GUI

概述

> GUI(Graphical User Interface)-图形化用户界面。用户和程序之间可以通过GUI能方便友好地进行交互。在Java语言中，JFC(Java Foundation Classed)是开发GUI的API集，它主要包含以下几个部分：

- AWT(抽象窗口工具包)：Java开发用户界面最初的工具包，是建立JFC的主要基础。
- Swing组件：建立在AWT之上，新的、功能更强大的图形组件包

AWT框架

Component类

- `Component`类是最核心的类,它是构成Java图形用户界面的基础，大部分组件都是由该类派生出来的。
- `Component`类主要由基本组件和容器`Container`组件组成。
- 容器`Container`组件主要分为：`Window`容器和`Panel`容器

Frame容器和Panel容器

Frame容器

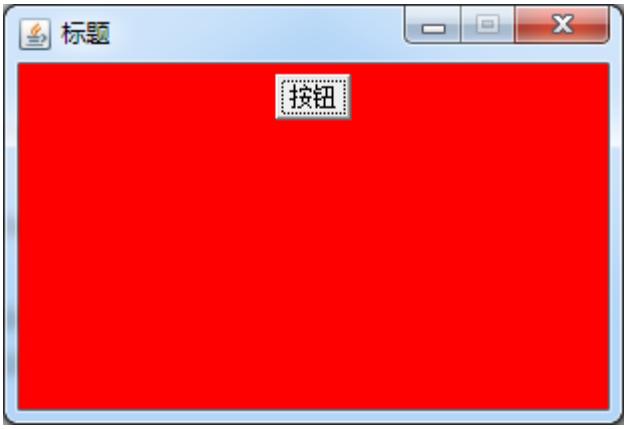
- Window是能独立存在的容器，它有一个子类Frame,它是一个带有标题和缩放角的窗口。
- Frame有一个构造方法`Frame(Stringtitle)`，可以设置标题
- 可以通过`add()`方法，在Frame容器中加入其他的组件。
- Frame容器有默认的布局管理器：`BorderLayout`。
- Frame被创建后，是不可见的，需要执行`setVisible(true)`设置可见

Panel容器

- Panel只能存在于其他的容器(Window或其子类)中才能显示出来。
- 通过Panel的默认构造方法Panel()可以创建一个Panel容器
- 容器不但能容纳组件，还能容纳其它容器，通过容器的嵌套可以制作出复杂的布局。

例子

```java
class MyFrame extends Frame {

    private static final long serialVersionUID = 6338484641644928054L;

    public MyFrame(String title) {
        super(title);
        init();
    }

    private void init() {
        // 设置大小
        this.setSize(300, 200);
        this.setBackground(Color.BLUE);
        // 设置窗口居中
        this.setLocationRelativeTo(null);
        // 禁止重置大小
        this.setResizable(false);
        // 创建一个panel
        Panel panel = new Panel();
        // 设置panel背景色为红色
        panel.setBackground(Color.RED);
        // 将panel容器添加到frame容器
        this.add(panel);
        // 创建一个button
        Button button = new Button("按钮");
        // 将button添加到panel容器
        panel.add(button);
        button.addActionListener(new ActionListener() {

            @Override
            public void actionPerformed(ActionEvent e) {
                // panel背景色在Red和Green中切换
                Color color = panel.getBackground();
                color = color.equals(Color.RED) ? Color.GREEN : Color.RED;
                panel.setBackground(color);
            }
        });
        // 设置可见性
        this.setVisible(true);
        // 关闭窗口
        this.addWindowListener(new WindowAdapter() {
            @Override
            public void windowClosing(WindowEvent e) {
                System.exit(0);
            }
        });
    }

}
```



布局管理器

概述

> 容器内可以存放不同的组件，容器内组件的摆放位置和大小有容器的布局管理器决定，接下来讲解一下4个布局管理器：

- FlowLayout：流布局管理器
- BorderLayout：边框布局管理器
- GridLayout：网格布局管理器
- CardLayout：卡片布局管理器

设置方法

```
public void setLayout(LayoutManager mgr)
```

FlowLayout

按照指定的对齐方式顺序摆放组件，若空间不够，则换行

构造方法

1. `FlowLayout()`：默认居中对齐
2. `FlowLayout(int align)`：指定对齐方式
3. `FlowLayout(int align, int hgap, int vgap)`：指定对齐方式，水平间距，垂直间距

例子

```
class FlowLayoutDemo extends Frame{

    private static final long serialVersionUID = -5581804027043150613L;

    public FlowLayoutDemo(String title) {
        super(title);
        init();
    }

    private void init() {
        //窗口大小
        this.setSize(300, 200);
        //居中
        this.setLocationRelativeTo(null);
        //设置布局管理器
        FlowLayout layout = new FlowLayout(FlowLayout.LEFT, 20, 40);
        this.setLayout(layout);
        //添加6个按钮
        this.add(new Button("btn1"));
        this.add(new Button("btn2"));
        this.add(new Button("btn3"));
        this.add(new Button("btn4"));
        this.add(new Button("btn5"));
        this.add(new Button("btn6"));
        //设置可见
        this.setVisible(true);
        this.addWindowListener(new WindowAdapter() {
            @Override
            public void windowClosing(WindowEvent e) {
                System.exit(0);
            }
        });
    }

}
```