

第6章 图形用户界面

图形用户界面（**Graphical User Interface, GUI**），使用图形方式借助菜单、按钮等标准界面元素和键盘、鼠标操作，实现人机交互。

- 6.1 AWT组件及其属性类
- 6.2 事件处理
- 6.3 Swing组件及事件
- 6.4 图形图像





第6章 图形用户界面

内容和要求:

1. 掌握**Java Swing**组件的使用方法，包括窗口、框架、对话框、面板、文本编辑框、按钮、组合框等，掌握多种布局方式，掌握窗口菜单和快捷菜单设计方式。
2. 理解委托事件处理模型，掌握不同组件、不同事件的事件处理方法，设计出能够响应事件的**Java**图形用户界面。
3. 熟悉在组件上绘图的方法。

重点：组件，布局，事件处理，异常处理。

难点：组件量大，方法众多；采用接口实现的委托事件处理模型。



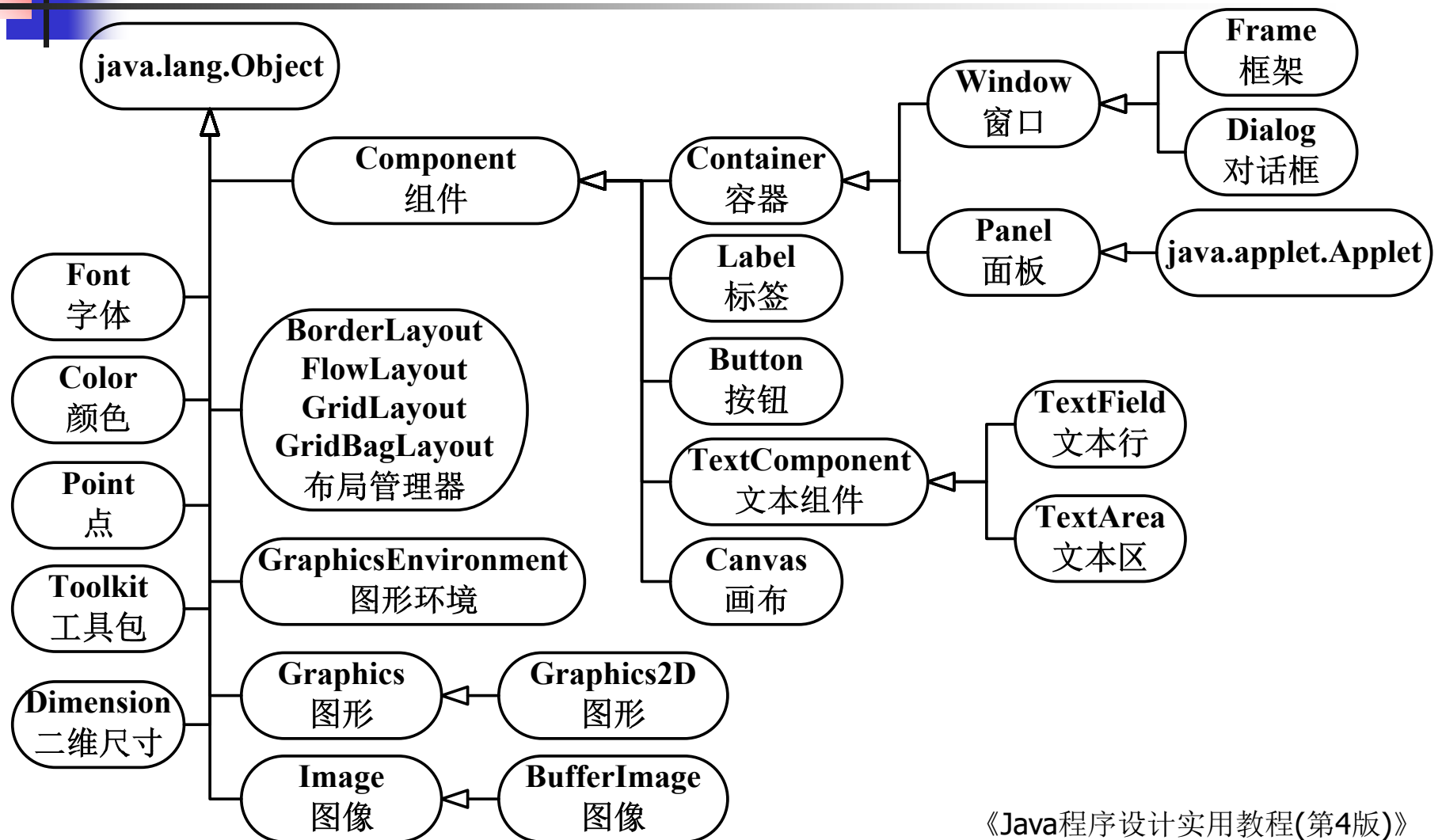
6.1 AWT组件及其属性类

Java的AWT和Swing

- ① **java.awt**包提供抽象窗口工具集（**Abstract Window Toolkit, AWT**）。
- ② **javax.swing**包提供**JDK 1.2**的**Swing**组件，它扩展了**AWT**组件的功能。

1. **6.1.1 AWT组件**
2. **6.1.2 布局管理**
3. **6.1.3 颜色和字体**

6.1.1 AWT组件

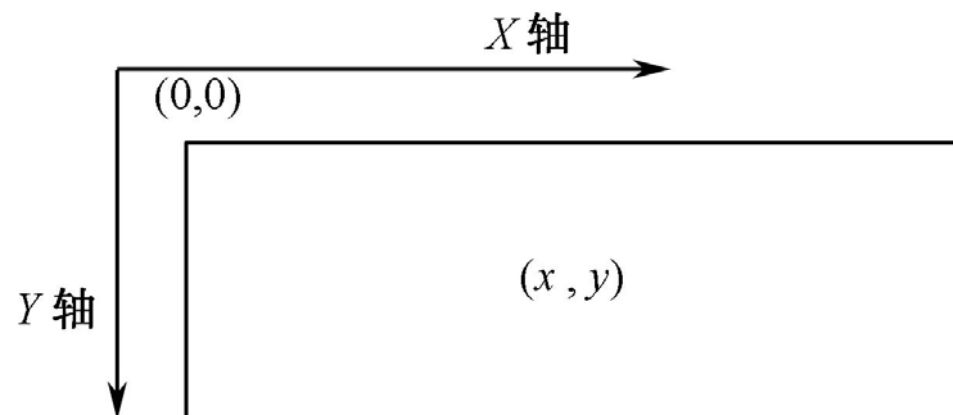


1. 组件

```
public abstract class Component extends Object
    implements ImageObserver, MenuContainer,
    Serializable
{
    public int getWidth()                // 宽度
    public int getHeight()              // 高度
    public void setSize(int width, int height) // 宽度和高度
    public int getX()                   // 位置的X坐标值
    public int getY()                   // 位置的Y坐标值
    public void setLocation(int x, int y)
        // 坐标位置, x、y指定组件左上角相对于容器的坐标位置
    public void setBounds(int x, int y, int width, int height)
        // 坐标位置和宽度、高度
}
```



图6.2 坐标系





Component（所有可以显示出来的图形元素的父类-抽象类）

```
public Color getForeground()           //获得组件的文本颜色  
public void setForeground(Color color) //设置组件的文本颜色  
public Color getBackground()         //获得组件的背景颜色  
public void setBackground(Color color) //设置组件的背景颜色  
public Font getFont()                //获得组件字体  
public void setFont(Font font)        //设置组件字体  
public void setVisible(boolean visible) //设置组件是否显示  
  
public void setEnabled(boolean enabled) //设置是否有效状态  
}
```



2. **Container**（其实例化的对象用于容纳其它的**Component**对象）

```
public class Container extends Component
```

```
{
```

```
    public void setLayout(LayoutManager mgr)
```

```
        // 设置布局，参数类型是接口，类型的多态，  
        实际参数是实现该接口的类的实例
```

```
    public Component add(Component comp)
```

```
        // 添加组件；参数类型是抽象类，类型的多态，  
        实际参数是该抽象类的子类实例，可添加任意组件
```

```
    public void remove(int i)           // 删除组件
```

```
}
```




3. 窗口和面板

```
public class Window extends Container implements Accessible  
// 窗口类
```

```
{  
    public void setLocationRelativeTo(Component comp)  
        // 将窗口置于屏幕中央  
}
```

```
public class Panel extends Container implements Accessible  
// 面板类
```

```
{  
    public Panel() // 默认FlowLayout布局，居中对齐  
    public Panel(LayoutManager layout) // layout指定布局管理器  
}
```



4. 框架和对话框

```
public class Frame extends Window
    implements MenuContainer // 框架
{
    public Frame()
    public Frame(String title)           // 参数title指定框架的标题

    public String getTitle()             // 获取框架的标题
    public void setTitle(String title)    // 设置或修改框架的标题
    public void setResizable(boolean resizable)
                                         // 设置框架是否可变大小
}

public class Dialog extends Window    // 对话框
```



5. 标签

public class Label extends Component implements Accessible

{

public static final int LEFT //左对齐, 默认值

public static final int CENTER //居中

public static final int RIGHT //右对齐

public Label()

public Label(String text) //text指定显示字符串

public Label(String text, int align) //align指定对齐方式

public String getText() //获得显示字符串

public void setText(String text) //设定显示字符串

}



6. 文本行

```
public class TextField extends TextComponent
{
    public TextField()
    public TextField(String text)      //text指定内容
    public TextField(int columns) //columns指定列数
    public TextField(String text, int columns)

    public String getText()           //获得内容
    public void setText(String text)  //设置内容
}
```



7. 按钮

```
public class Button extends Component  
    implements Accessibl
```

```
{
```

```
    public Button(String text)
```

```
        //text指定按钮标题
```

```
}
```



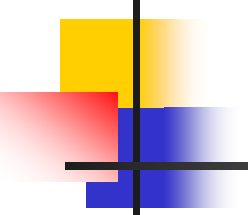
6.1.2 布局管理

1. **FlowLayout** (流布局管理器)

```
public class FlowLayout implements LayoutManager,  
    java.io.Serializable  
{  
    public static final int LEFT  = 0; //左对齐, 类常量, 全部大写  
    public static final int CENTER = 1; //居中  
    public static final int RIGHT  = 2; //右对齐  
  
    public FlowLayout()                //构造方法, 默认居中  
    public FlowLayout(int align)       //align参数指定对齐方式  
}
```

Panel面板的默认布局管理器

2. BorderLayout (边布局管理器)



```
public class BorderLayout implements
    LayoutManager2, java.io.Serializable
{
    public static final String NORTH = "North";
    public static final String SOUTH = "South";
    public static final String EAST = "East";
    public static final String WEST = "West";
    public static final String CENTER = "Center";
    //注意，字符串首字母大写
    public BorderLayout() //构造方法
}
```

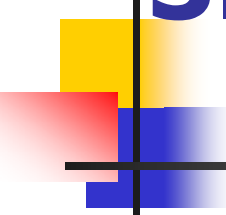
Window窗口的默认布局管理器



容器添加组件（边布局）

容器类提供以下添加组件方法，仅用于
BorderLayout布局。

**public void add(Component comp, Object
constraints)**



3. GridLayout（网格布局管理器）

```
public class GridLayout implements  
    LayoutManager, Serializable  
{  
    public GridLayout(int rows, int cols)  
        // 参数指定行、列  
}
```

容器类提供添加组件的方法，用于
FlowLayout和**GridLayout**布局。

```
public Component add(Component comp,  
    int index)
```



6.1.3 颜色和字体

1. 颜色

```
public class Color implements Paint, java.io.Serializable
{
    public Color(int r, int g, int b)           //以三元色值构造对象
    public Color(int rgb)                       //以RGB值构造对象
    public int getRed()                         //返回红色值
    public int getGreen()                      //返回绿色值
    public int getBlue()                       //返回蓝色值
    public int getRGB()                        //返回颜色的RGB值
    public Color brighter()                    //使颜色变浅
    public Color darker()                     //使颜色变深
}
```



2. 字体

```
public class Font implements java.io.Serializable
{
    public static final int PLAIN = 0;           // 常规
    public static final int BOLD = 1;            // 粗体
    public static final int ITALIC = 2;          // 斜体

    public Font(String name, int style, int size) // 字体名、字形、字号
    public String getName()                      // 返回字体名称
    public int getSize()                        // 返回字体大小
    public int getStyle()                      // 返回粗、斜体值
}
```

6.2 事件处理

6.2.1 委托事件模型

1. 事件和事件源

事件（**event**）是指一个状态的改变，或者一个活动的发生。
产生事件的组件称为**事件源**（**event source**）。

2. 事件类和事件监听器接口

```
public interface ActionListener extends  
    EventListener           //动作事件监听器接口  
{  
    public void actionPerformed(ActionEvent ev);  
                                //动作事件处理方法  
}
```



窗口事件监听器接口

```
public interface WindowListener extends EventListener
{
    public abstract void windowOpened(WindowEvent ev); //打开后
    public abstract void windowClosing(WindowEvent ev); //关闭时
    public abstract void windowClosed(WindowEvent ev) ; //关闭后
    public abstract void windowIconified(WindowEvent ev); //最小化
    public abstract void windowDeiconified(WindowEvent ev); //恢复
    public abstract void windowActivated(WindowEvent ev) ; //激活
    public abstract void windowDeactivated(WindowEvent ev) ;
                                     //变为不活动
}
```



3. 组件注册事件监听器对象

```
public class Button extends Component
    implements Accessibl
{
    public void addActionListener(ActionListener l)
        //注册动作事件监听器
    public void removeActionListener(ActionListener l)
        //取消注册动作事件监听器
}
button.addActionListener(this);
```



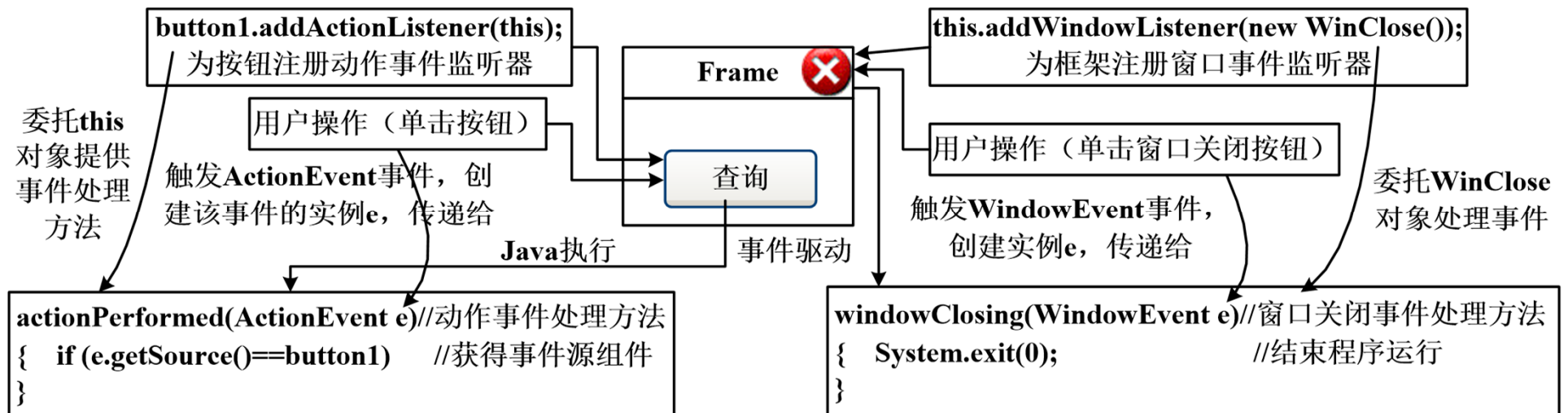
窗口对象注册窗口事件监听器

```
public class Window extends Container
    implements Accessible           // 窗口类
{
    void addWindowListener(WindowListener l)
                                   // 注册窗口事件监听器
    void removeWindowListener(WindowListener l)
                                   // 取消窗口事件监听器
}
frame.addWindowListener(this);
```

【例6.2】 Unicode字符查询

器。

1. 按钮动作事件的响应和处理，由当前类处理
2. 关闭窗口事件的响应和处理，委托**WinClose**类处理

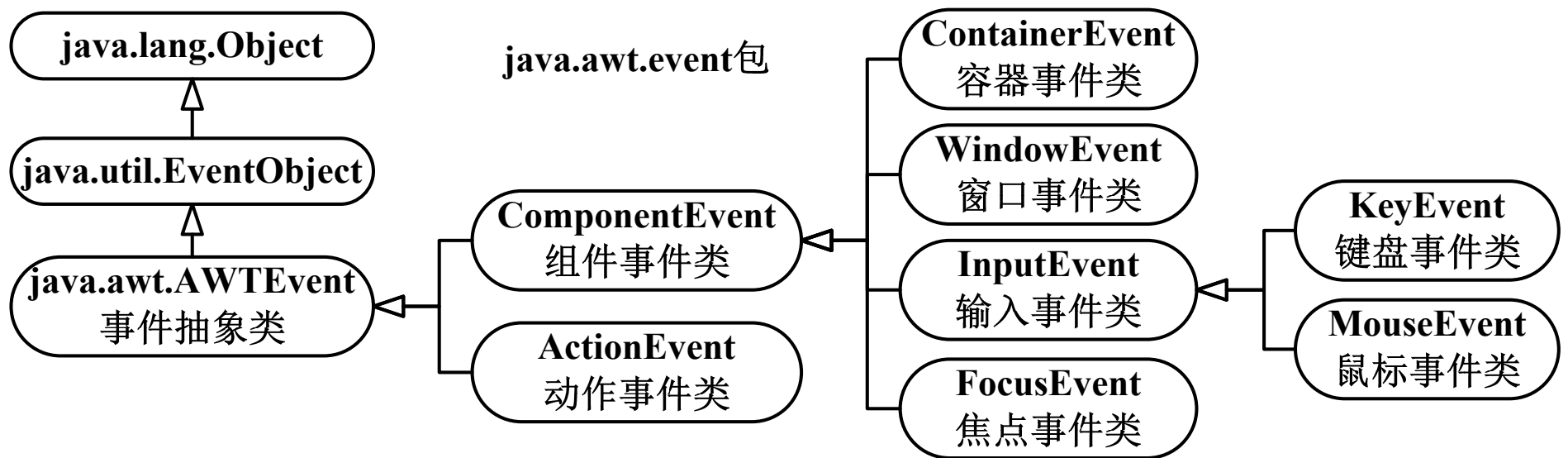


4. 图形用户界面的运行由事件驱动

- ① 不同组件的事件或同一组件的不同事件具有约定的执行次序。
- ② 各组件的事件处理方法是否执行以及执行次序，取决于用户操作。
- ③ 在一个事件处理方法中，程序按照语句的书写次序顺序执行。

6.2.2 AWT事件类和事件监听器接口

1. AWT事件类





AWT事件类

```
public class EventObject implements java.io.Serializable
{
    public Object getSource()           // 返回产生事件的事件源组件
    public String toString()           // 返回事件对象信息
}

public class ActionEvent extends AWTEvent
{
    public String getActionCommand()    // 获得按钮的标签
}
```



2. AWT事件监听器接口

- | | |
|-------------------------------|--------|
| 1. WindowListener | 窗口事件 |
| 2. ActionListener | 动作事件 |
| 3. MouseMotionListener | 鼠标移动事件 |
| 4. MouseListener | 鼠标事件 |
| 5. KeyListener | 键盘事件 |
| 6. FocusListener | 焦点事件 |

事件监听器接口的意义



4. AWT组件类中注册事件监听器的方法

```
public abstract class Component extends Object
    implements ImageObserver, MenuContainer,
    Serializable
{
    public void addKeyListener(KeyListener l)
    public void addMouseListener(MouseListener l)
    public void
        addMouseMotionListener(MouseMotionListener l)
}
```



6.3 Swing组件及事件

6.3.1 Swing组件与布局

6.3.2 文本显示和编辑组件

6.3.3 按钮组件

6.3.4 列表框和组合框组件

6.3.5 中间容器

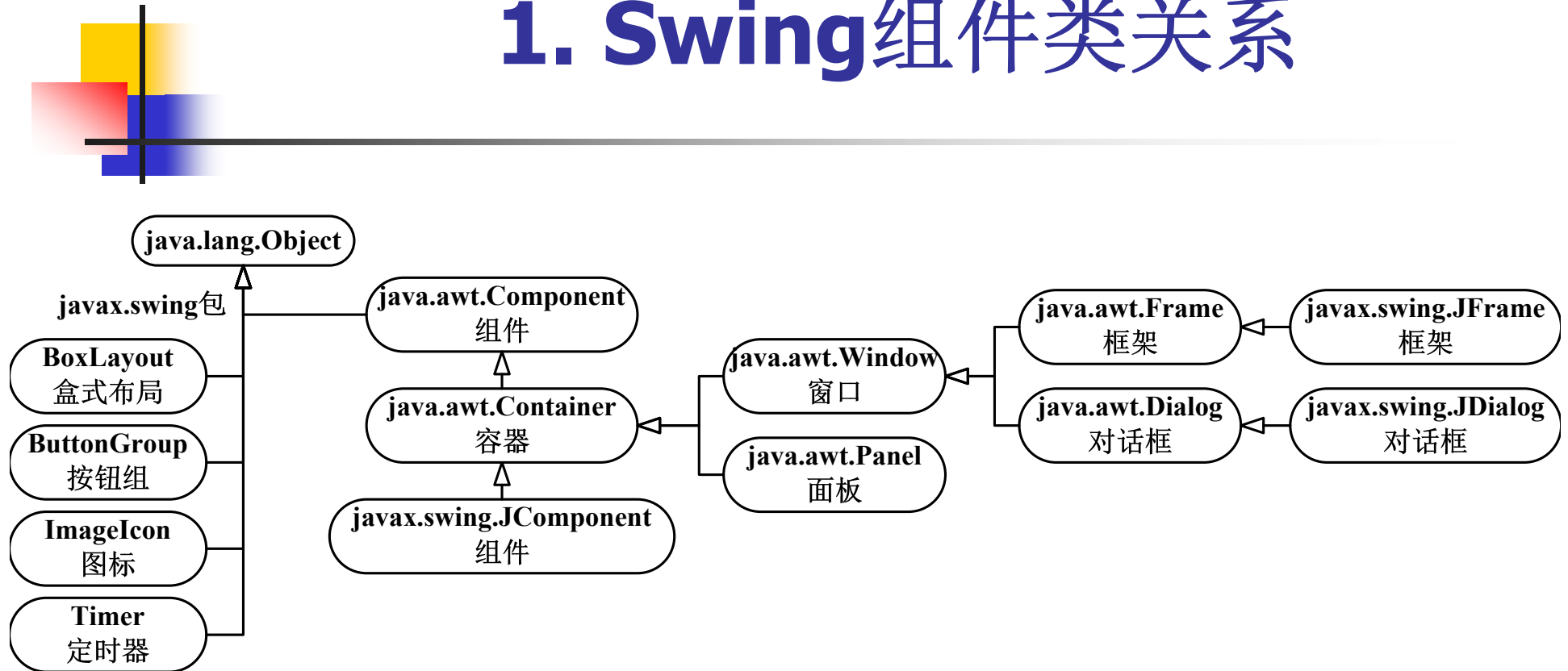
6.3.6 JOptionPane和JColorChooser对话框

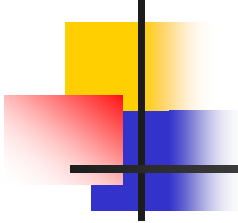
6.3.7 菜单组件

6.3.8 表格

6.3.1 Swing组件与布局

1. Swing组件类关系







2. Swing顶层容器 (JFrame、JDialog)

1. 内容窗格

```
public Container getContentPane()
```

2. 窗口关闭方式

```
public void setDefaultCloseOperation(int operation)
```

```
public interface WindowConstants
```

```
{
```

```
    public static final int DO_NOTHING_ON_CLOSE = 0; //什么也不做
```

```
    public static final int HIDE_ON_CLOSE = 1; //隐藏窗口
```

```
    public static final int DISPOSE_ON_CLOSE = 2; //释放窗口
```

```
    public static final int EXIT_ON_CLOSE = 3; //结束程序运行
```

```
}
```

```
jframe.setDefaultCloseOperation(EXIT_ON_CLOSE); //结束程序运行
```

```
jdiallog.setDefaultCloseOperation(HIDE_ON_CLOSE); //隐藏对话框
```



对话框

```
public class JDialog extends Dialog
    implements WindowConstants, Accessible,
    RootPaneContainer
{
    public JDialog(Frame owner) //owner指明拥有对
                                话框的框架，默认BorderLayout布局
    public JDialog(Frame owner, String title)
                                //title是对话框的窗口标题
    public JDialog(Frame owner, boolean modal)
                                //modal指明该对话框是否为模式窗口
    public JDialog(Frame owner, String title, boolean
    modal)
}
```

3. BorderLayout盒式布局 and Box容器



```
public class BoxLayout extends Object implements  
LayoutManager2, Serializable
```

```
{  
    public static final int X_AXIS        // 水平轴常量  
    public static final int Y_AXIS        // 垂直轴常量  
    public BoxLayout(Container target, int axis)  
}
```

```
public class Box extends JComponent implements  
Accessible  
{  
    public Box(int axis)  
}
```



6.3.2 文本显示和编辑组件

1. 标签

```
public JLabel(String text, Icon icon, int align)
    //构造方法，icon指定图标

public Icon getIcon()           //获得图标
public void setIcon(Icon icon) //设置图标
```

2. Icon图标

```
public class ImageIcon extends Object implements
    Icon, Serializable, Accessible //图标类
{
    ImageIcon(String filename)
        //filename指定图标文件名
}
```



文本行和文本区

```
public abstract class JTextComponent extends JComponent
    implements Scrollable, Accessible
{
    public String getText()                //获得文本行中的内容
    public void setText(String text)       //设置文本行中的内容
    public void setEditable(boolean edit) //设置文本行是否可编辑
    public boolean isEditable()            //判断文本行是否可编辑
}
```

3. 文本行 **JTextField**

4. 文本区 **JTextArea**

```
public class JTextArea extends JTextComponent
{
    public JTextArea()
    public JTextArea(String text)          //text指定初始显示文本
    public JTextArea(int rows, int columns) //指定行数和列数
    public JTextArea(String text, int rows, int columns)
}
```



5. Swing文本编辑事件

- 文本编辑事件类

javax.swing.event.CaretEvent

public void addCaretListener(CaretListener l)

**public interface CaretListener extends
EventListener** //文本编辑事件监听器接口

{

public void caretUpdate(CaretEvent ev);

}

- 动作事件类**ActionEvent**



【例6.3】 金额的中文大写形式。

1. 文本行响应文本编辑事件
2. 处理输入错误
3. 使用对话框
4. 实例内部类表示对象嵌套

【思考题6-2】 修改**MessageJDialog**类声明，或者**static**，或者声明为公有外部类。



6. 微调文本行组件及事件

```
public class JSpinner extends JComponent implements Accessible
{
    public JSpinner()
    public Object getValue()                // 获得值
    public void setValue(Object value)      // 设置值
    public void addChangeListener(ChangeListener l)
    public void removeChangeListener(ChangeListener l)
}

public interface ChangeListener extends EventListener
{
    public void stateChanged(ChangeEvent ev);
}
```




6.3.4 按钮组件

1. 按钮 **JButton**

public JButton(String text, Icon icon)

2. 复选框

**public class JCheckBox extends JToggleButton
implements Accessible**

{

public JCheckBox (String text)

public JCheckBox (String text, boolean selected)

**public JCheckBox (String text, Icon icon, boolean
selected)**

}



3. 单选按钮

```
public class JRadioButton extends JToggleButton
    implements Accessible
{
    public JRadioButton (String text)
    public JRadioButton (String text, boolean selected)
    public JRadioButton (String text, Icon icon,
                        boolean selected) //icon指定图标
}

public class ButtonGroup implements Serializable //按钮组类
{
    public ButtonGroup()
    public void add(AbstractButton button)    //添加按钮
    public void remove(AbstractButton button)
}
```



4. 抽象按钮类 **AbstractButton**

```
public abstract class AbstractButton extends JComponent
    implements ItemSelectable, SwingConstants
{
    public String getText()
    public void setText(String text)
    public boolean isSelected() //选中状态
    public void setSelected(boolean selected) //选中状态
    public Object[] getSelectedObjects() //选中对象数组

    public void addActionListener(ActionListener l) //动作事件
    public void removeActionListener(ActionListener l)
}
```



6.3.4 列表框和组合框

1. 列表框及其事件

(1) JList列表框

```
public class JList<T> extends JComponent
    implements Scrollable, Accessible    //列表框类
{
    public JList(T items[])           //由对象数组提供数据项
    public JList(ListModel<T> listmodel) //指定列表框模型
    public void setSelectedIndex(int i) //选中第i (≥0) 项
    public int getSelectedIndex()      //返回首个选中项序号
    public T getSelectedValue()        //返回首个选中项对象
    void addListSelectionListener(ListSelectionListener l)
                                         //注册列表框选择事件监听器
}
```



(2) 列表框模型

//默认列表框模型类

```
public class DefaultListModel<T> extends AbstractListModel<T>
{
    public DefaultListModel()
    public int getSize() //返回列表框的数据项数
    public T getElementAt(int i) //返回序号为i的组件
    public void setElementAt(T item, int i) //设置序号为i的组件为item
    public int indexOf(Object item) //返回item首次出现位置
    public void insertElementAt(T item, int i) //在i处插入item数据项
    public void addElement(T item) //添加item数据项
    public void removeElementAt(int i) //删除序号为i的数据项
    public boolean removeElement(T item) //删除首次出现的item项
    public void removeAllElements() //删除所有数据项
}
```



(3) 列表框选择事件

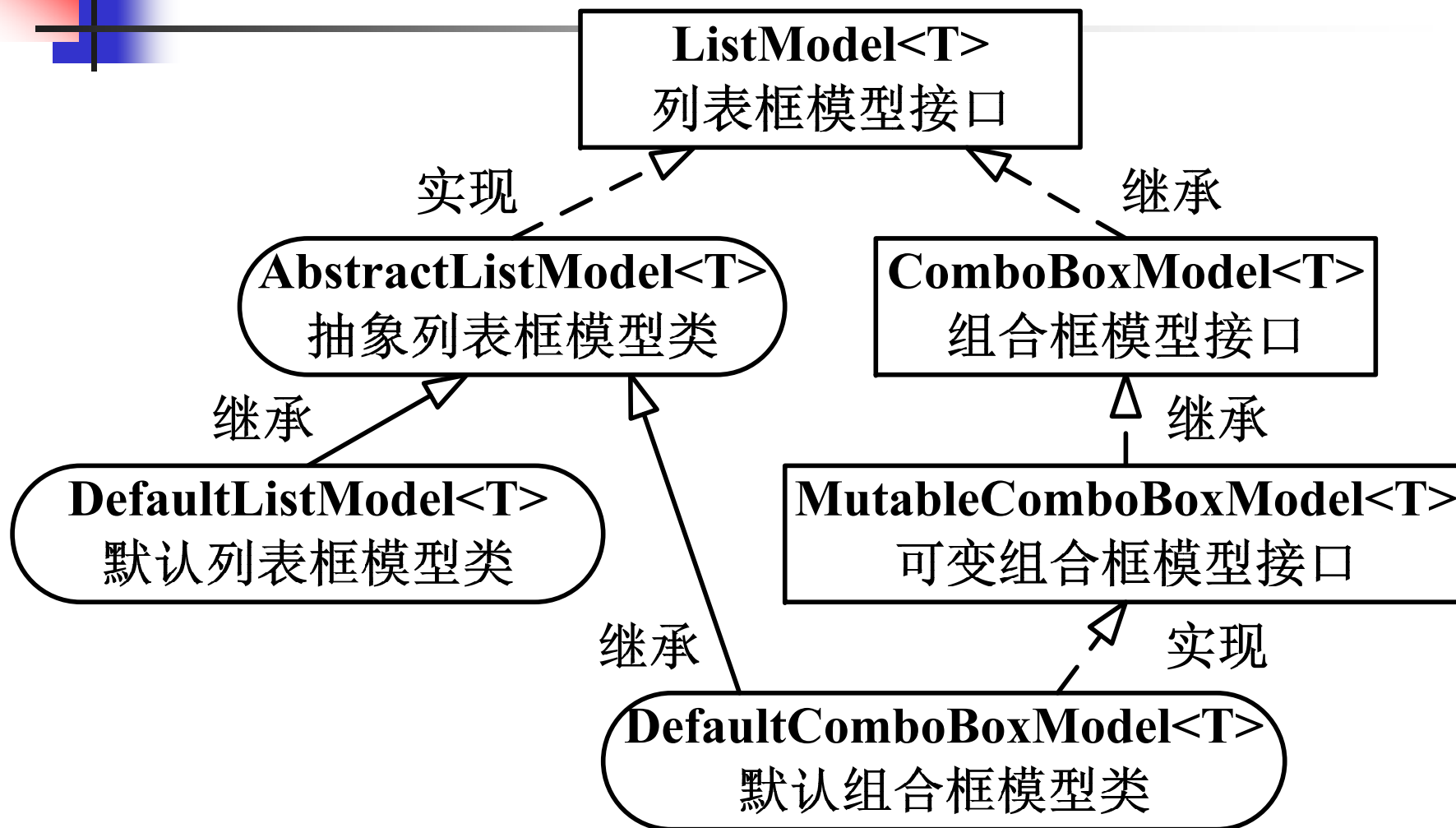
```
public interface ListSelectionListener
    extends EventListener
    // 列表框选择事件监听器接口
{
    public abstract void
        valueChanged(ListSelectionEvent ev);
        // 列表框选择事件处理方法
}
```

2. 组合框

(1) JComboBox组合框

```
public class JComboBox<T> extends JComponent implements
    ItemSelectable, ListDataListener, ActionListener, Accessible
{
    public JComboBox()
    public JComboBox(T items[])           // 由对象数组提供数据项
    public void setEditable(boolean edit ) // 是否可编辑，默认false
    public Object getSelectedItem()        // 返回选中数据项对象
    public void setSelectedItem(Object item) // 设置选中项
    public int getSelectedIndex()           // 返回选中数据项序号
    public void setSelectedIndex(int i)     // 选中第i ( $\geq 0$ ) 项
    public void addItem(T item)            // 添加数据项
    public void removeItem(Object item)     // 删除数据项
    public void removeItemAt(int i)         // 删除第i ( $\geq 0$ ) 项
    public void removeAllItems()            // 删除所有项，触发动作事件
    public void addActionListener(ActionListener listener)
}
```

列表框模型和组合框模型类关系





(2) 组合框响应动作事件

1. 在下拉列表中选择数据项，
e.getActionCommand()返回
“**comboBoxChanged**”；
2. 在文本行中按**Enter**键，
e.getActionCommand()返回
“**comboBoxEdited**”。



组合框触发**ActionEvent**事件

1. 首次调用**addItem(T item)**方法添加数据项时；
2. 调用**removeItem(Object obj)**、**removeItemAt(int i)**或**removeAllItems()**方法，删除最后仅剩一项时；
3. 调用**setSelectedIndex()**、**setSelectedItem(Object obj)**设置指定数据项为选中状态。



6.3.5 中间容器

1. 滚动窗格

```
public class JScrollPane extends  
    JComponent implements  
    ScrollPaneConstants, Accessible  
{  
    public JScrollPane(Component view)  
}
```



2. 分割窗格

```
public class JSplitPane extends JComponent implements
    Accessible
{
    public final static int VERTICAL_SPLIT = 0;    //垂直分割常量
    public final static int HORIZONTAL_SPLIT = 1; //水平分割常量

    public JSplitPane()                        //创建分割窗格
    public JSplitPane(int orientation)         //参数指定分割方向

    public void setDividerLocation(int location) //设置分割条位置
    public void setOneTouchExpandable(boolean expand)
        //当expand为true时, 提供一键展开按钮, 快速展开/折
        叠分隔条; 默认false
}
```



6.3.6 JOptionPane和JColorChooser对话框

1. JOptionPane对话框

① 消息对话框

```
JOptionPane.showMessageDialog(this, "请重新输入!");
```

② 确认对话框

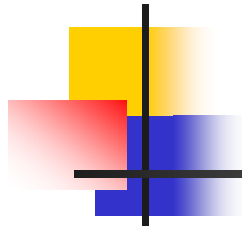
```
int i=JOptionPane.showConfirmDialog(this, "终止当前程序运行?");
```

③ 输入对话框

```
String password = JOptionPane.showInputDialog(this, "密码");
```

2. JColorChooser选择颜色对话框

```
public static Color showDialog(Component parent,  
String title, Color color )
```



【例6.4】 输入用户信息。

1. PersonJPanel面板

2. Person对象信息管理框架

① **equals(Object)**方法比较所有成员变量值，以区分对象

this.listmodel.removeElement(this.person.get());

② 查找操作只比较对象的某个或某些成员变量值，作为查找依据。在列表框模型数据项中查找**obj**对象：

public <T> void search(T obj, Equalable<? super T> e)

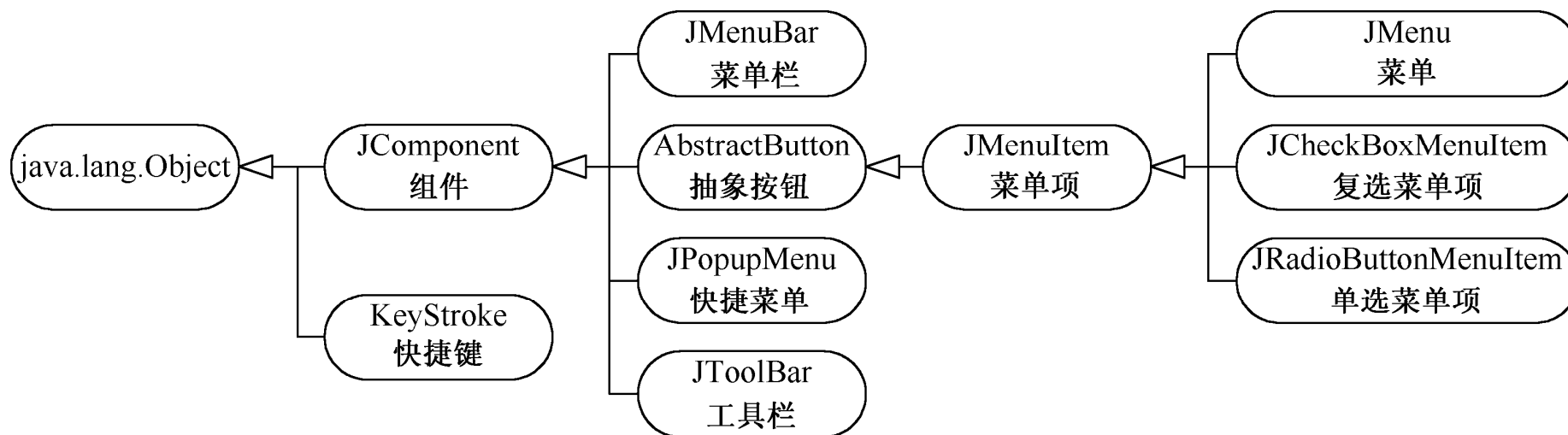
③ 排序。将列表框模型数据项排序：

public <T> void sort(Comparator<? super T> c)

6.3.7 菜单组件

1. 使用菜单的两种方式

- ① 窗口菜单
- ② 快捷菜单





2. 菜单栏

```
public class JMenuBar extends JComponent  
    implements Accessible, MenuElement  
{  
    public JMenuBar()  
    public JMenu add(JMenu menu) //添加菜单  
}
```

JFrame类提供**setJMenuBar()**方法将菜单栏放置在框架窗口上方，该方法声明如下：

```
public void setJMenuBar(JMenuBar menubar)
```




3. 菜单

```
public class JMenu extends JMenuItem
    implements Accessible, MenuElement
{
    public JMenuItem(String text)    // 菜单标题
    public JMenuItem add(JMenuItem item)
                                    // 添加菜单项
    public void addSeparator()      // 添加分隔线
    public JMenuItem insert(JMenuItem item, int i)
    public void insert(String text, int i)
    public void remove(JMenuItem item)
}
```



4. 菜单项

```
public class JMenuItem extends AbstractButton
    implements Accessible, MenuElement
{
    public JMenuItem(String text)    // 菜单标题
    public JMenuItem(String text, Icon icon)
                                    // icon指定菜单图标
    public void setAccelerator(KeyStroke keyStroke)
                                    // 设置快捷键
}
```



5. 选择菜单项（复选菜单项）

```
public class JCheckBoxMenuItem extends  
    JMenuItem implements SwingConstants,  
    Accessible  
{  
    public JCheckBoxMenuItem(String text)  
    public JCheckBoxMenuItem(String text,  
                                boolean selected)  
    public JCheckBoxMenuItem(String text, Icon icon,  
                                boolean selected )  
}
```



5. 选择菜单项（单选菜单项）

```
public class JRadioButtonMenuItem extends  
    JMenuItem implements Accessible  
{  
    public JRadioButtonMenuItem(String text)  
    public JRadioButtonMenuItem(String text,  
                                boolean selected)  
    public JRadioButtonMenuItem(String text,  
                                Icon icon, boolean selected)  
}
```



6. 快捷菜单

public class JPopupMenu extends JComponent implements Accessible, MenuElement

{

public JPopupMenu()

public JMenuItem add(JMenuItem item) // 添加菜单项

public void addSeparator() // 添加分隔线

public void show(Component invoker, int x, int y)

 // 在(x,y)位置处显示快捷菜单，**invoker**指定快捷菜单所依附的组件

}

java.awt.Component组件类提供下列**add()**方法用于任何组件添加快捷菜单：

public void add(PopupMenu popup) // 添加快捷菜单



7. 工具栏

```
public class JToolBar extends JComponent
    implements SwingConstants, Accessible
{
    public JToolBar()           //默认水平方向
    public void addSeparator()  //添加分隔线
}
```

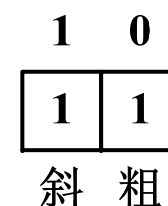
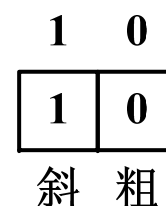
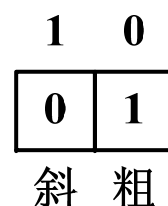
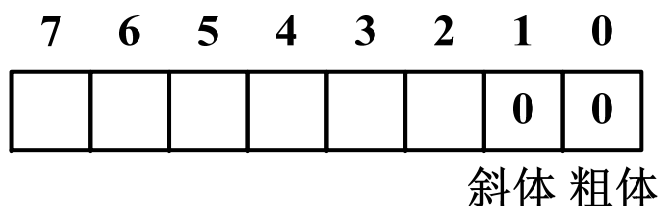


鼠标事件监听器接口

```
public interface MouseListener extends java.util.EventListener
    // 鼠标事件监听器接口
{
    public abstract void mouseClicked(MouseEvent ev); // 单击鼠标
    public abstract void mousePressed(MouseEvent ev); // 按下鼠标
    public abstract void mouseReleased(MouseEvent ev); // 放开鼠标
    public abstract void mouseEntered(MouseEvent ev); // 鼠标进入
    public abstract void mouseExited(MouseEvent ev); // 鼠标离开
}
public class MouseEvent extends InputEvent // 鼠标事件类
{
    public int getX() // 返回鼠标单击点的X坐标值
    public int getY() // 返回鼠标单击点的Y坐标值
    public int getButton()
        // 返回单击鼠标的键值，值为1、2、3分别对应鼠标左键、中键、右键
}
}
```

【例6.5】文本编辑器。

1. 获得本地系统的所有字体
2. 使用位运算更改字形



(a) 字形整数最低位结构, 常规字形值为0 (b) 粗体, 1 (c) 斜体, 2 (d) 粗体且斜体, 3

3. 字号组合框区分选择和编辑操作
4. 字号组合框添加不重复数据项
5. 当字号值不合适时抛出异常
6. 使用窗口菜单和快捷菜单



6.3.8 表格

```
public class JTable extends JComponent
{
    public JTable()                //构造空表格
    public JTable(Object[][] data, Object[] title)
        //data指定数据，title指定列标题
    public JTable(TableModel model)
        //构造指定表格模型的表格
    public int getSelectedRow();
        //返回表格当前选中行号，没有选定列时返回-1
}
```



默认表格模型类

```
public class DefaultTableModel extends AbstractTableModel
    implements Serializable
{
    public DefaultTableModel()
    public DefaultTableModel(int rowCount, int columnCount) //行列数
    public DefaultTableModel(Object titles[], int rowCount) //列标题
    public DefaultTableModel(Object data[][], Object titles[])
                                //指定数据项和列标题

    public int getRowCount() //返回行数
    public void setRowCount(int rowCount) //设置表格模型行数
    public Object getValueAt(int row, int column) //返回单元值
    public void setValueAt(Object value, int row, int column)
    public void addRow(Object data[]) //添加一行, data指定各列值
    public void removeRow(int row) //删除指定行
}
```



【例6.6】 银行贷款按月还本付息的计算。

等额本金还款法：

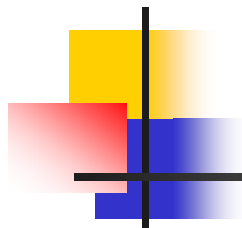
月还本金 = 贷款金额 ÷ (贷款年限***12**月)

本金余额 = 贷款金额 - 累计已还本金

月还利息 = 贷款利率 × 本金余额

月还本息 = 月还本金 + 月还利息

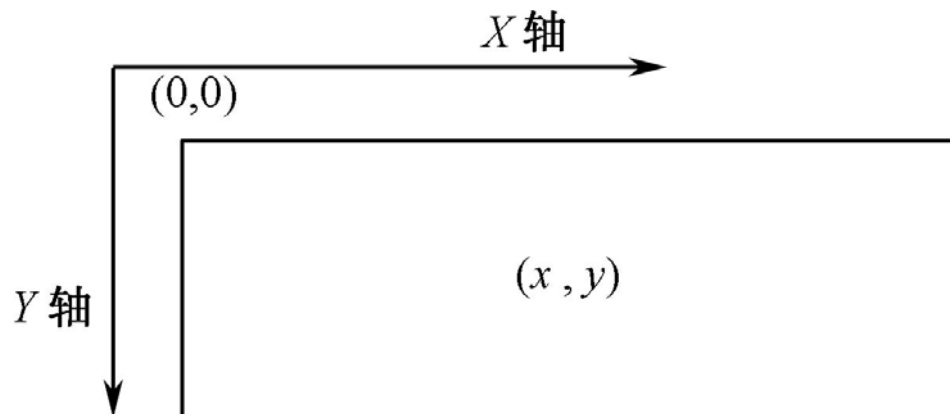
$$\text{每月还款额} = \frac{\text{贷款本金}}{\text{贷款期月数}} + (\text{贷款本金} - \text{已归还本金累计额}) \times \text{月利率}$$



6.4 图形图像

6.4.1 图形设计

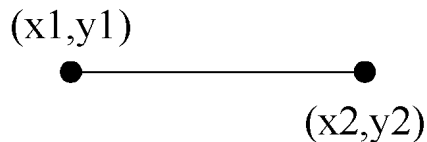
1. 绘图坐标系
2. 绘图类
3. 组件绘图方法
4. 画布
5. 重新绘制图形



2. 绘图类

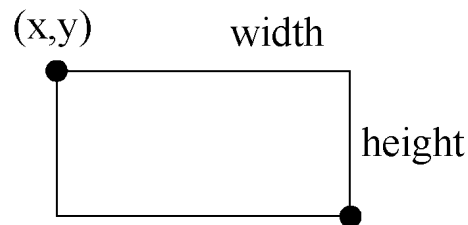
public abstract class Graphics extends Object

```
{  
    public abstract Color getColor();           // 获取当前颜色  
    public abstract void setColor(Color c);     // 设置颜色  
    public abstract void drawLine(int x1, int y1, int x2, int y2); // 画直线  
    public void drawRect(int x, int y, int width, int height) // 画矩形  
    public abstract void fillRect(int x, int y, int width, int height); // 填充矩形  
    public abstract void drawOval(int x, int y, int width, int height); // 画椭圆  
    public abstract void drawString(String str, int x, int y); // 显示字符串  
    public abstract Font getFont();           // 获得颜色  
    public abstract void setFont(Font font);   // 设置颜色  
}
```



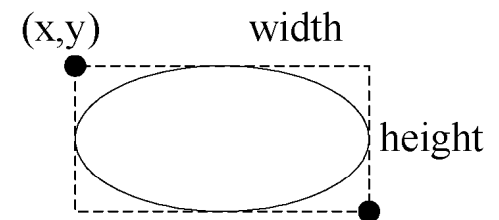
drawLine(x1, y1, x2, y2);

(a) 直线



drawRect(x, y, width, height);

(b) 矩形



drawOval(x, y, width, height);

(c) 椭圆



3. 组件绘图方法

public void paint(Graphics g) // 组件绘制图形

4. 画布

public Canvas()

5. 重新绘制图形

public void repaint() // 调用paint()方法刷新图形

【例6.7】 四叶玫瑰线的图形设计。

$$r = a \sin 2\theta$$



鼠标移动事件监听器接口

```
public interface MouseEventListener extends  
    java.util.EventListener  
{  
    public abstract void  
        mouseDragged(MouseEvent ev); // 鼠标拖动  
    public abstract void  
        mouseMoved(MouseEvent ev); // 鼠标移动  
}
```

【例6.8】 使用鼠标画直线。



6.4.2 图像

1. Image图像及其子类BufferedImage

```
public abstract class Image extends Object
{
    public abstract Graphics getGraphics()
        // 获得在图像上绘图的Graphics对象
}
```

2. Graphics支持图形图像

```
public abstract boolean drawImage(Image
    img, int x, int y, ImageObserver observer);
```




实验6 图形用户界面设计

- **目的：**设计具有图形用户界面的、能够响应事件的**Java**应用程序。
- **要求：**掌握**Java Swing**组件的使用方法，理解委托事件处理模型，掌握多种布局方式，掌握窗口菜单和快捷菜单设计方式，熟悉在组件上绘图的方法。
- **重点：**图形用户界面，响应事件。
- **难点：**事件处理。