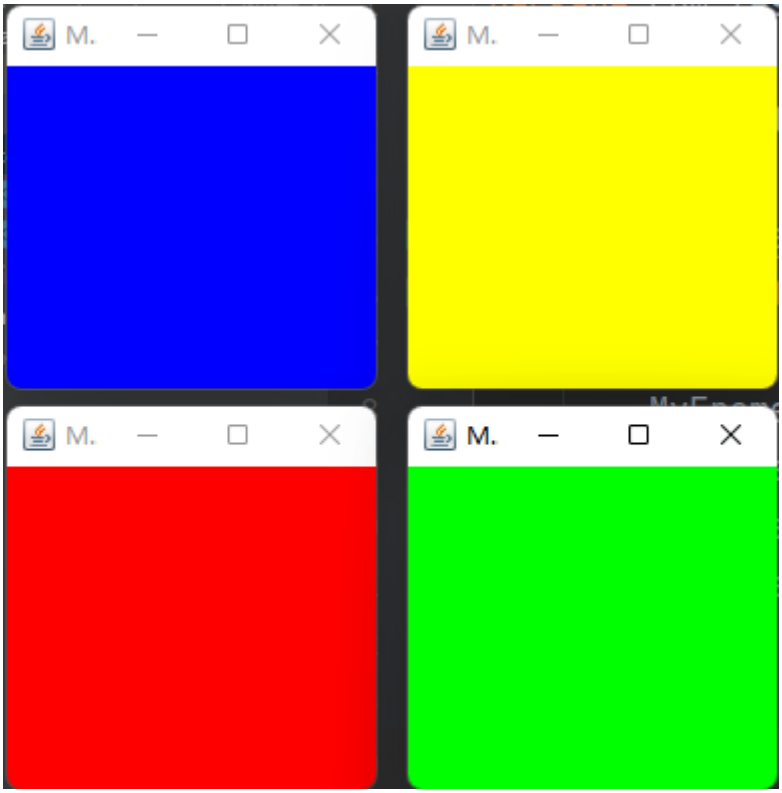
问题：窗口无法关闭，只能停止Java程序运行！

封装类以创建多个窗口

```
import java.awt.*;

public class TestFrame2 {
    public static void main(String[] args) {
        // 展示多个窗口
        MyFrame myFrame1 = new MyFrame(100, 100, 200, 200, Color.blue);
        MyFrame myFrame2 = new MyFrame(300, 100, 200, 200, Color.yellow);
        MyFrame myFrame3 = new MyFrame(100, 300, 200, 200, Color.red);
```

```
        MyFrame myFrame4 = new MyFrame(300, 300, 200, 200, Color.green);
    }
}

class MyFrame extends Frame {
    // 可能存在多个窗口，需要一个计数器
    static int id = 0;

    public MyFrame(int x, int y, int w, int h, Color color) {
        super("MyFrame" + (++id));
        setBackground(color);
        setBounds(x, y, w, h);
        setVisible(true);
    }
}
```

效果：



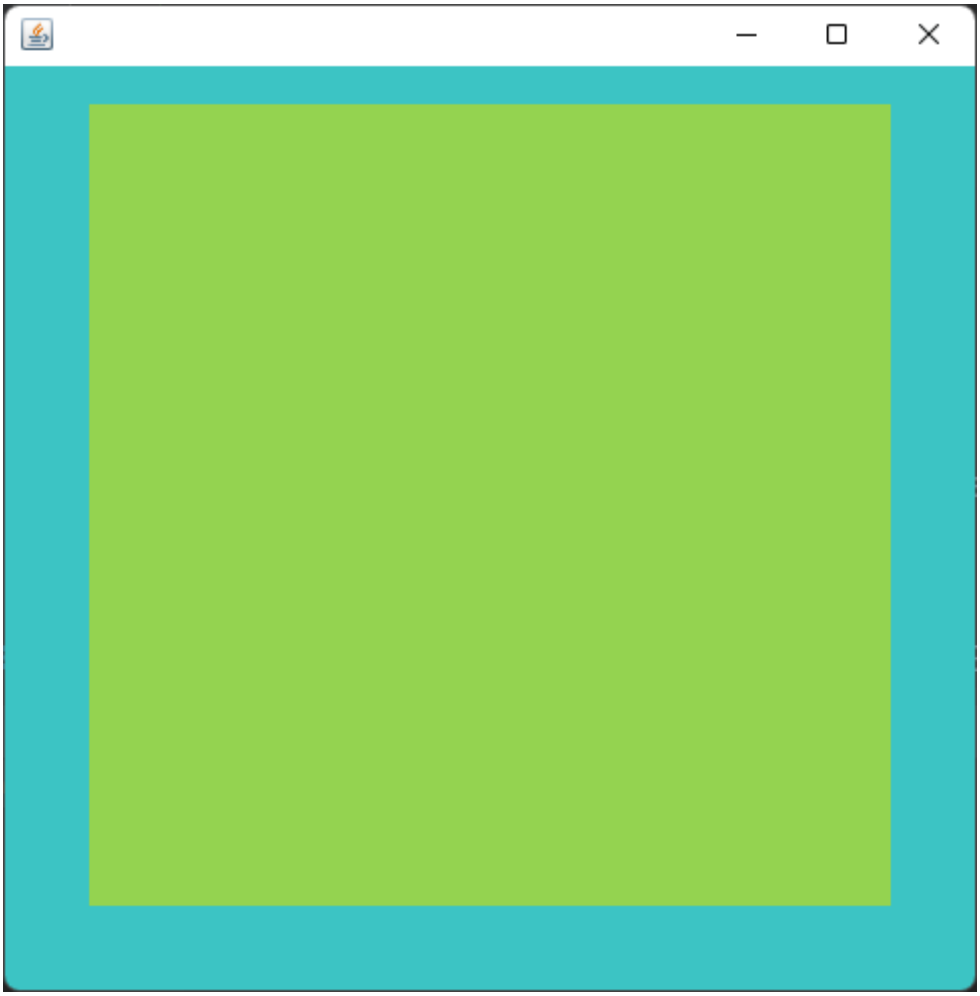
• 面板 Panel

解决了关闭事件！

```
import java.awt.*;
import java.awt.event.WindowAdapter;
import java.awt.event.WindowEvent;

// Panel 可以看成是一个空间，但是不能单独存在
public class TestPanel {
    public static void main(String[] args) {
```

```
        Frame frame = new Frame();

        Panel panel = new Panel();

        // 设置布局
        frame.setLayout(null);

        // 坐标
        frame.setBounds(300, 300, 500, 500);

        // 设置背景颜色
        frame.setBackground(new Color(60, 196, 196));

        // panel设置坐标，相对于frame
        panel.setBounds(50, 50, 400, 400);
        panel.setBackground(new Color(148, 211, 80));

        // 将panel添加到frame中
        frame.add(panel);

        frame.setVisible(true);

        // 监听窗口关闭事件
        // 适配器模式
        frame.addWindowListener(new WindowAdapter() {
            // 窗口点击关闭时要做的事情
            @Override
            public void windowClosing(WindowEvent e) {
                // 结束程序
                System.exit(0);
            }
        });
    }
}
```

效果：



- **布局管理器**
  - 流式布局

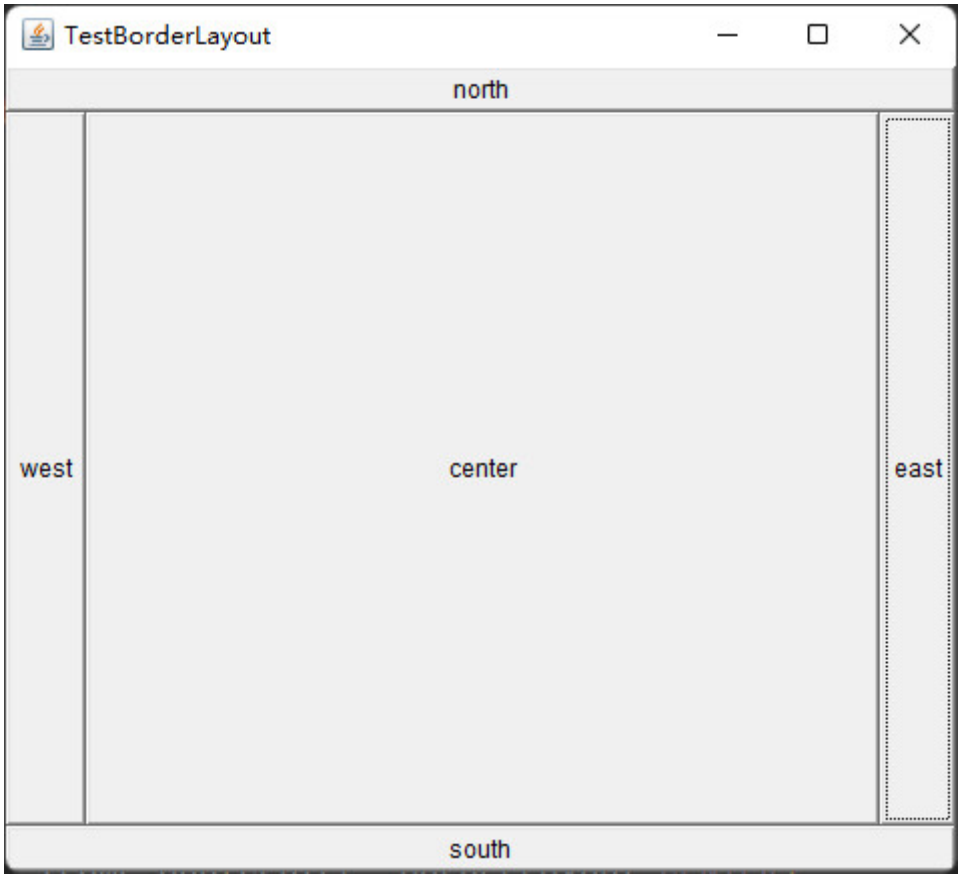
```
import java.awt.*;

public class TestFlowLayout {
    public static void main(String[] args) {
        Frame frame = new Frame();

        // 按钮组件
        Button button1 = new Button("button1");
        Button button2 = new Button("button2");
        Button button3 = new Button("button3");

        // 设置为流式布局
        // frame.setLayout(new FlowLayout());
        frame.setLayout(new FlowLayout(FlowLayout.LEFT));
        // frame.setLayout(new FlowLayout(FlowLayout.RIGHT));

        frame.setSize(200, 200);

        frame.setVisible(true);

```

```
            frame.add(button1);
            frame.add(button2);
            frame.add(button3);
        }
    }
```

- 东西南北中

```
import java.awt.*;

public class TestBorderLayout {
    public static void main(String[] args) {
        Frame frame = new Frame("TestBorderLayout");

        Button east = new Button("east");
        Button west = new Button("west");
        Button south = new Button("south");
        Button north = new Button("north");
        Button center = new Button("center");

        frame.add(east, BorderLayout.EAST);
        frame.add(west, BorderLayout.WEST);
        frame.add(south, BorderLayout.SOUTH);
        frame.add(north, BorderLayout.NORTH);
        frame.add(center, BorderLayout.CENTER);

        frame.setSize(200, 200);
        frame.setVisible(true);
    }
}
```

效果:



- 表格布局

```java
import java.awt.*;

public class TestGridLayout {
    public static void main(String[] args) {
        Frame frame = new Frame("TestGridLayout");

        Button btn1 = new Button("btn1");
        Button btn2 = new Button("btn2");
        Button btn3 = new Button("btn3");
        Button btn4 = new Button("btn4");
        Button btn5 = new Button("btn5");
        Button btn6 = new Button("btn6");

        frame.setLayout(new GridLayout(3, 2));

        frame.add(btn1);
        frame.add(btn2);
        frame.add(btn3);
        frame.add(btn4);
        frame.add(btn5);
        frame.add(btn6);

        frame.setVisible(true);
        // 自动布局
        frame.pack();
    }
```

```
27  }
```

效果:

- **练习**

```
1   import java.awt.*;
2
3   public class ExDemo {
4       public static void main(String[] args) {
5           Frame frame = new Frame("练习");
6
7           frame.setLayout(new GridLayout(2, 1));
8
9           // 4个面板
10          Panel panel1 = new Panel(new BorderLayout());
11          Panel panel2 = new Panel(new GridLayout(2, 1));
12          Panel panel3 = new Panel(new BorderLayout());
13          Panel panel4 = new Panel(new GridLayout(2, 1));
14
15          // 设置窗口
16          frame.setVisible(true);
17          frame.setSize(400, 300);
18          frame.setLocation(300, 400);
19          frame.setBackground(Color.white);
20
21          // 上面
22          panel1.add(new Button("east-1"), BorderLayout.EAST);
23          panel1.add(new Button("wast-1"), BorderLayout.WEST);
24
```

```
25            panel2.add(new Button("p2-btn-1"));
26            panel2.add(new Button("p2-btn-2"));
27
28            panel1.add(panel2, BorderLayout.CENTER);
29
30            // 下面
31            panel3.add(new Button("east-2"), BorderLayout.EAST);
32            panel3.add(new Button("wast-2"), BorderLayout.WEST);
33
34            // 中间4个
35            for (int i = 0; i < 4; i++) {
36                panel4.add(new Button("for-" + i));
37            }
38
39            panel3.add(panel4, BorderLayout.CENTER);
40
41            frame.add(panel1);
42            frame.add(panel3);
43
44            frame.addWindowListener(new WindowAdapter() {
45                @Override
46                public void windowClosing(WindowEvent e) {
47                    System.exit(0);
48                }
49            });
50        }
51    }
```

效果:

总结

1. Frame 是一个顶级窗口
2. Panel 无法单独显示，必须添加到某个容器中
3. 布局管理器
    1. 流式布局
    2. 东西南北中
    3. 表格布局
4. 大小、定位、背景颜色、可见性、监听

事件监听

介绍事件监听

```
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.awt.event.WindowAdapter;
import java.awt.event.WindowEvent;

public class TestActionEvent {
    public static void main(String[] args) {
        // 按下按钮，触发事件
```

```
        Frame frame = new Frame();
        Button button = new Button();

        MyActionListener myActionListener = new MyActionListener();
        button.addActionListener(myActionListener);

        frame.add(button, BorderLayout.CENTER);
        frame.pack();
        frame.setVisible(true);

        windowClose(frame);
    }

    // 关闭窗体的事件
    private static void windowClose(Frame frame) {
        frame.addWindowListener(new WindowAdapter() {
            @Override
            public void windowClosing(WindowEvent e) {
                System.exit(0);
            }
        });
    }
}

class MyActionListener implements ActionListener{

    @Override
    public void actionPerformed(ActionEvent e) {
        System.out.println("aaa");
    }
}
```

多个按钮共享一个事件监听

```
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.awt.event.WindowAdapter;
import java.awt.event.WindowEvent;

public class TestActionEvent2 {
    public static void main(String[] args) {
        // 两个按钮，监听同一个时间
        Frame frame = new Frame();
        Button button1 = new Button("start");
        Button button2 = new Button("stop");

        // 可以显示定义的触发会返回的命令，如果不定义，会返回默认值
        // 可以多个按钮只写一个监听
        button2.setActionCommand("button2-stop");
```

```
17
18          MyMonitor myMonitor = new MyMonitor();
19
20          button1.addActionListener(myMonitor);
21          button2.addActionListener(myMonitor);
22
23          frame.add(button1, BorderLayout.NORTH);
24          frame.add(button2, BorderLayout.SOUTH);
25
26          windowClose(frame);
27          frame.pack();
28          frame.setVisible(true);
29      }
30
31      // 关闭窗体的事件
32      private static void windowClose(Frame frame) {
33          frame.addWindowListener(new WindowAdapter() {
34              @Override
35              public void windowClosing(WindowEvent e) {
36                  System.exit(0);
37              }
38          });
39      }
40  }
41
42  class MyMonitor implements ActionListener {
43
44      // e.getActionCommand() 获得按钮的信息
45      @Override
46      public void actionPerformed(ActionEvent e) {
47          System.out.println("按钮被点击了:" + e.getActionCommand());
48      }
49  }
```

输入框 TextField 监听

```
1   import java.awt.*;
2   import java.awt.event.ActionEvent;
3   import java.awt.event.ActionListener;
4
5   public class TestText01 {
6       public static void main(String[] args) {
7           // 启动
8           new MyFrame();
9       }
10  }
11
12  class MyFrame extends Frame {
13      public MyFrame() {
14          TextField textField = new TextField();
```

```
            add(textField);

            // 监听这个文本框输入的文字
            MyActionListener myActionListener = new MyActionListener();

            // 按下Enter就会触发这个事件
            textField.addActionListener(myActionListener);

            // 替换编码
            textField.setEchoChar('*');

            pack();
            setVisible(true);
        }
    }

    class MyActionListener implements ActionListener {

        @Override
        public void actionPerformed(ActionEvent e) {
            // 获得一些资源
            TextField field = (TextField) e.getSource();
            // 获得输入框中的文本
            System.out.println(field.getText());
            // 按下Enter，清空输入框
            field.setText("");
        }
    }
```

- **练习：简易计算器**

oop 原则：组合大于继承！

```
class A extends B {

}

class A {
    public B b;
}
```

```
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
```

```java

// 简易计算器
public class TestCalc {
    public static void main(String[] args) {
        new Calculator();
    }
}

// 计算器类
class Calculator extends Frame {
    public Calculator() {

        // 3个文本框
        TextField num1 = new TextField(10);
        TextField num2 = new TextField(10);
        TextField num3 = new TextField(20);

        // 1个按钮
        Button button = new Button("=");
        button.addActionListener(new MyCalculatorListener(num1, num2, num3));


        // 1个标签
        Label label = new Label("+");

        setLayout(new FlowLayout());

        add(num1);
        add(label);
        add(num2);
        add(button);
        add(num3);

        pack();
        setVisible(true);
    }
}

// 监听器类
class MyCalculatorListener implements ActionListener {

    // 获取3个变量
    private TextField num1, num2, num3;

    public MyCalculatorListener(TextField num1, TextField num2, TextField
num3) {
        this.num1 = num1;
        this.num2 = num2;
        this.num3 = num3;
    }

```

```
53          @Override
54          public void actionPerformed(ActionEvent e) {
55              // 获取加数和被加数
56              int n1 = Integer.parseInt(num1.getText());
57              int n2 = Integer.parseInt(num2.getText());
58
59              // 将这个值加法运算后放到第三个框
60              num3.setText((n1 + n2) + "");
61
62              // 清除前两个框
63              num1.setText("");
64              num2.setText("");
65          }
66      }
```

效果：

完全改造为面向对象的写法：

```
1   import java.awt.*;
2   import java.awt.event.ActionEvent;
3   import java.awt.event.ActionListener;
4
5   // 简易计算器
6   public class TestCalc {
7       public static void main(String[] args) {
8           new Calculator().loadFrame();
9       }
10  }
11
12  // 计算器类
13  class Calculator extends Frame {
14
15      TextField num1, num2, num3;
16
17      public void loadFrame() {
18
19          // 3个文本框
20          num1 = new TextField(10);
21          num2 = new TextField(10);
22          num3 = new TextField(20);
```

```
23          Button button = new Button("=");
24          Label label = new Label("+");
25
26          button.addActionListener(new MyCalculatorListener(this));
27
28          setLayout(new FlowLayout());
29
30          add(num1);
31          add(label);
32          add(num2);
33          add(button);
34          add(num3);
35
36          pack();
37          setVisible(true);
38      }
39  }
40
41  // 监听器类
42  class MyCalculatorListener implements ActionListener {
43
44      // 获取计算器这个对象，在一个类中组合另外一个类
45      Calculator calculator = null;
46
47      public MyCalculatorListener(Calculator calculator) {
48          this.calculator = calculator;
49      }
50
51      @Override
52      public void actionPerformed(ActionEvent e) {
53          // 获取加数和被加数
54          int n1 = Integer.parseInt(calculator.num1.getText());
55          int n2 = Integer.parseInt(calculator.num2.getText());
56
57          // 将这个值加法运算后放到第三个框
58          calculator.num3.setText((n1 + n2) + "");
59
60          // 清除前两个框
61          calculator.num1.setText("");
62          calculator.num2.setText("");
63      }
64  }
```

内部类写法：

```
```

```java
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;

// 简易计算器
public class TestCalc {
    public static void main(String[] args) {
        new Calculator().loadFrame();
    }
}

// 计算器类
class Calculator extends Frame {

    TextField num1, num2, num3;

    public void loadFrame() {

        // 3个文本框
        num1 = new TextField(10);
        num2 = new TextField(10);
        num3 = new TextField(20);
        Button button = new Button("=");
        Label label = new Label("+");

        button.addActionListener(new MyCalculatorListener());

        setLayout(new FlowLayout());

        add(num1);
        add(label);
        add(num2);
        add(button);
        add(num3);

        pack();
        setVisible(true);
    }

    // 监听器类
    private class MyCalculatorListener implements ActionListener {

        @Override
        public void actionPerformed(ActionEvent e) {
            // 获取加数和被加数
            int n1 = Integer.parseInt(num1.getText());
            int n2 = Integer.parseInt(num2.getText());

            // 将这个值加法运算后放到第三个框
            num3.setText((n1 + n2) + "");

```

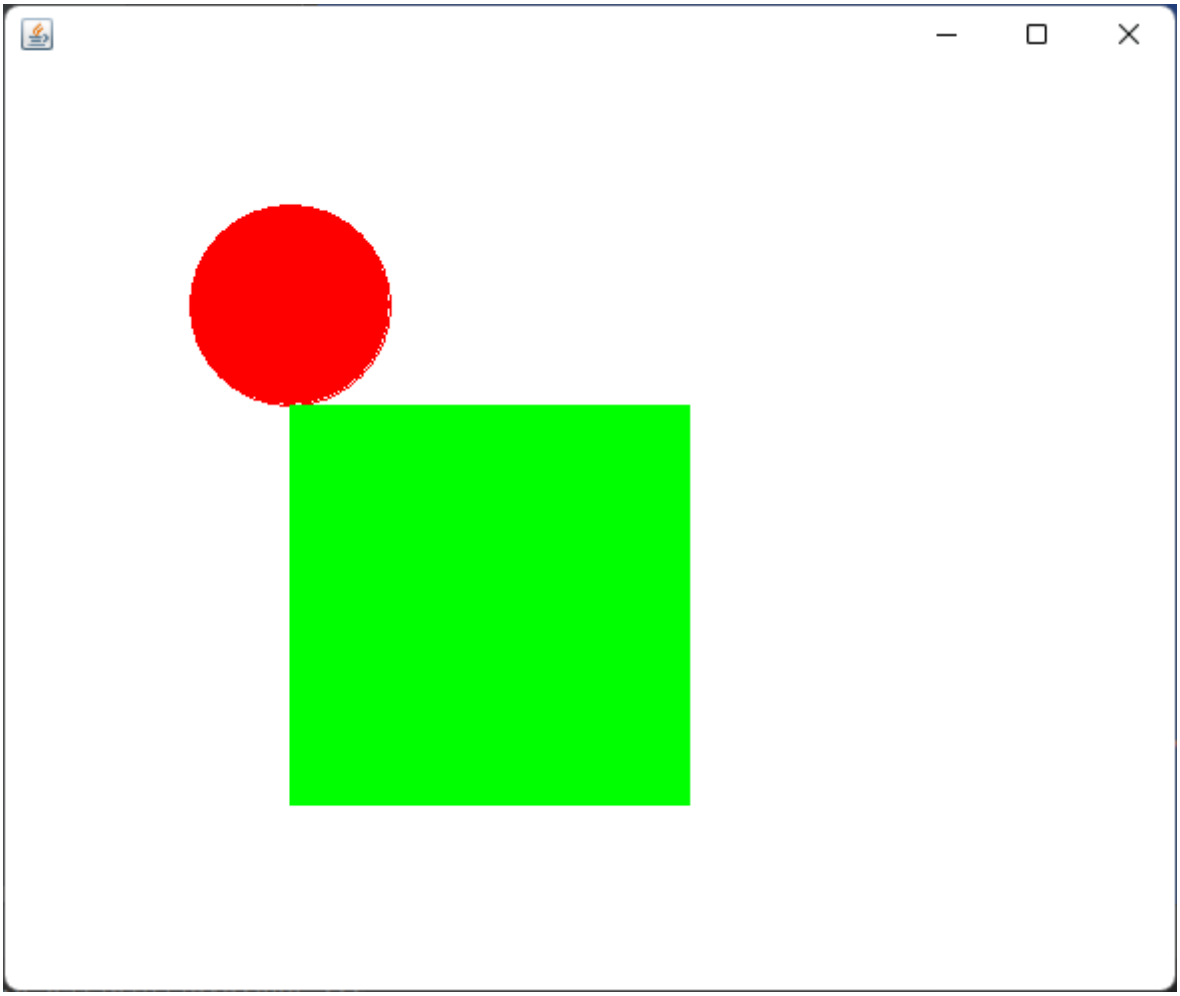
```
52                  // 清除前两个框
53                  num1.setText("");
54                  num2.setText("");
55              }
56          }
57      }
```

> “
>
> 内部类最大的好处：可以直接访问外部类

• 画笔 Paint

```
1   import java.awt.*;
2
3   public class TestPaint {
4       public static void main(String[] args) {
5           new MyPaint().LoadFrame();
6       }
7   }
8
9   class MyPaint extends Frame {
10
11      public void LoadFrame() {
12          setVisible(true);
13          setBounds(200, 200, 600, 500);
14      }
15
16      // 画笔
17      @Override
18      public void paint(Graphics g) {
19          // 画笔需要有颜色
20          g.setColor(Color.red);
21          g.drawOval(100, 100, 100, 100);
22          // 实心圆
23          g.fillOval(100, 100, 100, 100);
24
25          g.setColor(Color.green);
26          g.fillRect(150, 200, 200, 200);
27          // 画笔用完，还原到最初的颜色
28      }
29  }
```

效果：

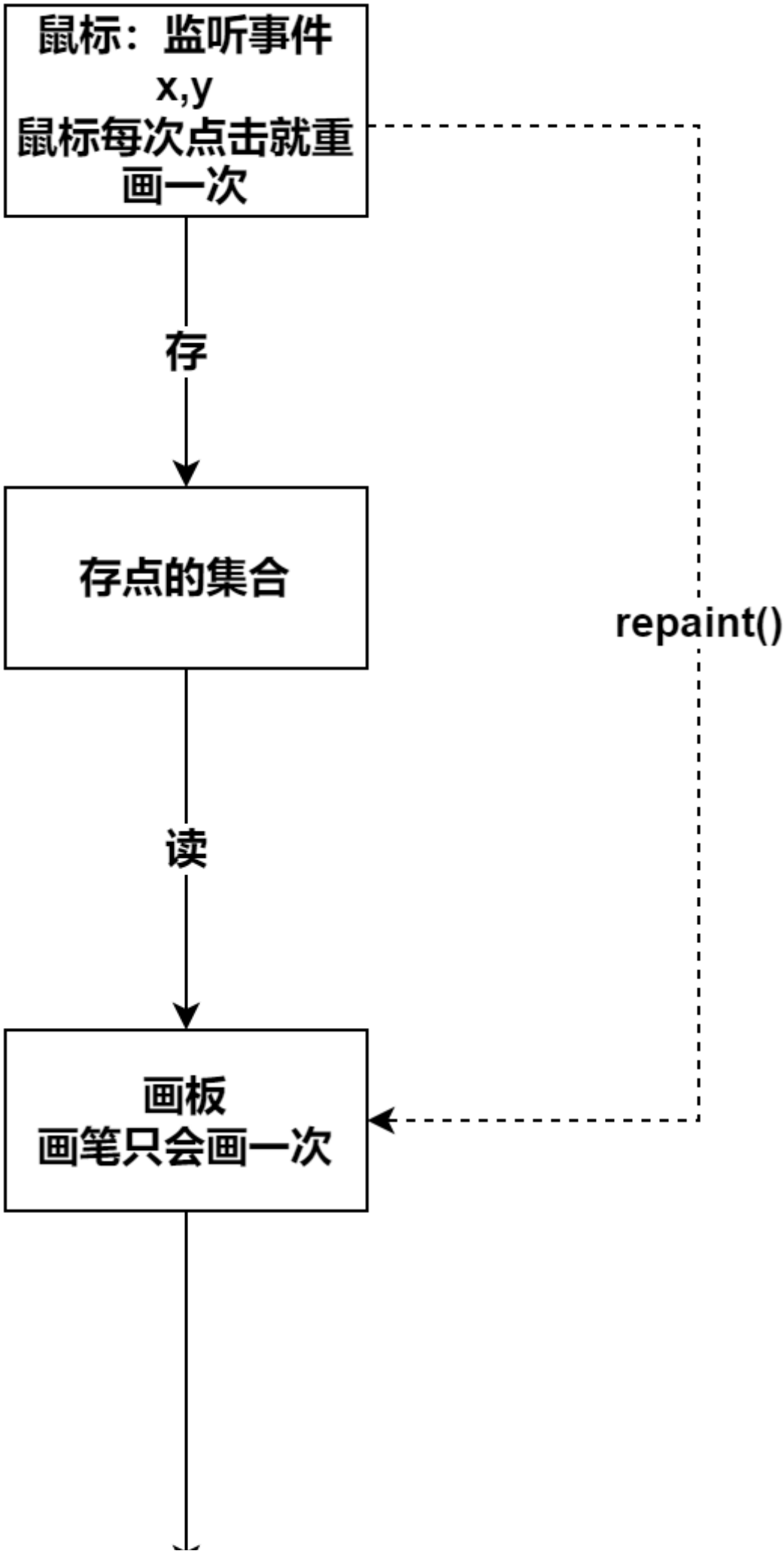


- **鼠标监听**

目的：实现鼠标画画

```
import java.awt.*;
import java.awt.event.MouseAdapter;
import java.awt.event.MouseEvent;
import java.util.ArrayList;
import java.util.Iterator;

public class TestMouseListener {
    public static void main(String[] args) {
        new MyFrame("画板");
    }
}

// 鼠标类
class MyFrame extends Frame {
    // 画画需要画笔，需要监听鼠标当前的位置，需要集合来存储这个点
    ArrayList points;

    public MyFrame(String title) {
        super(title);
        setBounds(200, 200, 400, 300);
```

```
21            setVisible(true);
22
23            // 存鼠标的点
24            points = new ArrayList<>();
25
26            // 鼠标监听器，正对这个窗口
27            this.addMouseListener(new MouseListener());
28        }
29
30        @Override
31        public void paint(Graphics g) {
32            // 画画，需要监听鼠标的事件
33            Iterator iterator = points.iterator();
34            while (iterator.hasNext()) {
35                Point point = (Point) iterator.next();
36                g.setColor(Color.blue);
37                g.fillOval(point.x, point.y, 10, 10);
38            }
39        }
40
41        // 添加一个点到界面上
42        public void addPoint(Point point) {
43            points.add(point);
44        }
45
46        private class MouseListener extends MouseAdapter {
47            @Override
48            public void mousePressed(MouseEvent e) {
49                MyFrame myFrame = (MyFrame) e.getSource();
50                // 这个点就是鼠标的点
51                myFrame.addPoint(new Point(e.getX(), e.getY()));
52
53                // 每次点击鼠标都需要重新画一遍
54                myFrame.repaint();
55            }
56        }
57    }
```


鼠标：监听事件
x,y
鼠标每次点击就重画一次
存
存点的集合
读
画板
画笔只会画一次
repaint()

- **窗口监听**

```java
import java.awt.*;
import java.awt.event.WindowAdapter;
import java.awt.event.WindowEvent;

public class TestWindow {
    public static void main(String[] args) {
        new WindowFrame();
    }
}

class WindowFrame extends Frame {
    public WindowFrame() {
        setVisible(true);
        setBackground(Color.blue);
        setBounds(100, 100, 200, 200);
        // addWindowListener(new MyWindowListener());

        this.addWindowListener(
                // 匿名内部类
                new WindowAdapter() {
                    // 关闭窗口
                    @Override
                    public void windowClosing(WindowEvent e) {
                        // 隐藏窗口
                        setVisible(false);

                        // 正常退出，非正常退出只需将status设置为1即可
                        System.exit(0);
                    }

                    // 激活窗口
                    @Override
                    public void windowActivated(WindowEvent e) {
                        WindowFrame frame = (WindowFrame) e.getSource();
                        frame.setTitle("被激活了");
                        System.out.println("windowActivated");
                    }
                });
    }
```

```
}
```

- **键盘监听**

```
import java.awt.*;
import java.awt.event.KeyAdapter;
import java.awt.event.KeyEvent;

public class TestKeyListener {
    public static void main(String[] args) {
        new KeyFrame();
    }
}

class KeyFrame extends Frame {
    public KeyFrame() {
        setBounds(1, 2, 300, 400);
        setVisible(true);

        this.addKeyListener(new KeyAdapter() {
            // 键盘按下
            @Override
            public void keyPressed(KeyEvent e) {
                // 获得键盘按下的键
                int keyCode = e.getKeyCode();
                System.out.println(keyCode);
                if (keyCode == KeyEvent.VK_UP) {
                    System.out.println("你按下了上键");
                }
                // 根据按下的不同键，产生不同的结果
            }
        });
    }
}
```

Swing

- **窗口、面板**

Dialog：用来被弹出，**默认就有关闭事件**！

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

public class JFrameDemo {

```

```
    public void init() {
        JFrame frame = new JFrame("这是一个JFrame窗口");
        frame.setVisible(true);
        frame.setBounds(100, 100, 200, 200);
        frame.setBackground(Color.cyan);

        // 设置文字JLabel
        JLabel jLabel = new JLabel("欢迎使用本软件! ");

        frame.add(jLabel);

        // 让文本标签居中 设置水平对齐
        jLabel.setHorizontalAlignment(SwingConstants.CENTER);

        // 容器实例化
        Container contentPane = frame.getContentPane();
        contentPane.setBackground(Color.white);

        // 关闭事件
        frame.setDefaultCloseOperation(WindowConstants.EXIT_ON_CLOSE);
    }

    public static void main(String[] args) {
        // 建立一个窗口
        new JFrameDemo().init();
    }
}
```

- **弹窗**

```
import javax.swing.*;
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;

// 主窗口
public class TestDialog extends JFrame {

    public TestDialog() {
        this.setVisible(true);
        this.setSize(700, 500);
        this.setDefaultCloseOperation(WindowConstants.EXIT_ON_CLOSE);

        Container container = this.getContentPane();

        // 绝对布局
        container.setLayout(null);

        // 按钮
```

```
        JButton button = new JButton("点击弹出一个对话框");
        button.setBounds(30, 30, 200, 50);

        // 点击按钮的时候，弹出一个弹窗
        button.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                // 弹窗
                new MyDialog();
            }
        });

        container.add(button);
    }

    public static void main(String[] args) {
        new TestDialog();
    }
}

// 弹窗的窗口
class MyDialog extends JDialog {
    public MyDialog() {
        this.setVisible(true);
        this.setBounds(100, 100, 500, 500);
        // this.setDefaultCloseOperation(WindowConstants.EXIT_ON_CLOSE);

        Container container = this.getContentPane();
        container.setLayout(null);

        container.add(new Label("Hello, java!"));
    }
}
```

- **标签**
  - Label

```
new JLabel("文本内容");
```

  - 图标 Icon

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

// 图标，需要实现类，继承JFrame
public class TestIcon extends JFrame implements Icon {
```

```
6
7       private int width;
8       private int height;
9
10      public TestIcon() {
11      }
12      public TestIcon(int width, int height) {
13          this.height = height;
14          this.width = width;
15      }
16
17      public static void main(String[] args) {
18          new TestIcon().init();
19      }
20
21      public void init() {
22          TestIcon icon = new TestIcon(15, 15);
23          // 图标放在标签上，也可以放在按钮上
24          JLabel jLabel = new JLabel("icontest", icon, SwingConstants.CENTER);

25
26          Container container = getContentPane();
27          container.add(jLabel);
28
29          this.setVisible(true);
30          this.setDefaultCloseOperation(WindowConstants.EXIT_ON_CLOSE);
31      }
32
33      @Override
34      public void paintIcon(Component c, Graphics g, int x, int y) {
35          g.fillOval(x, y, width, height);
36      }
37
38      @Override
39      public int getIconWidth() {
40          return this.width;
41      }
42
43      @Override
44      public int getIconHeight() {
45          return this.height;
46      }
47  }
```

- 图片图标 ImageIcon

```
1   import javax.swing.*;
2   import java.awt.*;
3   import java.net.URL;
4
```

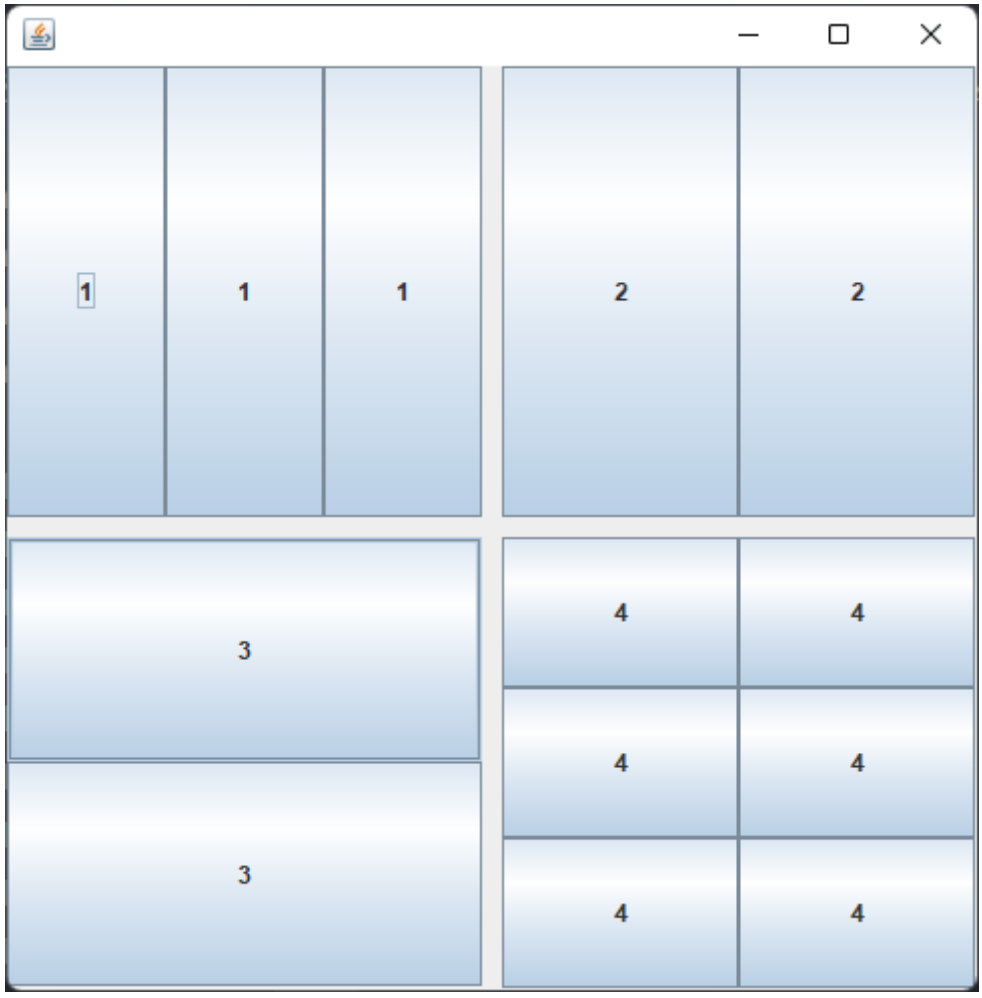
```
public class TestImageIcon extends JFrame {
    public static void main(String[] args) {
        new TestImageIcon();
    }

    public TestImageIcon() {
        // 获取图片的地址
        URL url = TestImageIcon.class.getResource("tx.jpg");
        JLabel jLabel = new JLabel("InageIcon");

        ImageIcon imageIcon = new ImageIcon(url);
        jLabel.setIcon(imageIcon);
        jLabel.setHorizontalAlignment(SwingConstants.CENTER);

        Container container = getContentPane();
        container.add(jLabel);

        setVisible(true);
        setBounds(100, 100, 200, 200);
        setDefaultCloseOperation(WindowConstants.EXIT_ON_CLOSE);
    }
}
```

- **面板**
  - JPanel

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

public class TestJPanel extends JFrame {
    public static void main(String[] args) {
        new TestJPanel();
    }

    public TestJPanel() {
        Container container = getContentPane();

        container.setLayout(new GridLayout(2, 1, 10, 10)); // 后面的参数是间距


        JPanel jPanel1 = new JPanel(new GridLayout(1, 3));
        JPanel jPanel2 = new JPanel(new GridLayout(1, 2));
        JPanel jPanel3 = new JPanel(new GridLayout(2, 1));
        JPanel jPanel4 = new JPanel(new GridLayout(3, 2));

        jPanel1.add(new JButton("1"));
        jPanel1.add(new JButton("1"));
        jPanel1.add(new JButton("1"));
```

```
22            jPanel2.add(new JButton("2"));
23            jPanel2.add(new JButton("2"));
24            jPanel3.add(new JButton("3"));
25            jPanel3.add(new JButton("3"));
26            jPanel4.add(new JButton("4"));
27            jPanel4.add(new JButton("4"));
28            jPanel4.add(new JButton("4"));
29            jPanel4.add(new JButton("4"));
30            jPanel4.add(new JButton("4"));
31            jPanel4.add(new JButton("4"));
32
33            container.add(jPanel1);
34            container.add(jPanel2);
35            container.add(jPanel3);
36            container.add(jPanel4);
37
38            setVisible(true);
39            setSize(500, 500);
40            setDefaultCloseOperation(WindowConstants.EXIT_ON_CLOSE);
41        }
42    }
```

效果:



- JScrollPanel

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

public class TestJScrollPanel extends JFrame {
    public static void main(String[] args) {
        new TestJScrollPanel();
    }

    public TestJScrollPanel() {
        Container container = getContentPane();

        // 文本域
        JTextArea jTextArea = new JTextArea(20, 50);
        jTextArea.setText("欢迎使用本软件！");

        JScrollPane jScrollPane = new JScrollPane(jTextArea);

        container.add(jScrollPane);

        setVisible(true);
        setBounds(100, 100, 300, 350);
        setDefaultCloseOperation(WindowConstants.EXIT_ON_CLOSE);
    }
}
```

效果：

按钮

- 图片按钮

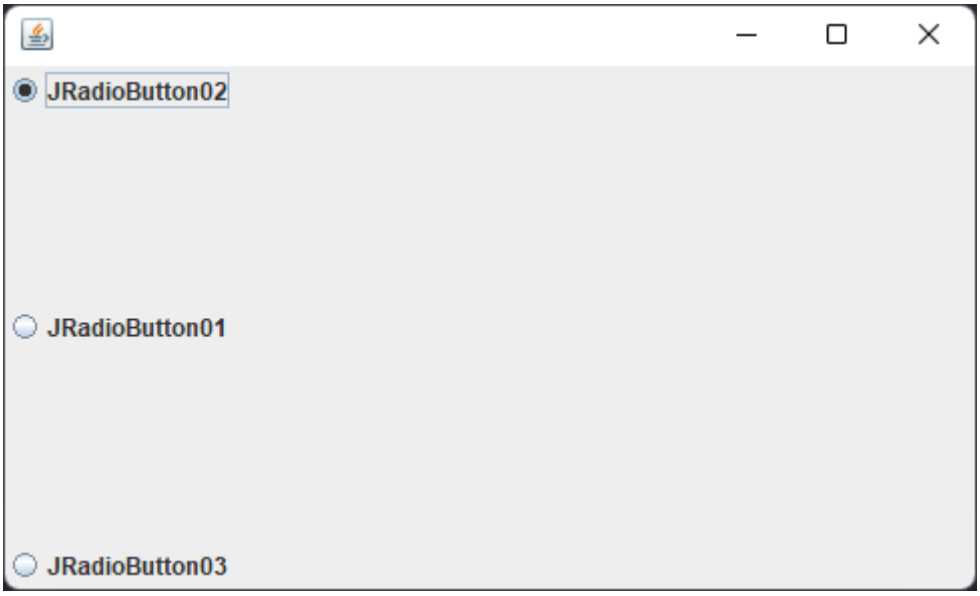
```
import javax.swing.*;
import java.awt.*;
import java.net.URL;

public class TestJButton extends JFrame {
    public static void main(String[] args) {
        new TestJButton();
    }

    public TestJButton() {
        Container container = getContentPane();
        // 将一个图片变为图标
        URL url = TestJButton.class.getResource("tx.jpg");
        ImageIcon imageIcon = new ImageIcon(url);

        // 把图标放在按钮上
        JButton jButton = new JButton();
        jButton.setIcon(imageIcon);
        jButton.setToolTipText("图片按钮");

        container.add(jButton);

        setVisible(true);
        setSize(500, 300);
        setDefaultCloseOperation(WindowConstants.EXIT_ON_CLOSE);
    }
}
```

- 单选按钮

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

public class TestJButton extends JFrame {
    public static void main(String[] args) {
        new TestJButton();
    }

    public TestJButton() {
        Container container = getContentPane();

        // 单选框
        JRadioButton jRadioButton01 = new JRadioButton("JRadioButton01");
        JRadioButton jRadioButton02 = new JRadioButton("JRadioButton02");
```

```
15          JRadioButton jRadioButton03 = new JRadioButton("JRadioButton03");
16
17          // 单选框只能选择一个，所以一般会进行分组，一个组中只能选择一个
18          ButtonGroup group = new ButtonGroup();
19          group.add(jRadioButton01);
20          group.add(jRadioButton02);
21          group.add(jRadioButton03);
22
23          container.add(jRadioButton01, BorderLayout.CENTER);
24          container.add(jRadioButton02, BorderLayout.NORTH);
25          container.add(jRadioButton03, BorderLayout.SOUTH);
26
27          setVisible(true);
28          setSize(500, 300);
29          setDefaultCloseOperation(WindowConstants.EXIT_ON_CLOSE);
30      }
31  }
```

效果：



- 复选按钮

```
1   import javax.swing.*;
2   import java.awt.*;
3
4   public class TestJButton extends JFrame {
5       public static void main(String[] args) {
6           new TestJButton();
7       }
8
9       public TestJButton() {
10          Container container = getContentPane();
11
12          // 复选框
13          JCheckBox checkBox01 = new JCheckBox("checkBox01");
```

```
14            JCheckBox checkBox02 = new JCheckBox("checkBox02");
15
16            container.add(checkBox01, BorderLayout.NORTH);
17            container.add(checkBox02, BorderLayout.SOUTH);
18
19            setVisible(true);
20            setSize(500, 300);
21            setDefaultCloseOperation(WindowConstants.EXIT_ON_CLOSE);
22        }
23    }
```

效果:

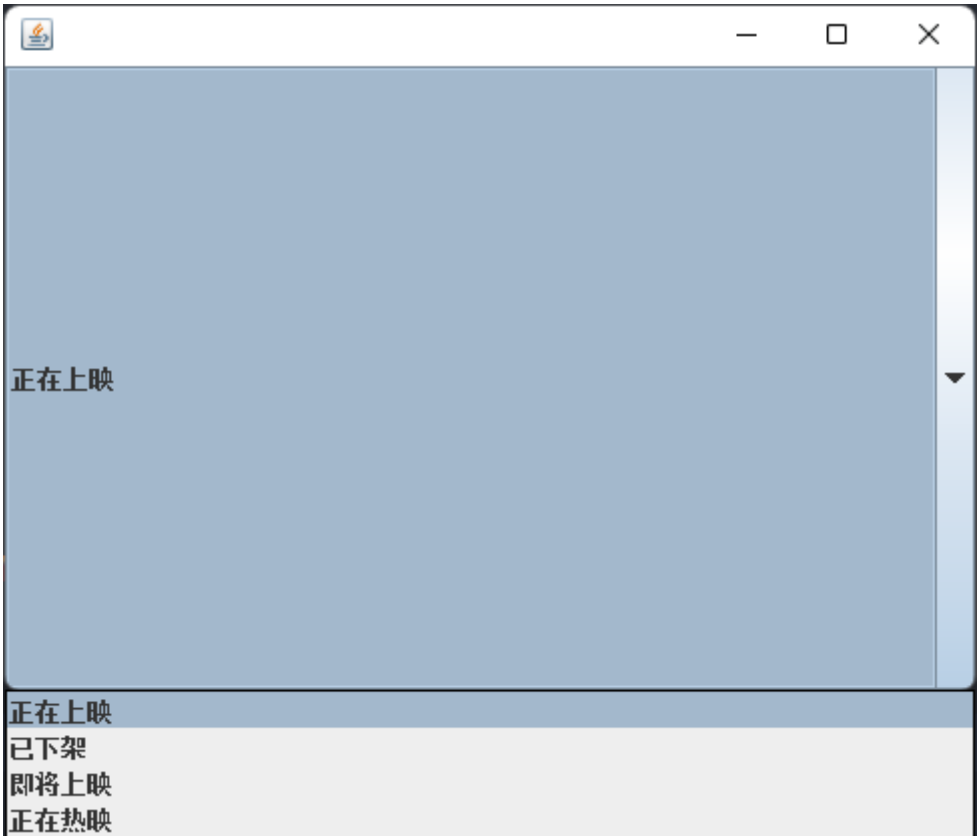
• 列表

- 下拉框

```
1   import javax.swing.*;
2   import java.awt.*;
3
4   public class TestComboBox extends JFrame {
5       public static void main(String[] args) {
6           new TestComboBox();
7       }
8
9       public TestComboBox() {
10          Container container = getContentPane();
11
12          JComboBox comboBox = new JComboBox();
13
14          comboBox.addItem("正在上映");
15          comboBox.addItem("已下架");
16          comboBox.addItem("即将上映");
17          comboBox.addItem("正在热映");
```

```
18
19          container.add(comboBox);
20
21          setVisible(true);
22          setSize(500, 350);
23          setDefaultCloseOperation(WindowConstants.EXIT_ON_CLOSE);
24      }
25  }
```

效果:

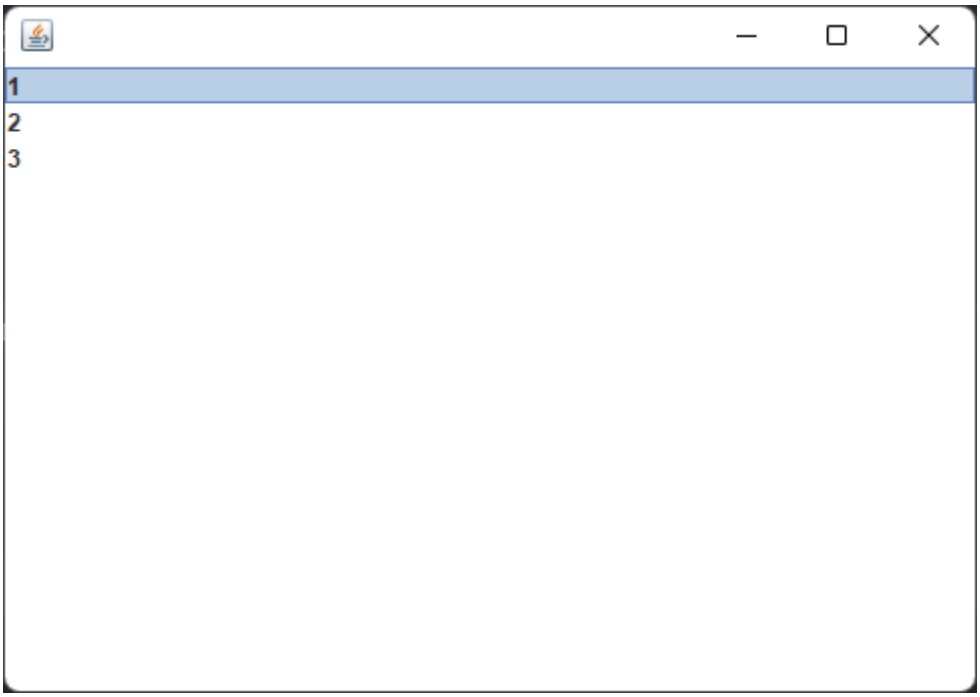


- 列表框

```
1   import javax.swing.*;
2   import java.awt.*;
3
4   public class TestComboBox extends JFrame {
5       public static void main(String[] args) {
6           new TestComboBox();
7       }
8
9       public TestComboBox() {
10          Container container = getContentPane();
11
12          // 生成列表的内容
13          String[] contents = {"1", "2", "3"};
14
15          JList list = new JList(contents);
16
```

```
            container.add(list);

            setVisible(true);
            setSize(500, 350);
            setDefaultCloseOperation(WindowConstants.EXIT_ON_CLOSE);
        }
    }
```

效果：



- 应用场景
    - 选择地区，或者一些单个选项
    - 列表，展示信息，一般是动态扩容！

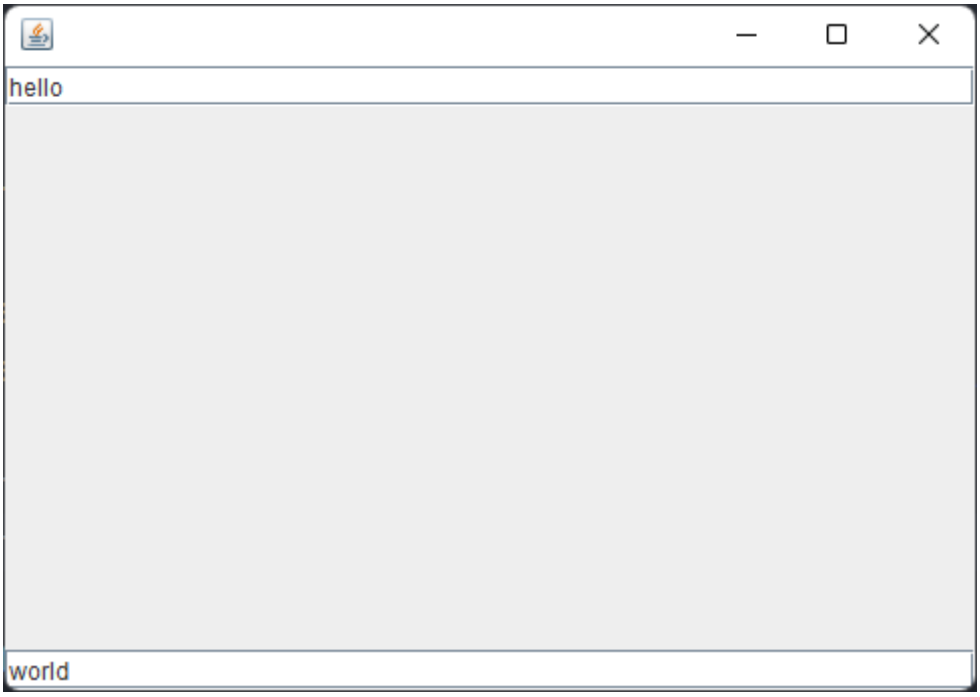
文本框

- 文本框

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

public class TestText extends JFrame {
    public static void main(String[] args) {
        new TestText();
    }

    public TestText() {
        Container container = getContentPane();

        JTextField textField = new JTextField("hello");
```

```
13          JTextField textField2 = new JTextField("world", 20);
14
15          container.add(textField, BorderLayout.NORTH);
16          container.add(textField2, BorderLayout.SOUTH);
17
18          setVisible(true);
19          setSize(500, 350);
20          setDefaultCloseOperation(WindowConstants.EXIT_ON_CLOSE);
21      }
22  }
```

效果:

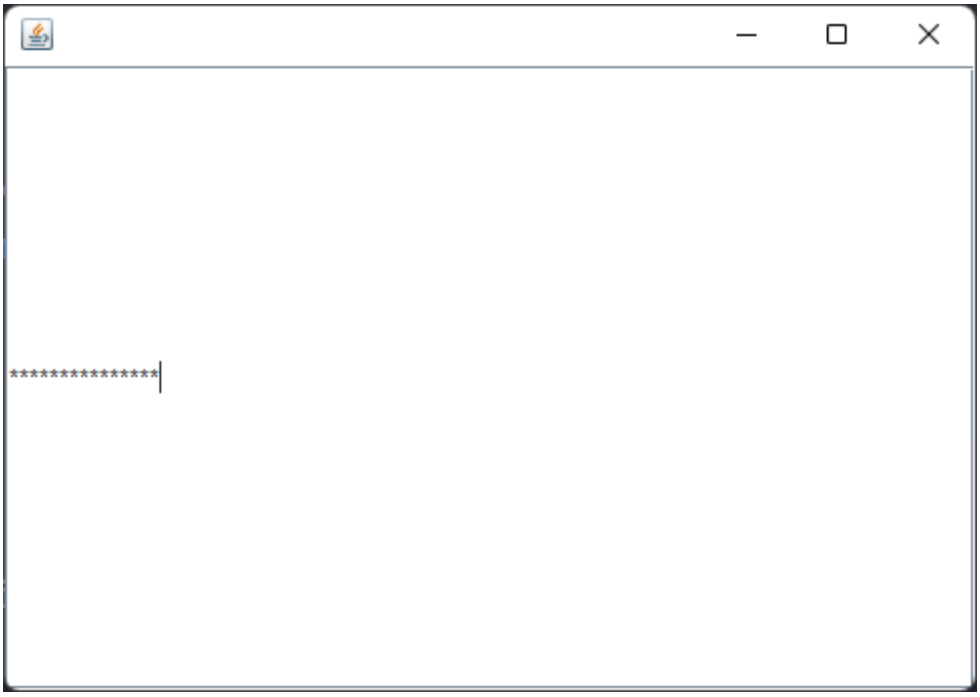


- 密码框

```
1   import javax.swing.*;
2   import java.awt.*;
3
4   public class TestText extends JFrame {
5       public static void main(String[] args) {
6           new TestText();
7       }
8
9       public TestText() {
10          Container container = getContentPane();
11
12          JPasswordField field = new JPasswordField();
13          field.setEchoChar('*');
14
15          container.add(field);
16
17          setVisible(true);
```

```
            setSize(500, 350);
            setDefaultCloseOperation(WindowConstants.EXIT_ON_CLOSE);
        }
    }
```

效果:



- 文本域

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

public class TestJScrollPanel extends JFrame {
    public static void main(String[] args) {
        new TestJScrollPanel();
    }

    public TestJScrollPanel() {
        Container container = getContentPane();

        // 文本域
        JTextArea jTextArea = new JTextArea(20, 50);
        jTextArea.setText("欢迎使用本软件! ");

        JScrollPane jScrollPane = new JScrollPane(jTextArea);

        container.add(jScrollPane);

        setVisible(true);
        setBounds(100, 100, 300, 350);
        setDefaultCloseOperation(WindowConstants.EXIT_ON_CLOSE);
    }
```

```
24    }
```

效果：

贪吃蛇

- StartGame.java

```
1   import javax.swing.*;
2
3   // 游戏的主启动类
4   public class StartGame {
5       public static void main(String[] args) {
6           JFrame frame = new JFrame();
7
8           frame.setVisible(true);
9           frame.setDefaultCloseOperation(WindowConstants.EXIT_ON_CLOSE);
10          frame.setBounds(10, 10, 900, 720);
11          frame.setResizable(false);
12
13          frame.add(new GamePanel());
14      }
15  }
```

- GamePanel.java

```
1   import javax.swing.*;
2   import java.awt.*;
```

```java
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.awt.event.KeyEvent;
import java.awt.event.KeyListener;
import java.util.Random;

// 游戏的面板
public class GamePanel extends JPanel implements KeyListener,
ActionListener {

    // 定义蛇的数据结构
    // 蛇的长度
    int length;
    // 蛇的X坐标
    int[] snakeX = new int[600];
    // 蛇的Y坐标
    int[] snakeY = new int[500];
    // 初始方向
    String fx;
    // 游戏当前的状态
    boolean isStart = false;

    // 游戏是否失败
    boolean isFail = false;

    // 定时器 以毫秒为单位
    Timer timer = new Timer(100, this);

    // 食物的坐标
    int foodX;
    int foodY;
    Random random = new Random();

    // 成绩
    int score;

    // 构造器
    public GamePanel() {
        init();
        // 获得焦点和键盘事件
        this.setFocusable(true);
        this.addKeyListener(this);
        // 游戏一开始定时器就启动
        timer.start();
        score = 0;
    }

    public void init() {
        length = 3;
        // 脑袋的坐标
        snakeX[0] = 100;
```

```
53            snakeY[0] = 100;
54            // 第一个身体的坐标
55            snakeX[1] = 75;
56            snakeY[1] = 100;
57            // 第二个身体的坐标
58            snakeX[2] = 50;
59            snakeY[2] = 100;
60
61            // 初始方向：向右
62            fx = "R";
63
64            foodX = 25 + 25 * random.nextInt(34);
65            foodY = 75 + 25 * random.nextInt(24);
66        }
67
68        // 绘制面板，游戏中的所有东西都用这个画笔来画
69        @Override
70        protected void paintComponent(Graphics g) {
71            super.paintComponent(g); // 清屏
72            // 绘制静态的面板
73            this.setBackground(Color.white);
74            // 头部广告栏
75            Data.header.paintIcon(this, g, 25, 11);
76            // 默认的游戏界面
77            g.fillRect(25, 75, 850, 600);
78
79            // 画积分
80            g.setColor(Color.white);
81            g.setFont(new Font("微软雅黑", Font.BOLD, 18));
82            g.drawString("长度" + length, 750, 35);
83            g.drawString("分数" + score, 750, 50);
84
85            // 画食物
86            Data.food.paintIcon(this, g, foodX, foodY);
87
88            // 把小蛇画上去
89            if (fx.equals("R")) {
90                Data.right.paintIcon(this, g, snakeX[0], snakeY[0]);
91            } else if (fx.equals("L")) {
92                Data.left.paintIcon(this, g, snakeX[0], snakeY[0]);
93            } else if (fx.equals("U")) {
94                Data.up.paintIcon(this, g, snakeX[0], snakeY[0]);
95            } else if (fx.equals("D")) {
96                Data.down.paintIcon(this, g, snakeX[0], snakeY[0]);
97            }
98
99            for (int i = 0; i < length; i++) {
100               Data.body.paintIcon(this, g, snakeX[i], snakeY[i]);
101           }
102
103           // 游戏状态
```

```
104             if (!isStart) {
105                 g.setColor(Color.white);
106                 // 设置字体
107                 g.setFont(new Font("微软雅黑", Font.BOLD, 40));
108                 g.drawString("按下空格开始游戏", 300, 300);
109             }
110 
111             if (isFail) {
112                 g.setColor(Color.red);
113                 // 设置字体
114                 g.setFont(new Font("微软雅黑", Font.BOLD, 40));
115                 g.drawString("游戏失败，按下空格重新开始", 300, 300);
116             }
117         }
118 
119         // 键盘监听事件
120         @Override
121         public void keyTyped(KeyEvent e) {
122 
123         }
124         @Override
125         public void keyPressed(KeyEvent e) {
126             int keyCode = e.getKeyCode();
127             if (keyCode == KeyEvent.VK_SPACE) {
128                 if (isFail) {
129                     // 重新开始
130                     isFail = false;
131                     init();
132                 } else {
133                     isStart = !isStart;
134                     repaint();
135                 }
136             }
137 
138             // 小蛇移动
139             if (keyCode == KeyEvent.VK_UP) {
140                 fx = "U";
141             } else if (keyCode == KeyEvent.VK_DOWN) {
142                 fx = "D";
143             } else if (keyCode == KeyEvent.VK_LEFT) {
144                 fx = "L";
145             } else if (keyCode == KeyEvent.VK_RIGHT) {
146                 fx = "R";
147             }
148         }
149 
150         @Override
151         public void keyReleased(KeyEvent e) {
152 
153         }
154         // 事件监听
```

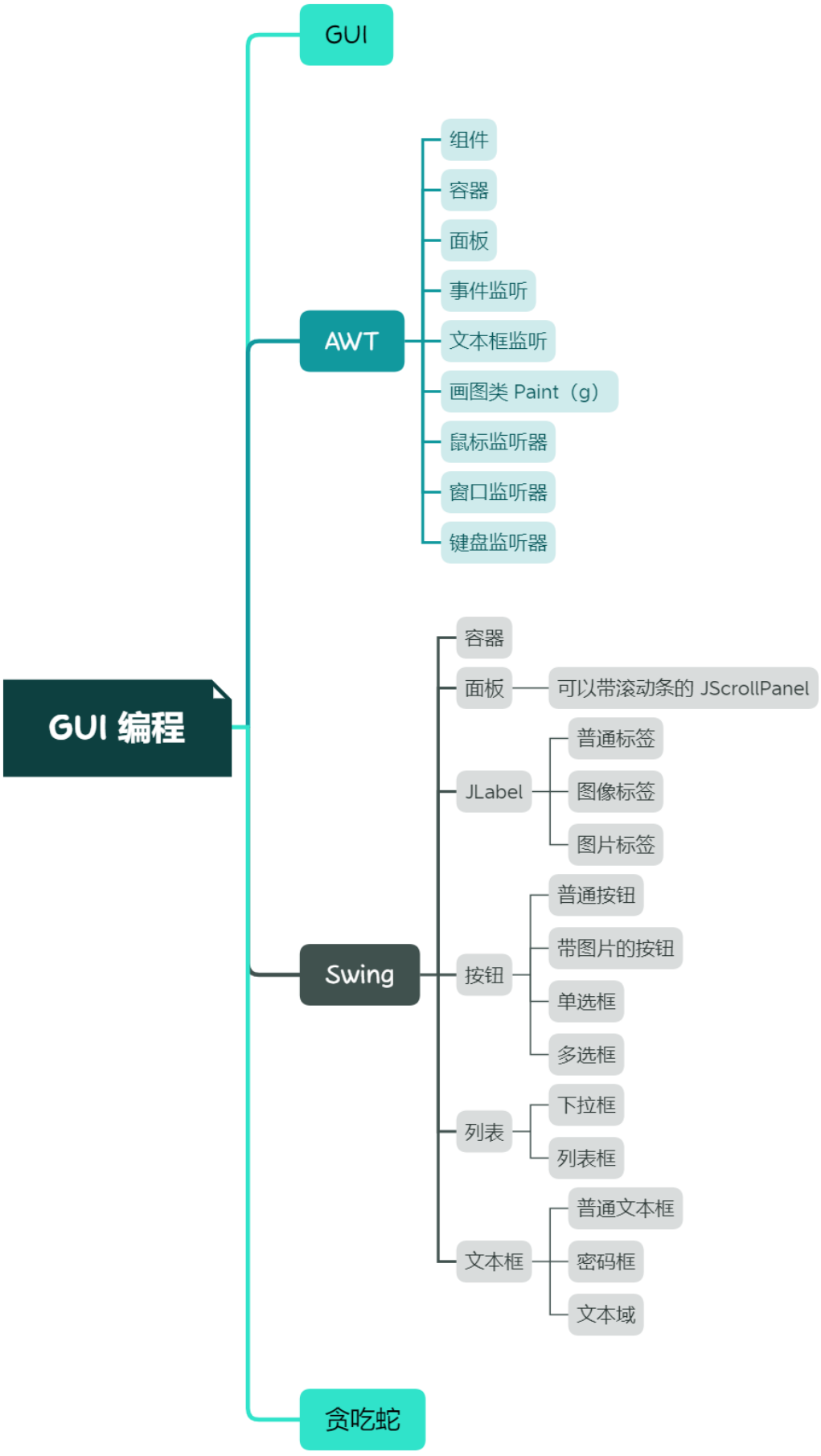
```
155         @Override
156         public void actionPerformed(ActionEvent e) {
157             // 如果游戏是开始状态，就让小蛇动起来
158             if (isStart && !isFail) {
159 
160                 // 吃食物
161                 if (snakeX[0] == foodX && snakeY[0] == foodY) {
162                     // 长度加1
163                     length++;
164                     // 分数加10
165                     score += 10;
166                     // 再次随机食物
167                     foodX = 25 + 25 * random.nextInt(34);
168                     foodY = 75 + 25 * random.nextInt(24);
169                 }
170 
171                 // 移动
172                 for (int i = length - 1; i > 0; i--) {
173                     // 向前移动一节
174                     snakeX[i] = snakeX[i - 1];
175                     snakeY[i] = snakeY[i - 1];
176                 }
177                 // 走向
178                 if (fx.equals("R")) {
179                     snakeX[0] = snakeX[0] + 25;
180                     // 边界判断
181                     if (snakeX[0] > 850) {
182                         snakeX[0] = 25;
183                     }
184                 } else if (fx.equals("L")) {
185                     snakeX[0] = snakeX[0] - 25;
186                     // 边界判断
187                     if (snakeX[0] < 25) {
188                         snakeX[0] = 850;
189                     }
190                 } else if (fx.equals("U")) {
191                     snakeY[0] = snakeY[0] - 25;
192                     // 边界判断
193                     if (snakeY[0] < 75) {
194                         snakeY[0] = 650;
195                     }
196                 } else if (fx.equals("D")) {
197                     snakeY[0] = snakeY[0] + 25;
198                     // 边界判断
199                     if (snakeY[0] > 650) {
200                         snakeY[0] = 75;
201                     }
202                 }
203 
204                 // 失败的判定：撞到自己
205                 for (int i = 1; i < length; i++) {
```

```
                    if (snakeX[0] == snakeY[i] && snakeY[0] == snakeY[i]) {
                        isFail = true;
                    }
                }

                repaint();
            }
            // 定时器开启
            timer.start();
        }
    }
```

- Data.java

```
import javax.swing.*;
import java.net.URL;

// 数据中心
public class Data {
    public static URL headerUrl =
Data.class.getResource("static/header.png");
    public static URL upUrl = Data.class.getResource("static/up.png");
    public static URL downUrl = Data.class.getResource("static/down.png");
    public static URL lrftUrl = Data.class.getResource("static/left.png");
    public static URL rightUrl = Data.class.getResource("static/right.png");

    public static URL bodyUrl = Data.class.getResource("static/body.png");
    public static URL foodUrl = Data.class.getResource("static/food.png");

    public static ImageIcon header = new ImageIcon(headerUrl);
    public static ImageIcon up = new ImageIcon(upUrl);
    public static ImageIcon down = new ImageIcon(downUrl);
    public static ImageIcon left = new ImageIcon(lrftUrl);
    public static ImageIcon right = new ImageIcon(rightUrl);
    public static ImageIcon body = new ImageIcon(bodyUrl);
    public static ImageIcon food = new ImageIcon(foodUrl);
}
```

GUI 编程总结


GUI 编程
GUI
AWT
组件
容器
面板
事件监听
文本框监听
画图类 Paint（g）
鼠标监听器
窗口监听器
键盘监听器
Swing
容器
面板
可以带滚动条的 JScrollPanel
JLabel
普通标签
图像标签
图片标签
按钮
普通按钮
带图片的按钮
单选框
多选框
列表
下拉框
列表框
文本框
普通文本框
密码框
文本域
贪吃蛇