# 1. 图形用户界面（GUI）

Java也可以开发图形用户界面（Graphics User Interface）的程序。使用图形界面程序更直观、更方便，所见即所得。不过我们学习的重点不在于图形用户界面程序，而是以后的互联网开发。因此这里是少量涉及，介绍某几个重要概念。不过更适合开发桌面应用程序的还是C、C++和.Net，效率高。

## 1.1 概述

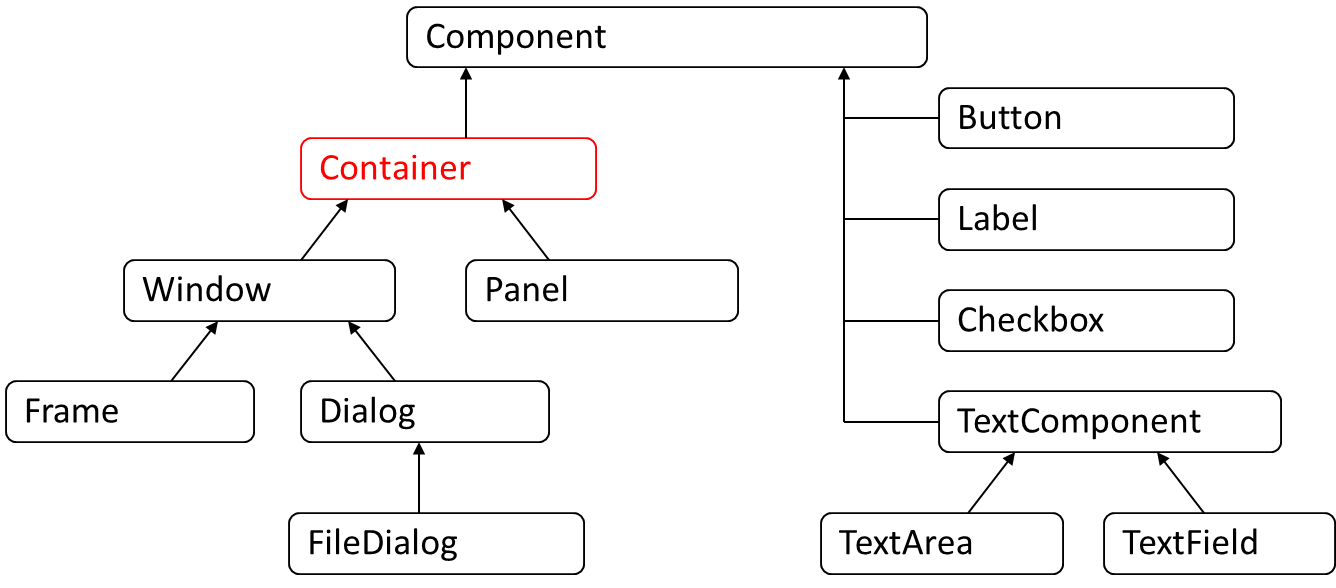
GUI中常用的类在下面的两个包中（这里不涉及JavaFX）：

（1）java.awt包：Abstract Window ToolKit (抽象窗口工具包)，通过调用本地系统方法实现功能，和操作系统关联较强，属重量级控件。

（2）javax.swing包：在AWT的基础上，建立的一套图形界面系统，其中提供了更多的组件，而且完全由Java实现。swing属轻量级控件，和操作系统无关。【所有带x的（javax）都是“扩展包”】。

由于我们重点不在于图形界面，因此只讲awt中的类，重要的在于其中的编程思想。

GUI的继承体系图：



简单了解一下。

## 1.2 一个简单的窗体

用Frame类创建一个窗体。用setTitle()设置标题，用setSize()设置窗体大小等等。都可在API中查看。注意窗体默认是不显示的，需要调用setVisible()，设置可见性为true才能看到窗口。具体内容见代码：

|  |
| --- |
| **package** com.zhang.test;  **import** java.awt.\*;  **public class** Demo {  **public static void** main(String[] args) {  *// 新建窗体对象* Frame f1 = **new** Frame();  f1.setTitle(**"第一个窗体"**); *// 设置标题* f1.setSize(400, 300); *// 设置窗体宽和高* f1.setLocation(400, 200); *// 设置窗体左上角坐标。* f1.setBackground(Color.***darkGray***); *// 设置背景颜色* f1.setVisible(**true**); *// 设置窗体显示* } } |

其中setLocation(x, y)设置窗体左上角在屏幕中的坐标，那么坐标原点就是屏幕的左上角，向右是x轴，向下是y轴。setBackground中设置颜色，用Color对象中提供的默认变量即可。

实际上，setSize()方法中还能传递Dimension对象（尺寸对象），此对象实际就是封装了宽高属性；setLocation()方法中还能传递Point对象(坐标点对象)，此对象实际就是封装了x和y两个坐标值。例如：

|  |
| --- |
| **package** com.zhang.test;  **import** java.awt.\*;  **public class** Demo {  **public static void** main(String[] args) {  *// 新建窗体对象* Frame f2 = **new** Frame();  f2.setTitle(**"第而个窗体"**);  f2.setSize(**new** Dimension(400, 300));  f2.setLocation(**new** Point(400, 200));  f2.setVisible(**true**);  } } |

还可以直接用setBounds方法一次性设置好窗体的大小和坐标。

|  |
| --- |
| **package** com.zhang.test;  **import** java.awt.\*;  **public class** Demo {  **public static void** main(String[] args) {  *// Frame的构造可以直接设置标题* Frame f3 = **new** Frame(**"这是第三个窗体"**);  *// 设置x, y, width, height* f3.setBounds(400, 200, 400, 300);  f3.setVisible(**true**);  } } |

因此有时可多看看API，找到更好用的方法。

简单的窗体创建过程：Frame f = new Frame();f.setVisible(true);在显示之前可以设置标题、尺寸等。

## 1.3 事件监听机制

运行上述代码时，发现点击窗口关闭按钮是没有反应的，只能在IDE中强制关闭程序。如果想让窗口关闭正常，那就就要进行事件处理。即要程序监听窗体关闭事件，并且程序要处理该事件。

在GUI中经常用到事件处理。事件处理有以下要点：

（1）拥有事件源，就是事件发生的地方，比如一个按钮。

（2）给事件源添加（注册）对某个事件的监听，这个事件是API定义好的，比如按钮点击事件、窗体关闭事件。

（3）给注册的监听事件编写事件处理，比如点击一个按钮要进行什么样的操作。

现在实现窗口的关闭。使用addWindowListener方法给事件源（即窗口）添加监听。此方法中传递一个WindowListener接口的实现类。此接口中就定义了窗体可以处理的各种事件。我们可以用匿名内部类。对于窗体关闭，我们只要处理WindowListener接口中的windowClosing事件，其他方法不需要实现。

|  |
| --- |
| **package** com.zhang.test;  **import** java.awt.\*; **import** java.awt.event.WindowEvent; **import** java.awt.event.WindowListener;  **public class** Demo {  **public static void** main(String[] args) {  *// Frame的构造可以直接设置标题* Frame f3 = **new** Frame(**"这是第三个窗体"**);  *// 设置x, y, width, height* f3.setBounds(400, 200, 400, 300);  f3.setVisible(**true**);  f3.addWindowListener(**new** WindowListener() {  @Override  **public void** windowOpened(WindowEvent e) {  }  @Override  **public void** windowClosing(WindowEvent e) {  *// 在此事件中关闭程序* System.*exit*(0);  }  @Override  **public void** windowClosed(WindowEvent e) {  }  @Override  **public void** windowIconified(WindowEvent e) {  }  @Override  **public void** windowDeiconified(WindowEvent e) {   }  @Override  **public void** windowActivated(WindowEvent e) {  }  @Override  **public void** windowDeactivated(WindowEvent e) {  }  });  } } |

上面就是在窗体关闭事件中直接退出程序。

## 1.4 适配器Adapter

上面写事件处理时，要实现接口所有的方法，太麻烦。有什么方法能只要重写接口中的一个方法呢？这就通过适配器来实现。

实现适配器的步骤：

（1）首先定义好接口（上述程序中，此接口已经由JDK定义好）；

（2）用一个抽象类来实现此接口，实现接口的所有方法。注意是实现，里面可以不写什么，但是一定要实现。那么，当需要用到此接口的实现时，直接用此抽象类的实现类对象即可。此时，此抽象类的实现类需要重写什么方法，就重写什么方法，因为虽然是抽象类，但是他们都“实现”了，因此只要重写需要的方法即可。这样就实现了适配器，该抽象类就是适配器。

实际上对于WindowListener，JDK也写好了适配器WindowAdapter。

例子：

|  |
| --- |
| **package** com.zhang.test;  **import** java.awt.\*; **import** java.awt.event.WindowAdapter; **import** java.awt.event.WindowEvent;  **public class** Demo {  **public static void** main(String[] args) {  *// Frame的构造可以直接设置标题* Frame f3 = **new** Frame(**"这是第三个窗体"**);  *// 设置x, y, width, height* f3.setBounds(400, 200, 400, 300);  f3.setVisible(**true**);  f3.addWindowListener(**new** WindowAdapter() {  @Override  **public void** windowClosing(WindowEvent e) {  System.*exit*(0);  }  });  } } |

当然，基本上JDK也都提供了事件处理配套的适配器，除非此事件处理接口只有一个方法。

其他例子：用户点击一下按钮，就改变窗体的背景颜色（红和绿）。

|  |
| --- |
| **package** com.zhang.test;  **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** Demo {  **public static void** main(String[] args) {  Frame f = **new** Frame(**"背景变色"**);  f.setBounds(400, 300, 400, 200);  *// 窗口关闭事件* f.addWindowListener(**new** WindowAdapter() {  @Override  **public void** windowClosing(WindowEvent e) {  System.*exit*(0);  }  });  f.setBackground(Color.***red***);  f.setVisible(**true**);  *//设置布局方式为流式布局，否则下面的按钮会占据整个区域（了解即可）* f.setLayout(**new** FlowLayout());   *// 添加按钮* Button btn = **new** Button(**"Change Background Color"**);  f.add(btn); *// 向窗体中添加按钮  // 按钮点击事件 这个ActionListener就没有适配器* btn.addActionListener(**new** ActionListener() {  @Override  **public void** actionPerformed(ActionEvent e) {  **if**(f.getBackground() == Color.***red***) {  *// 红色就绿* f.setBackground(Color.***green***);  } **else** {  *// 变红* f.setBackground(Color.***red***);  }  }  });  } } |

图形界面到此为止。不建议深入研究。主要了解其中思想