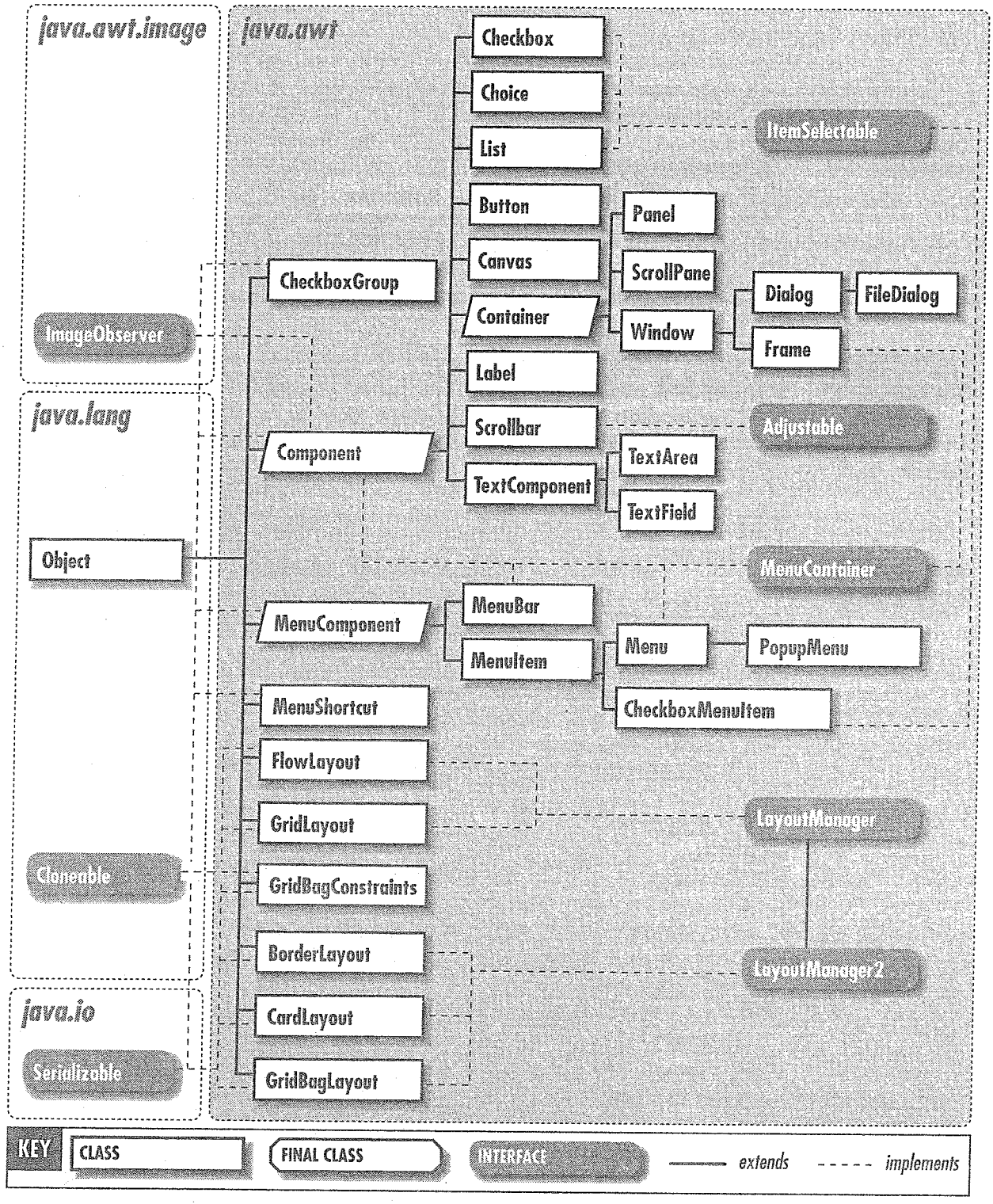

Figure 9–2: GUI classes of the java.awt package

```
public abstract interface ActiveEvent {
// Public Instance Methods
    public abstract void dispatch();
}
```

Implementations: java.awt.event.InvocationEvent

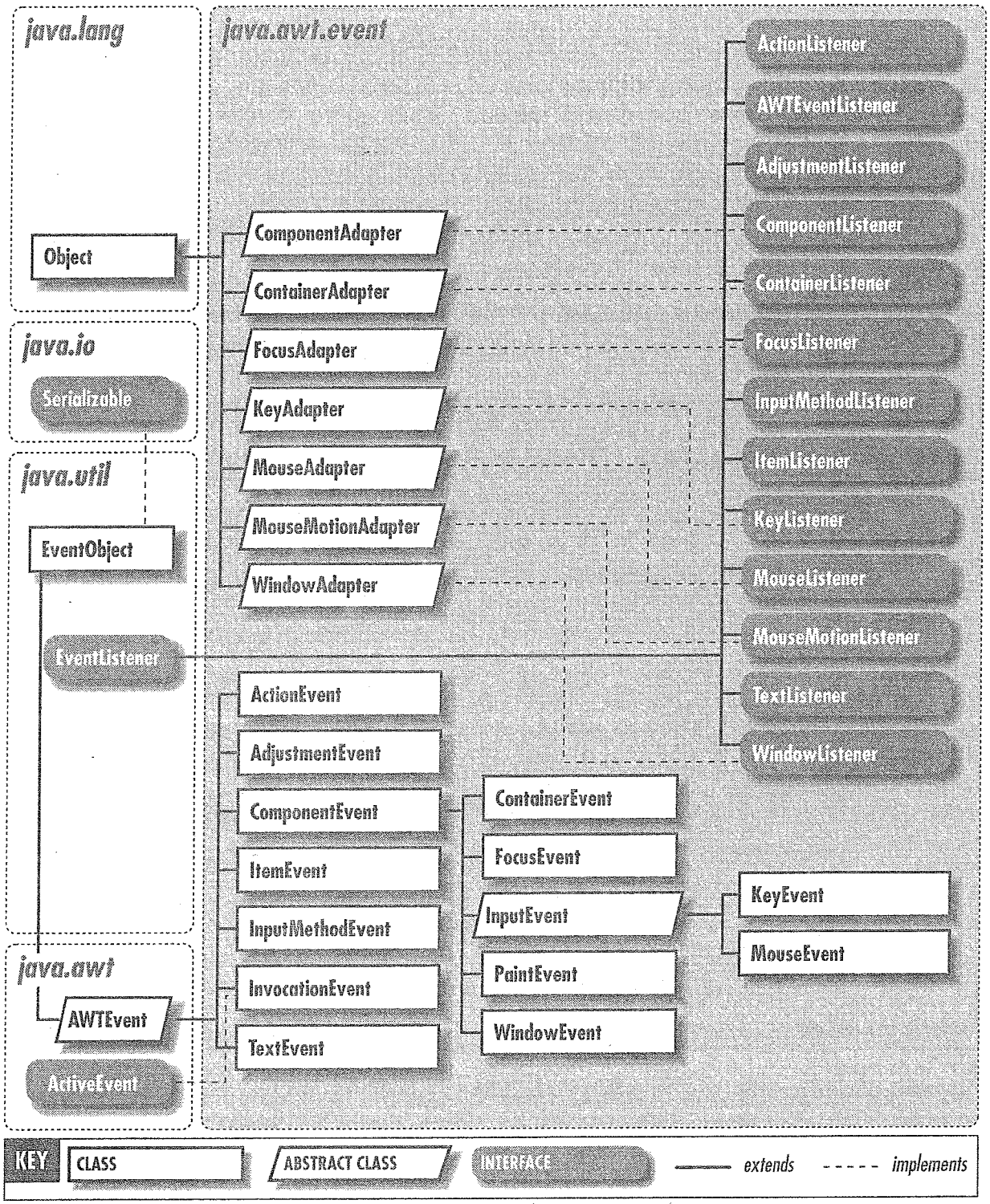

Figure 14–1: The java.awt.event package

ActionEvent

Java 1.1

java.awt.event

serializable event PJ1.1

An object of this class represents a high-level action event generated by an AWT component. Instead of representing a direct user event, such as a mouse or keyboard event, ActionEvent represents some sort of action performed by the user on an AWT component.

The getID() method returns the type of action that has occurred. For AWT-generated action events, this type is always ActionEvent.ACTION_PERFORMED; custom components can generate action events of other types.

Table 2–7: AWT Event Listeners and the Components That Use Them

Event Listener	*Listener Methods*	*Registered on*
ActionListener	actionPerformed()	AbstractButton, Button, ButtonModel, ComboBoxEditor, JComboBox, JFileChooser, JTextField, List, MenuItem, TextField, Timer
AdjustmentListener	adjustmentValue-Changed()	Adjustable, JScrollBar, Scrollbar
ComponentListener	componentHidden(), componentMoved(), componentResized(), componentShown()	Component
ContainerListener	componentAdded(), componentRemoved()	Container
FocusListener	focusGained(), focusLost()	Component
ItemListener	itemStateChanged()	AbstractButton, ButtonModel, Checkbox, CheckboxMenuItem, Choice, ItemSelectable, JComboBox, List
KeyListener	keyPressed(), keyReleased(), keyTyped()	Component
MouseListener	mouseClicked(), mouseEntered(), mouseExited(), mousePressed(), mouseReleased()	Component
MouseMotionListener	mouseDragged(), mouseMoved()	Component
TextListener	textValueChanged()	TextComponent
WindowListener	windowActivated(), windowClosed(), windowClosing(), windowDeactivated(), windowDeiconified(), windowIconified(), windowOpened()	Window

Table 2–8: Swing Event Listeners and the Components That Use Them

commonly used feature of JRootPane, however, is that it displays a JMenuBar passed to its setJMenuBar() method. (In AWT, you specify a MenuBar for a Frame by calling the setMenuBar() method of Frame.)

Layout Management

Some containers, such as JTabbedPane and JSplitPane, define a particular arrangement for their children. Other containers such as JPanel (and JFrame, JDialog, and other top-level containers that use JPanel as their default content pane) do not define any particular arrangement. When working with containers of this type, you must specify a LayoutManager object to arrange the children within the container.

AWT and Swing include various implementations of the java.awt.LayoutManager interface. Each arranges components in a different way. Table 2-5 lists the layout managers defined by AWT. Swing applications often rely on these AWT layout managers, but Swing also defines some of its own, which are listed in Table 2-6. Figure 2-2 shows how some of these layout managers arrange their children.

Table 2–5: AWT Layout Managers

Layout Manager	*Description*
BorderLayout	Lays out a maximum of five components: one along each of the four borders of the container and one in the center. When using this layout manager, you must add components to the container using a two-argument version of the add() method. The constraint argument should be one of the strings "North", "East", "South", "West", or "Center". Despite the simplicity of this layout system, this layout manager is used quite often.
CardLayout	Makes each component as large as the container and displays only one at a time. Various methods change the currently displayed component.
FlowLayout	Arranges components like words on a page: from left to right in rows and then top to bottom as each row fills up. Rows may be left, center, or right justified.
GridBagLayout	A flexible layout manager that arranges components in a grid with variable-sized cells. Allows explicit control over the way each component is resized when the container changes size. Requires a complex constraints set specified with the GridBagConstraints object.
GridLayout	Makes all components the same size and arranges them in a grid of specified dimensions.

the list of components it supports. Table 2-1 lists the heavyweight components provided by AWT, where heavyweight refers to components that are layered on top of native GUI components. The components listed are all classes in the `java.awt` package. One of the curious features of the AWT is that pull-down and pop-up menus, and the items contained within those menus, are not technically components. Instead of inheriting from `Component`, they inherit from `java.awt.MenuComponent`. Nevertheless, the various menu component classes are used in very much the same way that true components are, so I have included them in Table 2-1.

Table 2–1: Heavyweight AWT Components

Component Name	*Description*
`Button`	A graphical push button.
`Canvas`	A heavyweight component that displays a blank canvas, allowing a program to display custom graphics.
`Checkbox`	A toggle button that can be selected or unselected. Use the `Checkbox` group to enforce mutually exclusive or radio button behavior among a group of `Checkbox` components.
`CheckboxMenuItem`	A toggle button that can appear within a `Menu`.
`Choice`	An option menu or drop-down list. Displays a menu of options when clicked on and allows the user to select among this fixed set of options.
`Component`	The base class for all AWT and Swing components. Defines many basic methods inherited by all components.
`FileDialog`	Allows the user to browse the filesystem and select or enter a filename.
`Label`	Displays a single line of read-only text. Does not respond to user input in any way.
`List`	Displays a list of choices (optionally scrollable) to the user and allows the user to select one or more of them.
`Menu`	A single pane of a pull-down menu
`MenuBar`	A horizontal bar that contains pull-down menus.
`MenuComponent`	The base class from which all menu-related classes inherit.
`MenuItem`	A single item within a pull-down or pop-up menu pane.
`PopUpMenu`	A menu pane for a pop-up menu.
`Scrollbar`	A graphical scrollbar.
`TextArea`	Displays multiple lines of plain text and allows the user to edit the text.
`TextComponent`	The base class for both `TextArea` and `TextField`.
`TextField`	Displays a single line of plain text and allows the user to edit the text.

Table 2–3: AWT Containers

Container	*Description*
`Applet`	This subclass of `Panel` is actually part of the `java.applet` package. It is the base class for all applets. (See Chapter 7.)
`Container`	The base class from which all containers inherit.
`Dialog`	A window suitable for dialog boxes.
`Frame`	A window suitable for use as the main window of an application. In AWT, `Frame` is the only container that can contain a `MenuBar` and related menu components.
`Panel`	A generic container used to create nested layouts.
`ScrollPane`	A container that contains a single child and allows that child to be scrolled vertically and horizontally.
`Window`	A heavyweight window with no titlebar or other decoration, suitable for pop-up menus and similar uses.

Table 2–4: Swing Containers

Container	*Description*
`Box`	A general-purpose container that arranges children using the `BoxLayout` layout manager.
`JApplet`	A `java.applet.Applet` subclass that contains a `JRootPane` to add Swing features, such as support for menu bars to applets. Applets are discussed in Chapter 7.
`JDesktopPane`	A container for `JInternalFrame` components; simulates the operation of a desktop within a single window. Supports MDI (multiple document interface) application styles.
`JDialog`	The container used to display dialog boxes in Swing.
`JFrame`	The container used for top-level windows in Swing.
`JInternalFrame`	A lightweight nested window container. Behaves like a `JFrame` and displays a titlebar and resize handles but is not an independent window and is constrained to appear within the bounds of its parent container. Often used with `JDesktopPane`.
`JLayeredPane`	A container that allows its children to overlap and manages the stacking order of those children.
`JPanel`	A generic container for grouping Swing components. Typically used with an appropriate `LayoutManager`.
`JRootPane`	A complex container used internally by `JApplet`, `JDialog`, `JFrame`, `JInternalFrame`, and `JWindow`. Provides a number of important Swing capabilities to these top-level containers.
`JScrollPane`	A container that allows a single child component to be scrolled horizontally or vertically. Supports scrolling and non-scrolling header regions at the top and left of the scrolling region.