SWING教程

极客学院出版

前言

JAVA 提供了一组丰富的库来用平台独立的方式创建图形用户界面。在本文中我们将学习 SWING GUI 控件。

读者

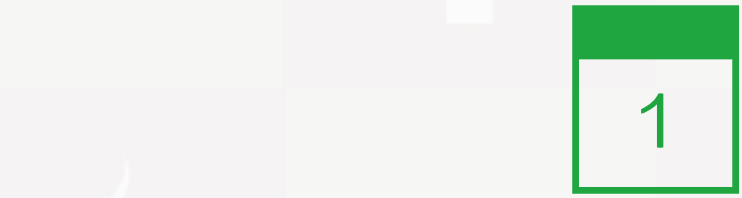
本教程是为愿意学习 JAVA GUI 来用简易的步骤编程的软件专业人员而设计的。本教程将会给你带来对 JAVA GUI 编程的概念有更好的理解，并且在完成本教程之后，你将会把自己的专业知识提升到一个较高的水平。

预备知识

在继续学习本教程之前，你应该对 Java 编程语言、文本编辑器和执行程序等有一个基本的了解。

更新日期	更新内容
2015-07-24	SWING 教程

目录



SWING – 概述

Swing API 是一组可扩展的 GUI 组件，用来创建基于 JAVA 的前端/ GUI 应用程序。它是建立在 AWT API 之上，并且作为 AWT API 的替代者，因为它的几乎每一个控件都对应 AWT 控件。 Swing 组件遵循模型 – 视图 – 控制器架构来满足下面的准则。

- 一个单一的 API 足够支持多种外观和风格。
- API 是模拟驱动的，这样最高层级的 API 不需要有数据。
- API 是使用 Java Bean 模式的，这样 Builder Tools 和 IDE 可以为开发者提供更好的服务来使用它。

MVC 架构

Swing API 架构用下列的方式来遵循基于松散的 MVC 架构。

- 模型表示组件的数据。
- 视图表示组件数据的可视化表示形式。
- 控制器接受用户在视图上的输入，并且在组件的数据上反映变化。
- Swing 组件把模型作为一个单独的元素，并且把视图和控制器部分组合成用户界面的元素。使用这种方式，Swing 具有可插拔的外观与风格架构。

Swing 特点

- **轻量级** —— Swing 组件是独立的本地操作系统的 API，因为 Swing API 控件通常采用纯 JAVA 代码而不是采用底层的操作系统调用来呈现。
- **丰富的控件** —— Swing 提供了一组丰富的先进的控件，如树，页签窗体，滑动条，颜色选择器，表格控件
- **高级自定义** —— Swing 控件可以用非常简单的方法来自定义，因为可视化外观是独立于内部表示的。
- **可插拔的外观和风格** —— 基于 Swing 的 GUI 应用程序的外观和风格可以在运行时根据有效的值改变。

2

SWING – 环境安装

本节将指导你如何在你的机器上下载和设置 Java。请按照以下步骤来设置环境。

在链接 Download Java 上, Java SE 免费提供的。所以你根据你的操作系统下载一个版本。

在你的机器上，按照说明下载 java 和运行 .exe 来安装 Java。一旦在你的机器上安装了 Java，你将需要设置环境变量来指向正确的安装目录：

设置 windows 2000/XP 的路径：

假设你已经在 *c:\Program Files\java\jdk* 目录上安装了 Java：

- 右击 ‘我的电脑’，并且选择 ‘属性’。
- 点击 ‘高级’ 标签下的 ‘环境变量’ 按钮。
- 现在更改 ‘路径’ 变量，以便它也包含 Java 可执行文件的路径。例如，如果路径当前设置为 `C:\WINDOWS\SYSTEM32`，然后更改你的路径为 `C:\WINDOWS\SYSTEM32;c:\Program Files\java\jdk\bin`。

设置 windows 95/98/ME 的路径：

假设你已经在 *c:\Program Files\java\jdk* 目录上安装了 Java：

- 编辑 `C:\autoexec.bat` 文件，并且在最后添加下行： `SET PATH=%PATH%;C:\Program Files\java\jdk\bin`

设置 Linux，UNIX，Solaris，FreeBSD 的路径：

环境变量路径应该设置为指向 java 二进制文件已经安装的位置。如果你在这方面遇到困难，参考你的 shell 文档。

例如，如果你使用 *bash* 作为你的 shell，然后你将添加下行到最后 `.bashrc: export PATH=/path/to/java:$PATH`

流行的 Java 编辑器：

为了编写 Java 程序，你将需要一个文本编辑器。在市场上有甚至更复杂的可用 IDE。但是现在，你可以考虑下列之一：

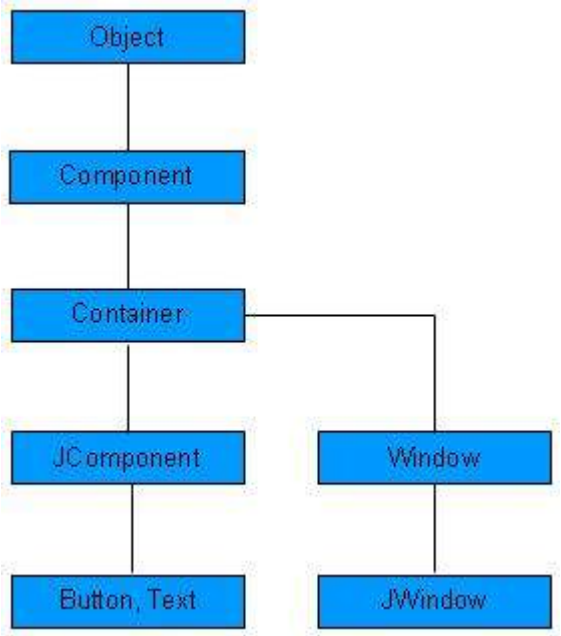
- Notepad： 在 Windows 机器上，你可以使用任何简单的文本编辑器，如 Notepad（本教程推荐），Text Pad。
- Netbeans：是一个开源而且免费的 Java IDE，它可以从 http://www.netbeans.org/index.html 下载。
- Eclipse： 也是一个 Java IDE，它是由 eclipse 开源社区开发的，可以从 http://www.eclipse.org/ 下载。

3

SWING – 控件

每个用户界面考虑有以下三个主要方面：

- **UI 元素：** 有用户最终看到并且与之交互的核心视觉元素。GWT 提供了一个大量的广泛使用和常见的元素列表，本教程我们将涉及从基本到复杂的变化。
- **布局：** 他们定义应该如何在屏幕上组织 UI 元素，并且提供一个最终的外观和风格给 GUI（图形用户界面）。这部分将在布局这一章涉及。
- **行为：** 当用户与 UI 元素交互时，这些事件发生。这部分将在事件处理这一章涉及。



每个 SWING 控件从下列组件类的等级继承属性。

序号	类 & 描述
1	Component Container 是 SWING 的非菜单用户界面控件的一个抽象基类。组件代表一个用图形表示的对象
2	Container Container 是一个组件，它可以包含其他 SWING 组件。
3	JComponent JComponent 是一个所有 swing UI 组件的基类。为了使用继承自 JComponent 的一个 swing 组件，组件必须是一个包容层次结构，它的根是一个顶层的 Swing 容器。

SWING UI 元素:

下列是当使用 SWING 来设计 GUI 时常用的控件列表。

Sr. No.	控件 & 描述
1	JLabel JLabel 对象是一个在容器中放置文本的组件。
2	JButton 该类创建一个有标签的的按钮。
3	JColorChooser JColorChooser 提供一个控制面板，设计允许用户操作和选择颜色。
4	JCheck Box JCheckBox 是一个图形化的组件，它的状态要么是 on（true）要么是 off（false）。
5	JRadioButton JRadioButton 类是一个图形化的组件，在一个组中，它的状态要么是 on（true）要么是off（false）。
6	JList JList 组件呈现给用户一个滚动的文本项列表。
7	JComboBox JComboBox 组件呈现给用户一个显示菜单的选择。
8	JTextField JTextField 对象是一个文本组件，它允许编辑单行文本。
9	JPasswordField JPasswordField 对象是一个专门用于密码输入的文本组件。
10	JTextArea JTextArea 对象是一个文本组件，它允许编辑多行文本。
11	ImageIcon ImageIcon 控件是一个图标界面的实现，它从图像描绘图标
12	JScrollbar Scrollbar 控件代表一个滚动条组件，为了让用户从值的范围中选择。
13	JOptionPane JOptionPane 提供了一组提示用户输入值的标准对话框，或者通知他们其他东西。
14	JFileChooser JFileChooser 控件代表一个对话框窗口，用户可以从该对话框窗口选择一个文件。
15	JProgressBar 随着任务完成的进展，进度条显示任务完成的百分比。
16	JSlider JSlider 让用户在有界区间内通过滑动旋钮图形化地选择一个值。
17	JSpinner JSpinner 是一个单行输入字段，它让用户从一个有序序列中选择一个数字或者一个对象值。

4

SWING 事件处理

什么是事件?

改变对象的状态被称为事件，即事件描述源的状态变化。事件产生用户与图形用户界面组件交互的结果。例如，点击一个按钮，移动鼠标，通过键盘输入一个字符，从列表中选择一个项目，滚动页面都是导致一个事件发生的活动。

事件的类型

事件可以大致分为两类::

- **前台事件** – 这些事件需要用户的直接互动。它们是由在图形用户界面中人与图形组件交互的结果而产生的。例如，点击一个按钮，移动鼠标，通过键盘输入一个字符，从列表中选择一个项目，滚动页面等。
- **后台事件** – 这些事件需要最终用户的交互，它们被称为后台事件。操作系统的中断，硬件或软件故障，计时器过期，操作完成都是后台事件的例子。

什么是事件处理?

事件处理是一种机制，如果一个事件发生时，它控制该事件，并且决定应该会发生什么。这种机制具有被称为事件处理程序的代码，当一个事件发生时，它是可执行的。Java 使用代理事件模型来处理事件。该模型定义了标准的机制来生成和处理事件。让我们来简要介绍这个模型。

代理事件模型具有下列主要参与者，即：

- **源** – 源是一个对象，事件发生在该对象上。源负责提供发生事件的信息给它的处理器。Java 提供源对象的类。
- **监听器** – 它也被称为作为事件处理。监听器负责产生对一个事件的响应。从 Java 实现的角度来看，监听器也是一个对象。监听器等待直到它接收到一个事件。一旦收到该事件，监听器进程的事件就返回。

这种方法的好处是，用户界面逻辑完全从生成事件的逻辑中分开。用户界面元素能够把事件的处理委派给一段单独的代码。在这个模型中，监听器需要用源对象注册，以便监听器能够接收事件通知。这是一个有效的处理事件的方式，因为这些事件通知只发送给那些想要接收它们的监听器。

事件处理所涉及的步骤

- 用户单击按钮，然后生成事件。
- 现在有关事件类的对象是自动创建的，源和事件的信息在同一对象得到填充。
- 事件对象被转发到注册监听器类的方法中。
- 该方法现在得到执行并且返回。

有关监听器要记住的要点

- 为了设计一个监听器类，我们必须开发一些监听器接口。这些监听器接口预测一些公共的抽象回调方法，这些方法必须由监听器类来实现。
- 如果你没有实现任何预定义的接口，那么你的类不能作为源对象的监听器类。

回调方法

这些方法由 API 提供者来提供，由应用程序员来定义，并且由应用程序开发者来调用。这里的回调方法代表一个事件方法。响应一个事件 java jre 将触发回调方法。所有这些回调方法在监听器接口中被提供。

如果一个组件需要一些监听器监听它的事件，源必须自己注册给监听器。

事件处理例子

使用你选择的任何编辑器创建下面的 Java 程序，在 D:/ > SWING > com > tutorialspoint > gui >

SwingControlDemo.java

```
package com.tutorialspoint.gui;
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
public class SwingControlDemo {
  private JFrame mainFrame;
  private JLabel headerLabel;
  private JLabel statusLabel;
  private JPanel controlPanel;
  public SwingControlDemo(){
    prepareGUI();
  }
  public static void main(String[] args){
    SwingControlDemo swingControlDemo = new SwingControlDemo();
    swingControlDemo.showEventDemo();
  }
  private void prepareGUI(){
    mainFrame = new JFrame("Java SWING Examples");
    mainFrame.setSize(400,400);
    mainFrame.setLayout(new GridLayout(3, 1));
    headerLabel = new JLabel("",JLabel.CENTER );
    statusLabel = new JLabel("",JLabel.CENTER);
    statusLabel.setSize(350,100);
    mainFrame.addWindowListener(new WindowAdapter() {
      public void windowClosing(WindowEvent windowEvent){
        System.exit(0);
      }
    });
    controlPanel = new JPanel();
    controlPanel.setLayout(new FlowLayout());
    mainFrame.add(headerLabel);
    mainFrame.add(controlPanel);
    mainFrame.add(statusLabel);
    mainFrame.setVisible(true);
  }
  private void showEventDemo(){
    headerLabel.setText("Control in action: Button");
```

```
    JButton okButton = new JButton("OK");
    JButton submitButton = new JButton("Submit");
    JButton cancelButton = new JButton("Cancel");
    okButton.setActionCommand("OK");
    submitButton.setActionCommand("Submit");
    cancelButton.setActionCommand("Cancel");
    okButton.addActionListener(new ButtonClickListener());
    submitButton.addActionListener(new ButtonClickListener());
    cancelButton.addActionListener(new ButtonClickListener());
    controlPanel.add(okButton);
    controlPanel.add(submitButton);
    controlPanel.add(cancelButton);
    mainFrame.setVisible(true);
  }
  private class ButtonClickListener implements ActionListener{
    public void actionPerformed(ActionEvent e) {
      String command = e.getActionCommand();
      if( command.equals( "OK" ))  {
        statusLabel.setText("Ok Button clicked.");
      }
      else if( command.equals( "Submit" ) )  {
        statusLabel.setText("Submit Button clicked.");
      }
      else  {
        statusLabel.setText("Cancel Button clicked.");
      }
    }
  }
}
```

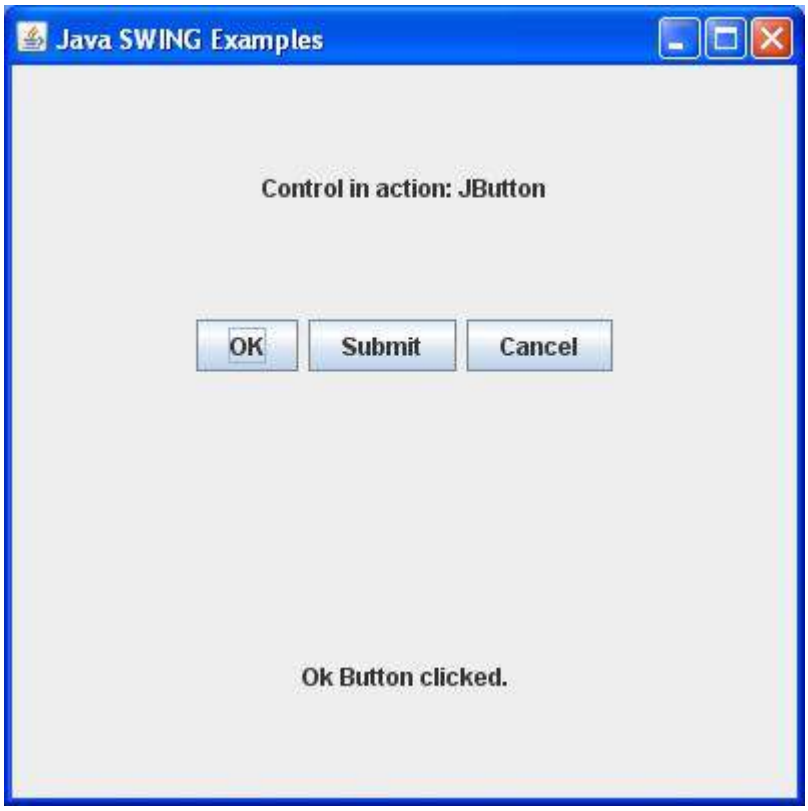
使用命令提示符编译该程序。转到 D:/ > SWING 并输入下面的命令。

```
D:\AWT>javac com\tutorialspoint\gui\SwingControlDemo.java
```

如果没有错误出现，就意味着编译成功。使用下面的命令来运行程序。

```
D:\AWT>java com.tutorialspoint.gui.SwingControlDemo
```

验证下面的输出


Java SWING Examples
Control in action: JButton
OK
Submit
Cancel
Ok Button clicked.

5

SWING 事件类

事件类代表事件。Java 提供各种事件类，但是我们将讨论更频繁使用的那些事件类。

EventObject 类

它是派生所有事件状态对象的根类。所有事件都是用对象，源的引用来构造的，即逻辑上认为是问题最初发生的事件的对象。这个类定义在 java.util 包中。

类声明

下面是 java.util.EventObject 类的声明：

```
public class EventObject
  extends Object
    implements Serializable
```

字段

下面是 java.util.EventObject 类的字段：

- protected Object source ——事件最初发生的对象。

类构造函数

S.N.	构造函数 & 描述
1	EventObject(Object source) 构造一个典型的事件。

类方法

S.N.	方法 & 描述
1	Object getSource() 事件最初发生的对象。
2	String toString() 返回这个 EventObject 的字符串表示。

方法继承

这个类从下面的类中继承方法：

- java.lang.Object

SWING 事件类：

下面是常用的事件类。

Sr. No.	控件 & 描述
1	AWTEvent 它是所有 SWING 事件的根事件类。这个类和它的子类取代了最初的 java.awt.Event 类。
2	ActionEvent 当单击按钮或双点击列表的项时，生成 ActionEvent。
3	InputEvent InputEvent 类是所有组件层输入事件的根事件类。
4	KeyEvent 在按下一个字符时，按键事件生成。
5	MouseEvent 这个事件表明一个鼠标动作发生在一个组件中。
6	WindowEvent 这个类的对象代表一个窗口状态的变化。
7	AdjustmentEvent 这个类的对象代表由可调整的对象发出的调整事件。
8	ComponentEvent 这个类的对象代表一个窗口状态的变化。
9	ContainerEvent 这个类的对象代表一个窗口状态的变化。
10	MouseMotionEvent 这个类的对象代表一个窗口状态的变化。
11	PaintEvent 这个类的对象代表一个窗口状态的变化。

6

SWING 事件监听器

事件监听器代表负责处理事件的接口。Java 提供了各种事件监听器类，但我们将讨论更频繁使用的那些事件监听器类。一个事件监听器的每个方法有一个参数作为一个对象，该对象是 EventObject 类的子类。例如，鼠标事件监听器的方法将接受 MouseEvent 的实例，其中 MouseEvent 是 EventObject 派生的。

EventListner 接口

它是一个标记接口，每一个监听器接口必须扩展它。这个类定义在 java.util 包中。

类声明

下面是 java.util.EventListener 接口的声明：

```
public interface EventListener
```

SWING 事件监听器接口：

下面是常用的事件监听器列表。

Sr. No.	控件 & 描述
1	ActionListener 这个接口用于接收动作事件。
2	ComponentListener 这个接口用于接收组件事件。
3	ItemListener 这个接口用于接收项目事件。
4	KeyListener 这个接口用于接收按键事件。
5	MouseListener 这个接口用于接收鼠标事件。
6	WindowListener 这个接口用于接收窗口事件。
7	AdjustmentListener 这个接口用于接收调整事件。
8	ContainerListener 这个接口用于接收容器事件。
9	MouseMotionListener 这个接口用于接收鼠标移动事件。
10	FocusListener 这个接口用于接收焦点事件。

7

SWING 事件适配器

适配器是用于接收各种事件的抽象类。这些类中的方法是空的。这些类的存在是为了方便创建监听器对象。

SWING 适配器：

下面是当在 SWING 中监听 GUI 事件时常用的适配器列表。

Sr. No.	适配器 & 描述
1	FocusAdapter 用于接收焦点事件的抽象适配器类。
2	KeyAdapter 用于接收按键事件的抽象适配器类。
3	MouseAdapter 用于接收鼠标事件的抽象适配器类。
4	MouseMotionAdapter 用于接收鼠标移动事件的抽象适配器类。
5	WindowAdapter 用于接收窗口事件的抽象适配器类。

8

SWING 布局

引言

布局意味着容器内组件的安排。我们可以说，用其他方式在容器的特定位置放置组件。布局控件的任务是由布局管理器自动完成的。

布局管理器

布局管理器自动放置容器内的所有组件。如果我们不使用布局管理器，然后组件也能由默认的布局管理器放置。手工布局控件也是可能的，但是由于以下两个原因，它变得非常困难。

- 在容器内处理大量的控件是非常繁琐的。
- 通常当我们需要安排组件时，没有给出该组件的宽度和高度信息，。

Java 为我们提供了各种布局管理器来放置控件。属性如大小，形状和排列从一个布局管理器到其他的布局管理器变化。当小应用程序或应用程序窗口的大小改变时，组件的大小，形状和排列的组件也相应的变化，即布局管理器适应小应用程序视图或应用程序窗口的尺寸。

布局管理器与每一个容器对象相关联。每一个布局管理器是实现布局管理接口的类的一个对象。

下面是接口定义的布局管理器的功能。

序号	接口 & 描述
1	LayoutManager LayoutManager 接口声明那些需要由类来实现的方法，这些类的对象将充当一个布局管理器。
2	LayoutManager2 LayoutManager2 是 LayoutManager 的子接口。这个接口是为那些知道如何基于布局约束对象来布局容器的类。

AWT 布局管理器类：

下面是当使用 AWT 设计 GUI 时常用的控件列表。

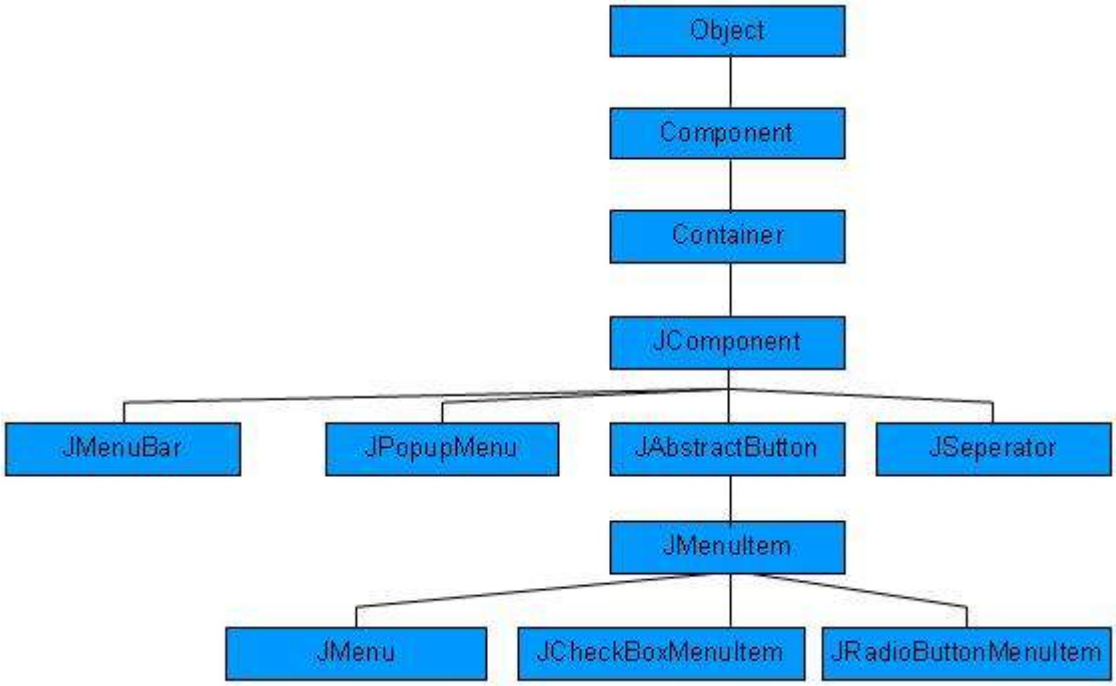
序号	布局管理器 & 描述
1	BorderLayout Borderlayout 安排组件适应于五个地区：东、西、北、南和中心。
2	CardLayout CardLayout 对象把容器中的每一个组件看成一个卡片。一次只有一个卡片是可见的。
3	FlowLayout FlowLayout 是默认的布局。它用定向流动来布局组件。
4	GridLayout GridLayout 用一个矩形网格形式来管理组件。
5	GridBagLayout 这是最灵活的布局管理器类。在不需要相同大小的组件的情况下，GridBagLayout 对象垂直、水平或沿着它们的基线来排列组件。
6	GroupLayout GroupLayout 分层次地归类组件，为了在一个容器中放置它们。
7	SpringLayout SpringLayout 根据一组约束安置与它相关的容器的孩子。

9

SWING 菜单类

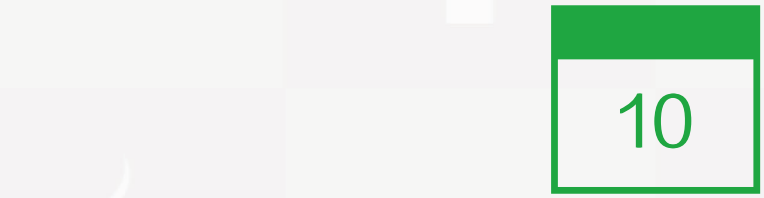
正如我们所知道每个顶层窗口有一个菜单栏与它相关联。这个菜单栏包括各种菜单可用的选择给最终用户。而且每个选择包含被称为下拉菜单的选项列表。菜单和菜单项的控件都是 MenuComponent 类的子类。

菜单层次结构


Object
Component
Container
JComponent
JMenuBar
JPopupMenu
JAbstractButton
JSeperator
JMenuItem
JMenu
JCheckBoxMenuItem
JRadioButtonMenuItem

菜单控件

Sr. No.	控件 & 描述
1	JMenuBar JMenuBar 对象是与顶层窗口相关联的
2	JMenuItem 菜单中的项目必须属于 JMenuItem 或任何它的子类。
3	JMenu JMenu 对象是从菜单栏中显示的一个下拉菜单组件。
4	JCheckBoxMenuItem JCheckBoxMenuItem 是 JMenuItem 的子类。
5	JRadioButtonMenuItem JRadioButtonMenuItem 是 JMenuItem 的子类。
6	JPopupMenu JPopupMenu 可以在一个组件内的指定位置动态地弹出。



10

SWING 容器

容器是 SWING GUI 组件的组成部分。一个容器提供了一个可以放置组件的空间。在 AWT 中，一个容器是组件本身，并且它增加了功能来添加组件本身。下面是需要考虑的注意事项。

- 容器的子类被称为容器。例如 JPanel，JFrame 和 JWindow。
- 容器可以仅仅添加组件到自身。
- 一个默认的布局使用 setLayout 方法来呈现在每个可以被重写的容器中。

SWING 容器：

下面是当使用 SWING 设计 GUI 事件时常用的容器列表。

序号	容器 & 描述
1	Panel JPanel 是一个最简单的容器。它提供了任何其他组件可以被放置的空间，包括其他面板。
2	Frame JFrame 是一个带有标题和边界的顶层窗口。
3	Window JWindow 对象是一个没有边界和菜单条的顶层窗口。

极客学院
jikexueyuan.com
中国最大的IT职业在线教育平台

更多信息请访问
http://wiki.jikexueyuan.com/project/swing/