1. Система Swing

Swing API — это набор классов, который обеспечивает более мощные и гибкие компоненты, чем AWT. В дополнение к знакомым компонентам типа кнопок, флажков и меток Swing поставляет несколько интересных добавлений, включая панели со вкладками, панели с прокруткой, деревья и таблицы. Даже знакомые компоненты, такие как кнопки, имеют в Swing больше возможностей. Например, с кнопкой можно связать как изображение, так и текстовую строку. Кроме того, изображение может изменяться, когда изменяется состояние кнопки.

В отличие от AWT-компонентов, Swing-компоненты не реализованы специфическим для платформы кодом. Вместо этого они написаны полностью на Java и, поэтому, платформно-независимы. Для описания таких элементов используется термин облегченный (lightweight).

Число классов и интерфейсов в пакетах Swing достаточно велико, так что здесь приводится краткий обзор только некоторых из них.

В табл. 19.1 показаны классы Swing-компонентов, которые используются в этой книге.

Таблица 19.1. Классы Swing-компонентов

Класс	Описание
AbstractButton	Абстрактный суперкласс для кнопок Swing
ButtonGroup	Инкапсулирует взаимоисключающий набор кнопок
ImageIcon	Инкапсулирует значок
JApplet	Swing-версия класса Applet
JButton	Класс Swing-кнопок
JCheckBox	Класс Swing-флажков
JComboBox	Инкапсулирует combo box (комбинация раскрывающегося списка и текстового поля)
JLabel	Swing-версия метки
JRadioButton	Swing-версия переключателей
JScrollPane	Инкапсулирует прокручиваемую панель
JTabbedPane	Инкапсулирует панели с вкладками
JTable	Инкапсулирует таблицы или сетки
JTextField	Swing-версия текстового поля
JTree	Инкапсулирует деревья

Относящиеся к Swing классы содержатся в пакете javax.swing и его подпакетах, таких как javax.swing.tree. Существует много других Swing-классов и интерфейсов, которые в данной главе не рассматриваются. Здесь мы разберем лишь некоторые Swing-компоненты и проиллюстрируем их на примерах апплетов.

1.1. Класс JApplet

Фундаментальным для Swing является класс JApplet, который расширяет класс Applet. Апплеты, которые используют Swing-компоненты, должны быть подклассами JApplet. JApplet богат функциональными возможностями, которых нет в Applet. Например, JApplet поддерживает различные "панели", такие как панель содержания (content pane), прозрачная ("стеклянная") панель (glass pane) и корневая панель (root pane). Обсудим одно различие между Applet и JApplet, которое используется в приаодимых далее примерах апплетов. При добавлении компонента к экземпляру JApplet не вызывайте метод add() для апплета. Вместо этого, вызовите add() для панели содержания JApplet-объекта. Панель содержания может быть получена с помощью следующего метода:

```
Container getContentPane()
```

Чтобы добавить компонент в панель содержания, можно использовать метод add() класса Container. Его форма:

```
void add(camp)
```

где camp — компонент, который будет добавлен к панели содержания.

Значки и метки

В Swing значки инкапсулированы классом ImageIcon, который рисует значок из изображения. Ниже показаны два его конструктора:

```
ImageIcon(String filename)
ImageIcon (URL url)
```

Первая форма использует изображение в файле с именем filename, а вторая форма — в ресурсе, расположенном по URL-адресу url.

Класс ImageIcon реализует интерфейс Icon, который объявляет методы, представленные в табл. 26.2.

Таблица 19.2. Методы класса ImageIcon

Метод	Описание
int getIconHeight()	Возвращает высоту значка в пикселах

`int getIconWidth ()`	Возвращает ширину значка в пикселах
`void paintIcon(Component comp, Graphics g, int x, int y)`	Рисует значок в позиции (x, y) с графическим контекстом g Дополнительная ин формация об операции рисования обес печена в comp

Метки Swing — экземпляры класса `JLabel`, который расширяет `JComponent`. Он может отображать тексты и/или значки. Вот некоторые из его конструкторов:

```
JLabel(Icon i)
JLabel(String s)
JLabel(String s, Icon i, int align)
```

Здесь `s` и `i` – текст и значок, используемый для метки. Параметр `align` определяет выравнивание и имеет значения `left`, `right` или `center`. Эти константы определены в интерфейсе `SwingConstants`, наряду с несколькими другими, используемыми Swing-классами.

Значок и текст, связанный с меткой, можно считывать и записывать следующими методами:

```
Icon getIcon()
String getText()
void setIcon(Icon i)
void setText(String s)
```

Здесь `i` и `s` — значок и текст, соответственно.

Следующий пример показывает, как можно создать и отобразить метку, содержащую как значок, так и строку. Апплет начинается с получения панели содержания. Затем, создается объект `ImageIcon` для файла `france.gif`. Он используется как второй параметр конструктора `JLabel`. Первый и последний параметры для конструктора `JLabel` — текст метки и выравнивание. Наконец, метка добавляется к панели содержания.

Программа 145. Кнопка с картинкой

```
// Файл JLabelDemo.java
import java.awt.*;
import javax.swing.*;
/*
<applet code = "JLabelDemo" width = 250 height = 150>
</applet>
*/
public class JLabelDemo extends JApplet {
   public void init() {
   // Получить панель содержания
      Container contentPane = getContentPane();
   // Создать значок
      ImageIcon ii = new ImageIcon("Russian.gif");
   // Создать метку
      JLabel jl = new JLabel("Russian", ii, JLabel.CENTER);
```

```
    // Добавить метку к панели содержания
       contentPane.add(jl);
    }
}
```

Окно апплета показано на рис.1.

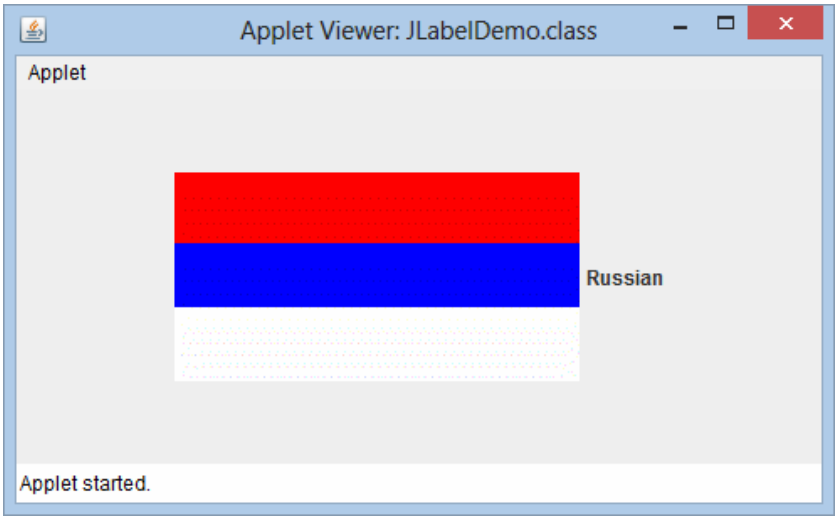

Рис. 1. Окно апплета и кнопка с изображением

Текстовые поля

Поле текста Swing инкапсулировано классом `jTextComponent`, который расширяет `JComponent`. Он обеспечивает функциональные возможности, которые являются общими для текстовых Swing-компонентов. Один из его подклассов — `jTextField`, позволяет редактировать одну строку текста. Вот некоторые из его конструкторов:

```
JTextField()
JTextField(int cols)
JTextField (String s, int cols)
JTextField(String s)
```

Здесь `s` – строка, которая будет представлена; `cols` – число позиций в текстовом поле.

Следующий пример показывает, как можно создать текстовое поле. Апплет начинается с получения его панели содержания и затем для нее устанавливается поточное размещение в качестве менеджера компоновки. Далее, создается объект `JTextField` и добавляется к панели содержания.

Программа 146. Текстовое поле

```
// Файл JTextFieldDemo.java
import java.awt.*;
import javax.swing.*;
/*
<applet code = "JTextFieldDemo" width = 300 height = 50>
</applet>
*/
public class JTextFieldDemo extends JApplet {
   JTextField jtf;
   public void init() {
   // Получить панель содержания
      Container contentPane = getContentPane();
      contentPane.setLayout(new FlowLayout()) ;
   // Добавить текстовое поле к панели содержания
      jtf = new JTextField(15);
      contentPane.add(jtf);
   }
}
```

Вывод этого апплета представлен на рис. 26.2.

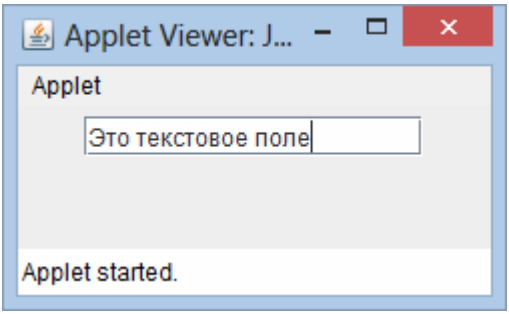

Рис. 2. Апплет с текстовым полем

Кнопки

Кнопки Swing обладают свойствами, которых нет в классе `Button`, определенном в AWT. Например, с кнопкой Swing можно связать изображение. Кнопки Swing — это подклассы класса `AbstractButton`, который расширяет `JComponent`. `AbstractButton` содержит много методов, которые позволяют управлять поведением кнопок, флажков и переключателей. Например, можно определять различные пиктограммы для отображения компонента, когда он отжат (`disabled`), нажат (`pressed`), или выбран (`selected`). Некоторую пиктограмму можно использовать как значок "`наезда`" (`rollover`), который отображается, когда курсор мыши установлен поверх этого компонента ("`наехал`" на него). Ниже следуют описания форматов методов, которые управляют этим поведением:

```
void setDisabledIcon(Icon di)
```

```
void setPressedIcon(Icon pi)
void setSelectedIcon(Icon si)
void setRolloverIcon(Icon ri)
```

Здесь `di`, `pi`, `si` и `ri` — пиктограммы, которые нужно использовать для этих различных состояний.

Текст, связанный с кнопкой, можно читать и записывать с помощью следующих методов:

```
String getText()
void setText(String s)
```

Здесь `s` — текст, который нужно связать с кнопкой.

При нажатии кнопки конкретные подклассы `AbstractButton` генерируют `action`-события. Блоки прослушивания регистрируют и отменяют регистрацию для этих событий с помощью следующих методов:

```
void addActionListener (ActionListener al)
void removeActionListener (ActionListener al)
```

Здесь `al` — блок прослушивания событий действия. `AbstractButton` — это суперкласс для кнопок, флажков и переключателей. Рассмотрим каждый из них.

Класс JButton

Класс `JButton` обеспечивает функциональные возможности кнопки. `JButton` позволяет связать с кнопкой изображение, строку или и то и другое. Некоторые из его конструкторов:

```
JButton(Icon i)
JButton(String s)
JButton(String s. Icon i)
```

Здесь `s` и **`i`** — строка и изображение, используемые для кнопки.

Следующий пример демонстрирует четыре кнопки и текстовое поле. Каждая кнопка отображает пиктограмму, которая представляет флажок страны. Когда кнопка нажимается, в текстовом поле выводится название этой страны. Апплет начинается с получения панели содержания и установки для нее менеджера компоновки. Создаются четыре кнопки-изображения и добавляются к панели содержания. Затем апплет регистрируется, чтобы принимать генерируемые кнопками action-события. Далее, создается текстовое поле и добавляется к апплету. Наконец, обработчик action-событий отображает командную строку, которая связана с кнопкой. Для представления этой строки используется текстовое поле.

Программа 147. Кнопки с рисунками

```
// Файл JButtonDemo.java
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
/*
<applet code = "JButtonDemo" width = 250 height = 300>
</applet>
*/
public class JButtonDemo extends JApplet implements ActionListener {
   JTextField jtf;
   public void init() {
   // Получить панель содержания
      Container contentPane = getContentPane();
      contentPane.setLayout(new FlowLayout()) ;
   // Добавить кнопки в панель содержания
      ImageIcon russia = new ImageIcon("russian.gif");
      JButton jb = new JButton(russia);
      jb.setActionCommand("Russia");
      jb.addActionListener(this);
      contentPane.add(jb);

      ImageIcon france = new ImageIcon("france.gif");
      jb = new JButton(france);
      jb.setActionCommand("France");
      jb.addActionListener(this);
      contentPane.add(jb);

      ImageIcon germany = new ImageIcon("germany.gif");
      jb = new JButton(germany);
      jb. setActionCommand ("Germany") ;
      jb.addActionListener(this);
      contentPane.add(jb);

      ImageIcon japan = new ImageIcon("japan.gif");
      jb = new JButton(japan);
      jb.setActionCommand("Japan");
      jb.addActionListener(this);
      content Pane.add(jb) ;

   // Добавить текстовое поле в панель содержания
      jtf = new JTextField(15);
      contentPane.add(jtf);
   }
   public void actionPerformed(ActionEvent ae) {
      jtf.setText(ae.getActionCommand());
   }
}
```

Вывод этого апплета представлен на рис.3.

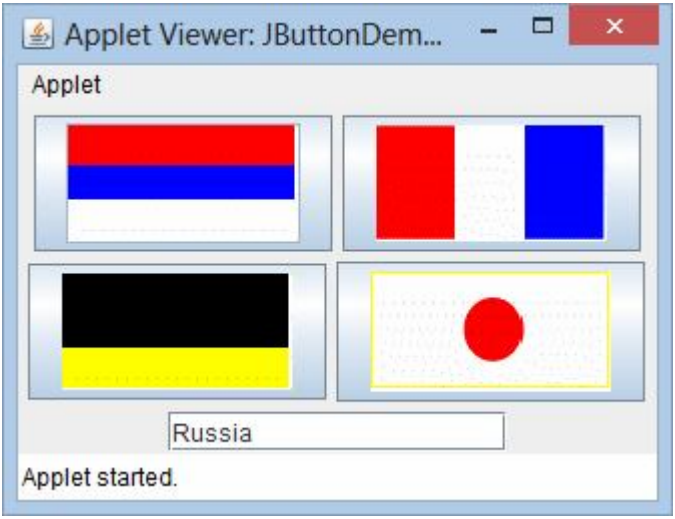

Рис. 3. Кнопки с изображениями

Флажки

Класс `JCheckBox`, который обеспечивает функциональные возможности флажка, является конкретной реализацией класса `AbstractButton`.

Некоторые из его конструкторов:

```
JCheckBox(Icon i)
JCheckBox(Icon i, boolean state)
JCheckBox(String s)
JCheckBox(String s, boolean state)
JCheckBox(String s,  Icon i)
JCheckBox(String s, Icon z, boolean state)
```

Здесь используются следующие параметры: `i` — изображение для кнопки, `s` — текст. Если `state = true`, флажок первоначально выбран. В противном случае — нет.

Состояние флажка может быть изменено с помощью следующего метода:

```
void (boolean state)
```

Здесь параметр `state` должен быть `true`, если нужно, чтобы флажок был установлен (помечен).

Следующий пример показывает, как можно создать апплет, отображающий четыре флажка и текстовое поле. Когда флажок помечается, его подпись отображается в текстовом поле. Сначала получена панель содержания для объекта `JApplet`, и в качестве ее

менеджера компоновки устанавливается поточное размещение. Затем к панели содержания добавлены четыре флажка, и назначены пиктограммы для нормального (без метки), rollover- (с "наездом" указателя мыши) и выбранного (с меткой) состояний. Далее апплет регистрируется, чтобы принимать item-события. Наконец, в панель содержания добавляется текстовое поле.

Когда флажок помечается (выбирается) или сбрасывается (отменяется выбор), генерируется item-событие. Оно обрабатывается методом itemStateChanged(). Внутри itemStateChanged() метод getItem() получает объект JCheckBox, который генерирует событие. Метод getText() получает подпись для этого флажка и использует его для вывода внутри текстового поля.

Программа 148. Флажки с рисунками

```
// Файл JCheckBoxDemo.java
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
/*
<applet code = "JCheckBoxDemo" width = 400 height = 50>
</applet>
*/
public class JCheckBoxDemo extends JApplet implements ItemListener {
   JTextField jtf;
   public void init() {
   // Получить панель содержания
      Container contentPane = getContentPane();
      contentPane.setLayout(new FlowLayout());
   // Создать пиктограммы
      ImageIcon normal = new ImageIcon("normal.gif");
      ImageIcon rollover = new ImageIcon("rollover.gif");
      ImageIcon selected = new ImageIcon("selected.gif");
   // Добавить флажки в панель содержания
      JCheckBox cb = new JCheckBox("C", normal);
      cb.setRolloverIcon(rollover);
      cb.setSelectedIcon(selected);
      cb.addItemListener(this);
      contentPane.add(cb);

      cb = new JCheckBox("C++", normal);
      cb.setRolloverIcon(rollover);
      cb.setSelectedIcon(selected);
      cb.addItemListener(this);
      contentPane.add(cb);

      cb = new JCheckBox("Java", normal);
      cb.setRolloverIcon(rollover);
      cb.setSelectedIcon(selected);
      cb.addItemListener(this);
      contentPane.add(cb);
```

```
        cb = new JCheckBox("Perl",  normal);
        cb.setRolloverIcon(rollover);
        cb.setSelectedIcon(selected);
        cb.addItemListener(this);
        contentPane.add(cb);
    // добавить текстовое поле в панель содержания
        jtf = new JTextField(15);
        contentPane.add(jtf);
    }
    public void itemStateChanged(ItemEvent ie) {
        JCheckBox cb = (JCheckBox)ie.getItem();
        jtf.setText(cb.getText());
    }
}
```

Вывод этого апплета представлен на рис.4.

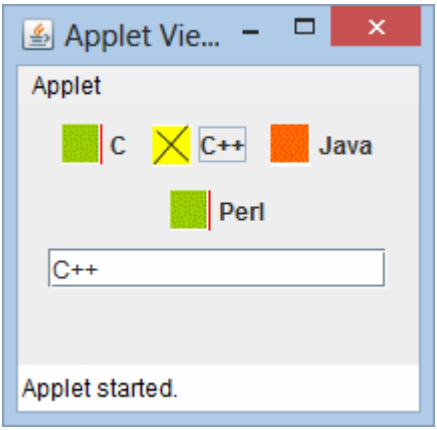

Рис. 4. Флажки с изображениями

Переключатели

Переключатели поддерживаются классом `JRadioButton`, который является конкретной реализацией класса `AbstractButton`. Некоторые из его конструкторов:

```
JRadioButton(Icon i)
JRadioButton(Icon z, boolean state)
JRadioButton(String s)
JRadioButton(String s, boolean state)
JRadioButton(String s, Icon i)
JRadioButton(String s, Icon i, boolean state)
```