

❖ 1 ❖ GUI 编程

基本信息

课程名称： Java 应用与开发

授课教师： 王晓东

授课时间： 第六周

参考教材： 本课程参考教材及资料如下：

- 陈国君主编，Java 程序设计基础（第 5 版），清华大学出版社，2015.5
- Bruce Eckel, Thinking in Java (3rd)

教学目标

1. 了解用 Java 开发桌面软件图形用户界面的常用工具集
2. 掌握 AWT 的常用组件和视觉控制
3. 深入理解 GUI 事件处理机制
4. 了解 Applet，特别是其历史渊源，了解与 Applet 类似的技术
5. 理解 Swing 和 AWT 的关系，学习使用 Swing 的典型组件构建较复杂的图形界面程序

授课方式

理论课： 多媒体教学、程序演示

实验课： 上机编程

教学内容

用 Java，我们可以开发构建平台无关的图形界面程序。常用的 GUI 库包括 AWT、Swing、JavaFX，或者我们可以采用多语言混合开发的模式。本节主要介绍使用 AWT 及 Swing 的进行 GUI 开发的相关技术细节。

1.1 GUI 组件及布局

GUI (Graphical User Interface) 即图形用户界面，Java 主要分为 AWT 和 Swing 两大系列 GUI API。其中，AWT (Abstract Window Toolkit) 即为抽象窗口工具集，是 Java 包含的最基础的图形界面库，AWT 相关软件包主要包括：

java.awt 包 提供基本 GUI 组件、视觉控制和绘图工具 API。

java.awt.event 包 提供 Java GUI 事件处理 API。

1.1.1 组件和容器

组件 (Component) 是图形用户界面的基本组成元素，凡是能够以图形化方式显示在屏幕上并能够与用户进行交互的对象均为组件，如菜单、按钮、标签、文本框、滚动条等。组件包含以下特征：

- 组件不能独立地显示出来，必须将组件放在一定的容器中才可以显示出来。
- JDK 的 java.awt 包中定义了多种 GUI 组件类，如 Menu、Button、Label、TextField 等。
- 抽象类 java.awt.Component 是除菜单相关组件之外所有 Java AWT 组件类的根父类，该类规定了 GUI 组件的基本特性，如尺寸、位置和颜色效果等，并实现了作为一个 GUI 部件所应具备的基本功能。
- java.awt.MenuComponent 是所有与菜单相关的组件的父类。

容器 (Container) 实际上是 Component 的子类，容器类对象本身也是一个组件，具有组件的所有性质，另外还具有容纳其它组件和容器的功能。容器类对象可使用方法 add() 添加组件。

AWT 包含的两种主要的容器类型如下：

java.awt.Window 可自由停泊的顶级窗口。

`java.awt.Panel` 可作为容器容纳其他组件，但不能独立存在，必须被添加到其他容器（如 `Frame`）中。

图1.1展示了 AWT 主要组件和容器的接口与实现类之间的层次关系，需要深入理解并掌握。

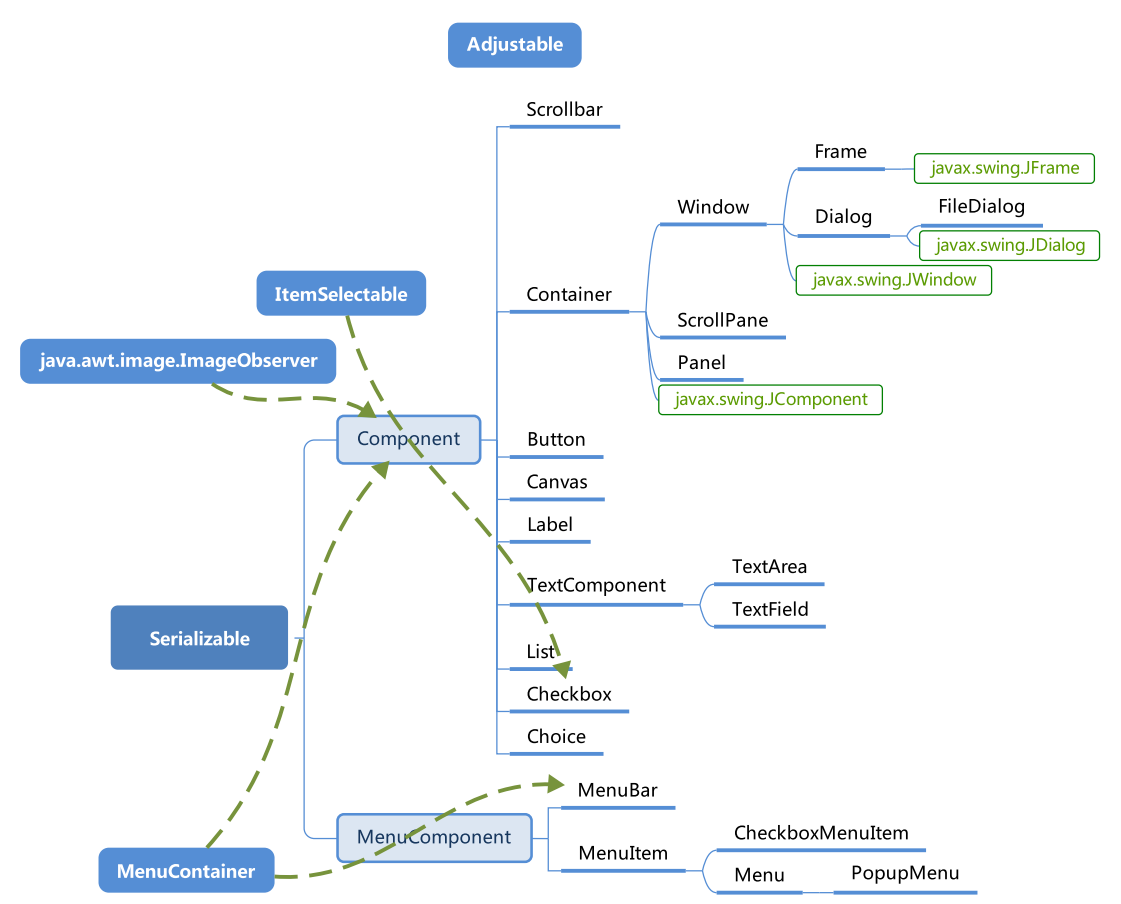


图 1.1 AWT 组件和容器层次架构

1.1.2 常用的组件和容器

AWT 常用的组件和容器如下表所示。

1.1.3 Frame 类

`Frame` 类的显示效果是一个标准的图形窗口，它封装了 GUI 组件的各种属性信息，如尺寸、可见性等。

组件类型	父类	说明
Button	Component	可接收点击操作的矩形 GUI 组件
Canvas	Component	用于绘图的面板
Checkbox	Component	复选框组件
CheckboxMenuItem	MenuItem	复选框菜单项组件
Choice	Component	下拉式列表框，内容不可改变
Component	Object	抽象的组件类
Container	Component	抽象的容器类
Dialog	Window	对话框组件，顶级窗口、带标题栏
FileDialog	Dialog	用于选择文件的平台相关对话框
Frame	Window	基本的 Java GUI 窗口组件
Label	Component	标签类
List	Component	包含内容可变的条目的列表框组件
MenuBar	MenuComponent	菜单条组件
Menu	MenuItem	菜单组件
MenuItem	MenuComponent	菜单项组件
Panel	Container	基本容器类，不能单独停泊
PopupMenu	Menu	弹出式菜单组件
Scrollbar	Component	滚动条组件
ScrollPane	Container	带水平及垂直滚动条的容器组件
TextComponent	Component	TextField 和 TextArea 的基本功能
TextField	TextComponent	单行文本框
TextArea	TextComponent	多行文本域
Window	Container	抽象的 GUI 窗口类，无布局管理器

1. Frame 对象的显示效果是一个可自由停泊的顶级“窗口”，带有标题和尺寸重置角标。
2. Frame 默认初始化为不可见的，可以调用 Frame 对象的 setVisible(true) 方法使之变为可见。
3. 作为容器 Frame 还可使用 add() 方法包含其他组件。

课程配套代码 ▶ sample.awt.FrameSample.java

1.1.4 组件定位

Java 组件在容器中的定位由**布局管理器**决定。如要人工控制组件在容器中的定位，可取消布局管理器，然后使用 Component 类的下述成员方法设置：

- setLocation()
- setSize()
- setBounds()

组件布局在桌面窗口的定位参考图1.2。

1.1.5 Panel 类

- Panel 提供容纳组件的空间。
- Panel 不能独立存在，必须被添加到其他容器中。
- 可以采用和所在容器不同的布局管理器。

课程配套代码 ▶ sample.awt.FrameWithPanelSample.java

1.1.6 布局管理器

容器对其中所包含组件的排列方式，包括组件的位置和大小设定，被称为容器的布局（Layout）。

为了使图形用户界面具有良好的平台无关性，Java 语言提供了布局管理器来管理容器的布局，而不建议直接设置组件在容器中的位置和尺寸。布局管理器类层次如下：

屏幕坐标系(0,0)

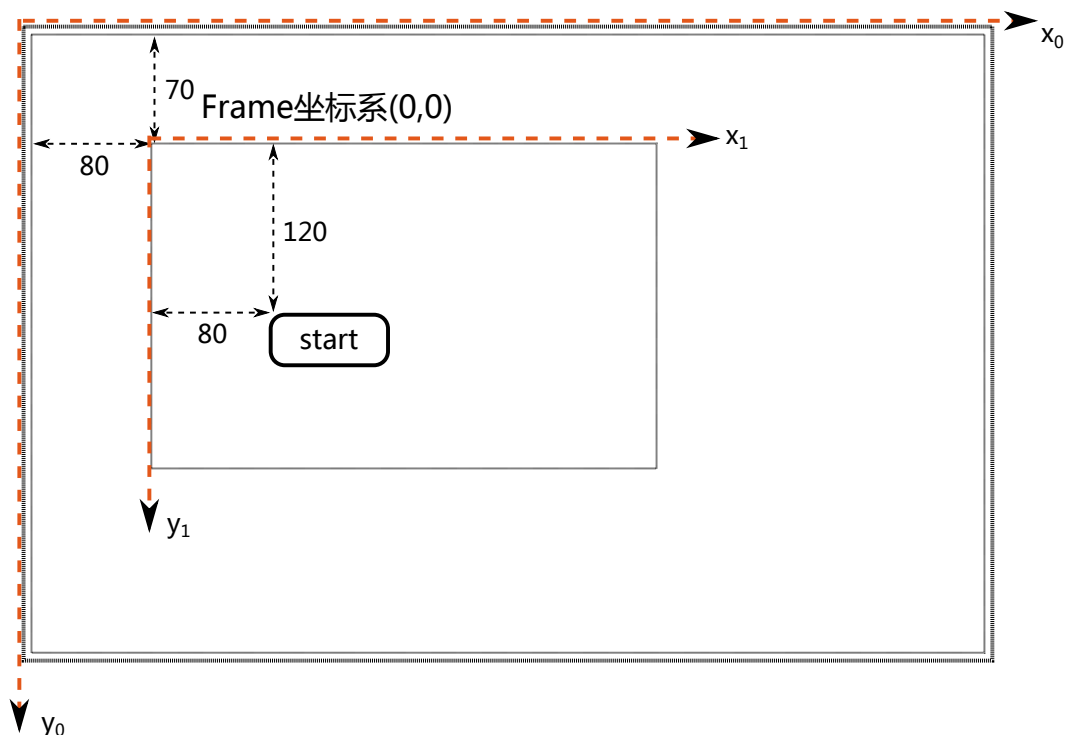


图 1.2 组件定位参照系

LayoutManager----FlowLayout

|
+----GridLayout

LayoutManager2----BorderLayout

|
+----CardLayout
|
+----GridBagLayout

说明

LayoutManager2 是 LayoutManager 的子接口。

每个容器都有一个布局管理器，当容器需要对某个组件进行定位或判断其大小尺寸时，就会调用其对应的布局管理器。可以在容器创建后调用其 `setLayout()` 方法设置其布局管理器类型。Container 类型容器没有默认的布局管理器，即其 `layoutMgr` 属性为 `null`，在其子类中才进行分化。

常用容器的默认的布局管理器如图所示。

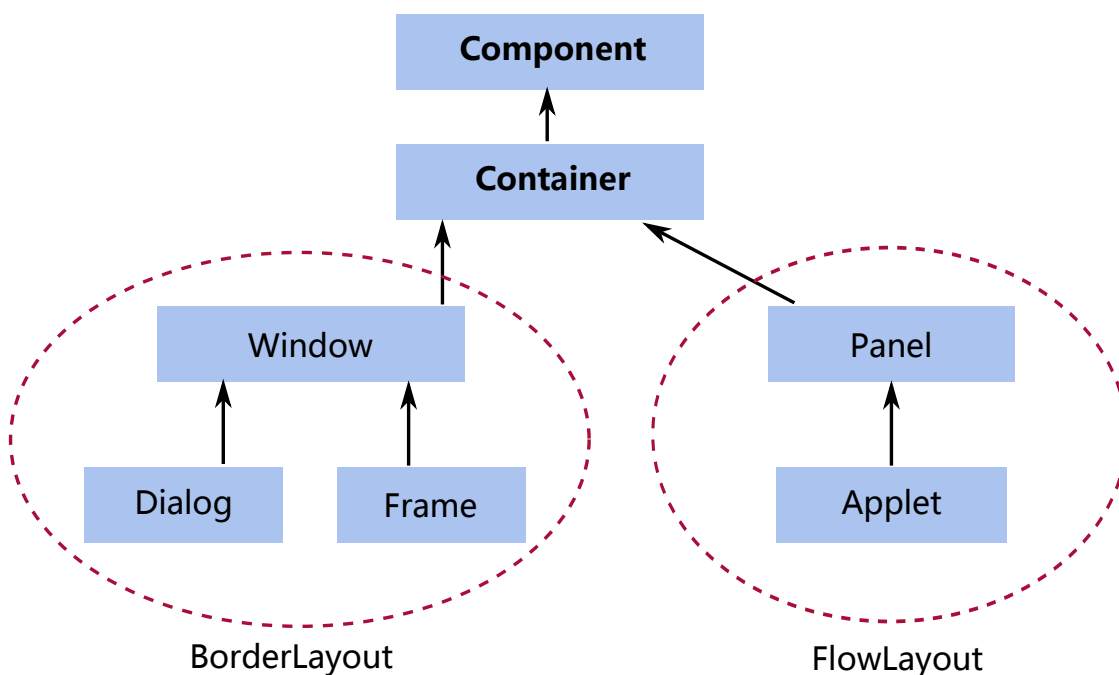


图 1.3 容器默认布局管理器

FlowLayout

流式布局是 **Panel**（及其子类）类型容器的默认布局管理器类型。其布局效果如下：


- 组件在容器中按照加入次序逐行定位，行内从左到右，一行排满后换行。
- 不改变组件尺寸，即按照组件原始大小进行显示。
- 组件间的对齐方式默认为居中对齐，也可在构造方法中设置不同的组件间距、行距及对齐方式。

流式布局管理器提供以下构造方法：

- **public FlowLayout()**
组件对齐方式默认为居中对齐，组件的水平和垂直间距默认为 5 个像素。
- **public FlowLayout(int align)**
显式设定组件的对齐方式，组件的水平和垂直间距默认为 5 个像素。
FlowLayout.LEFT

FlowLayout.RIGHT
FlowLayout.CENTER

- **public FlowLayout(int align, int hgap, int vgap)**
显式设定组件的对齐方式、组件的水平和垂直间距。

课程配套代码  sample.awt.layout.FlowLayoutSample.java

BorderLayout

边界布局是 Window 及其子类（包括 Frame、Dialog）容器的默认布局管理器。其布局效果如下：

- BorderLayout 将整个容器的布局划分成东、西、南、北、中五个区域，组件只能被添加到指定的区域。如不指定组件的加入部位，则默认加入到 Center 区域。
- 每个区域只能加入一个组件，如加入多个，则先前加入的组件会被遗弃。
- 组件尺寸被强行控制，即与其所在区域的尺寸相同。

边界布局管理器提供如下构造方法：

- **public BorderLayout()**
构造一个 BorderLayout 布局管理器，其所包含的组件/区域间距为 0。
- **public BorderLayout(int hgap, int vgap)**
构造一个 BorderLayout 布局管理器，根据参数的组件/区域间距。

❖ BorderLayout 型布局容器尺寸缩放原则

- 北、南两个区域只能在水平方向缩放（宽度可调整）。
- 东、西两个区域只能在垂直方向缩放（高度可调整）。
- 中部可在两个方向上缩放。

GridLayout

网格布局的效果如下：

- 将容器区域划分成规则的矩形网格，每个单元格区域大小相等，组件被添加到每个单元格中，按组件加入顺序先从左到右填满一行后换行，行间从上到下。
- GridLayout 型布局的组件大小也被布局管理器强行控制，与单元格同等大小，当容器尺寸发生改变时，组件的相对位置保持不变，但大小自动调整。

网格布局管理器提供如下构造方法：

- **public GridLayout()** 所有组件于一行中，各占一列。
- **public GridLayout(int rows, int cols)**
通过参数指定布局的行数和列数。
- **public public GridLayout(int rows, int cols, int hgap, int vgap)**
通过参数指定布局的行数、列数，以及组件间水平间距和垂直间距。

课程配套代码 ▶ sample.awt.layout.GridLayoutSample.java

CardLayout

卡片布局的效果如下：

- 将多个组件在同一容器区域内交替显示，相当于多张卡片摞在一起，只有最上面的卡片是可见的。
- 可以按名称显示某一张卡片，或按先后顺序依次显示，也可以直接定位到第一张或最后一张卡片。

卡片布局管理器提供以下主要方法：

- **public void first(Container parent)**
- **public void last(Container parent)**
- **public void previous(Container parent)**
- **public void next(Container parent)**
- **public void show(Container parent, String name)**

课程配套代码 ▶ sample.awt.layout.CardLayoutSample.java

1.1.7 容器的嵌套使用

利用容器嵌套可以在某个原本只能包含一个组件的区域中显示多个组件。

课程配套代码 ▶ sample.awt.layout.FlowLayoutSample.java

1.2 GUI 事件处理

1.2.1 Java 事件和事件处理机制

从 JDK 1.1 开始, Java 采用了一种名为“事件代理模型”(Event Delegation Model)的事件处理机制。基本原理如下:

1. 事先定义多种事件类型
2. 约定各种 GUI 组件在与用户交互时, 遇到特定操作则会触发相应的事件, 即自动创建事件类对象并提交给 Java 运行时系统
3. 系统接收到事件类对象后, 立即将其发送给专门的事件处理对象, 该对象调用其事件处理方法, 处理先前的事件类型对象, 实现预期的处理逻辑

若需要关注某个组件产生的事件, 则可以在该组件上注册适当的事件处理方法, 实际上注册的事件处理器方法所属类型的一个对象——事件监听器。如图1.4所示。

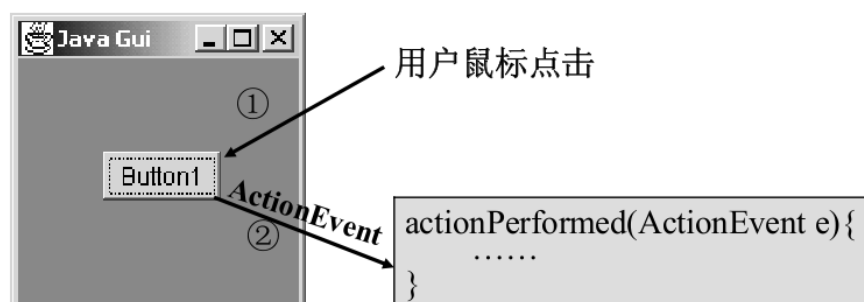


图 1.4 事件处理机制示例