第三章

软件开发也要拼脸蛋—UI开发的点点滴滴

主讲：王海

本章目标

- 了解Android中的UI元素
- 能够使用布局管理器对界面进行管理
- 掌握界面交互事件处理机制及实现步骤
- 能够熟练使用常用的Widget简单组件
- 掌握Dialog对话框的使用

Android UI元素

Android中界面元素：

- **视图**：所有可视界面元素（通常称为控件或小组件）的基类
- **视图容器**：视图类的扩展，其中包含多个子视图
- **布局管理**：管理组件的布局格式，组织界面中组件的呈现方式
- **Activity**：用于为用户呈现窗口或屏幕
- **Fragment**：针对不同屏幕尺寸时，优化UI布局以及创建可重用的UI元素

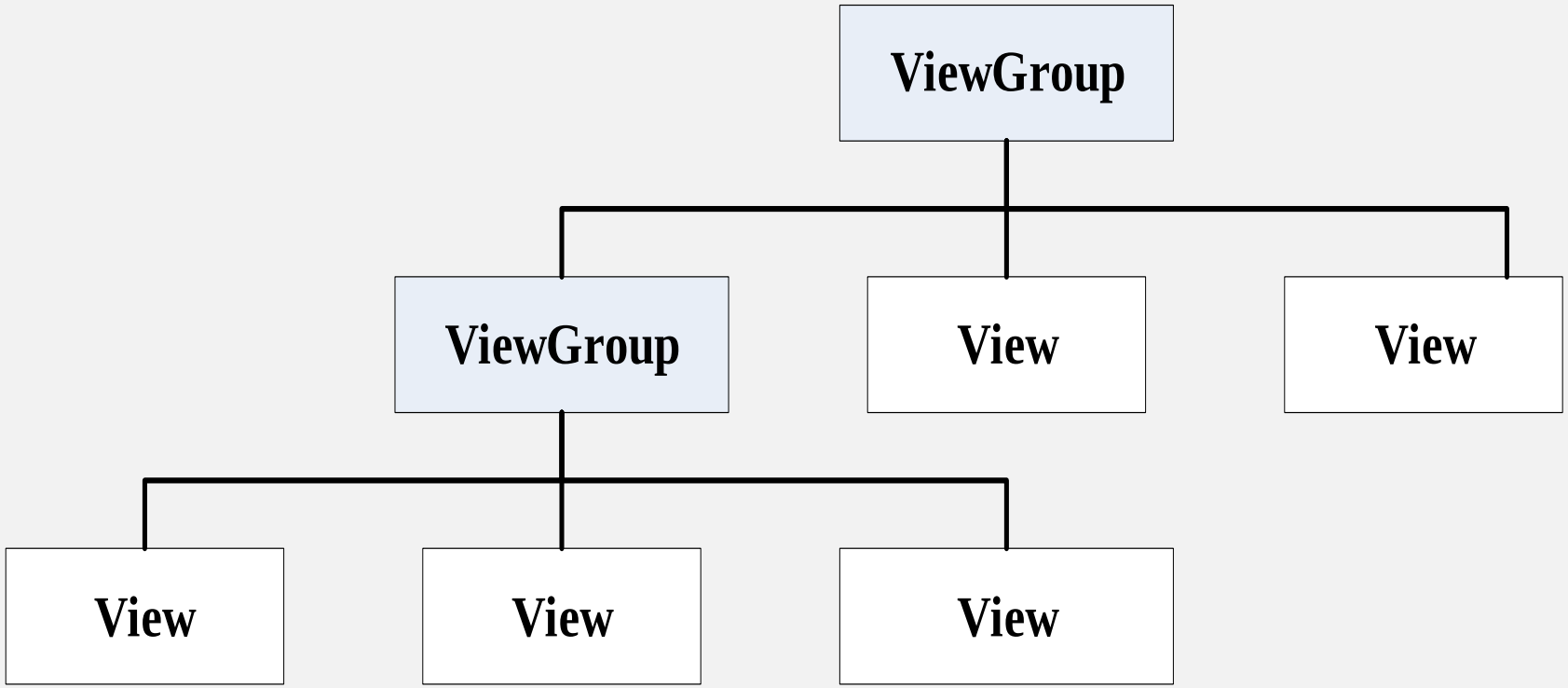
视图

- View的常见子类

类名	功能描述
TextView	文本视图
EditText	编辑文本框
Button	按钮
Checkbox	复选框
RadioGroup	单选按钮组
Spinner	下拉列表
AutoCompleteTextView	自动完成文本框
DataPicker	日期选择器
TimePicker	时间选择器
DigitalClock	数字时钟
AnalogClock	模拟时钟
ProgessBar	进度条
RatingBar	评分条
SeekBar	搜索条
GridView	网格视图
LsitView	列表视图
ScrollView	滚动视图

视图容器

- ViewGroup类通常作为其他组件的容器使用



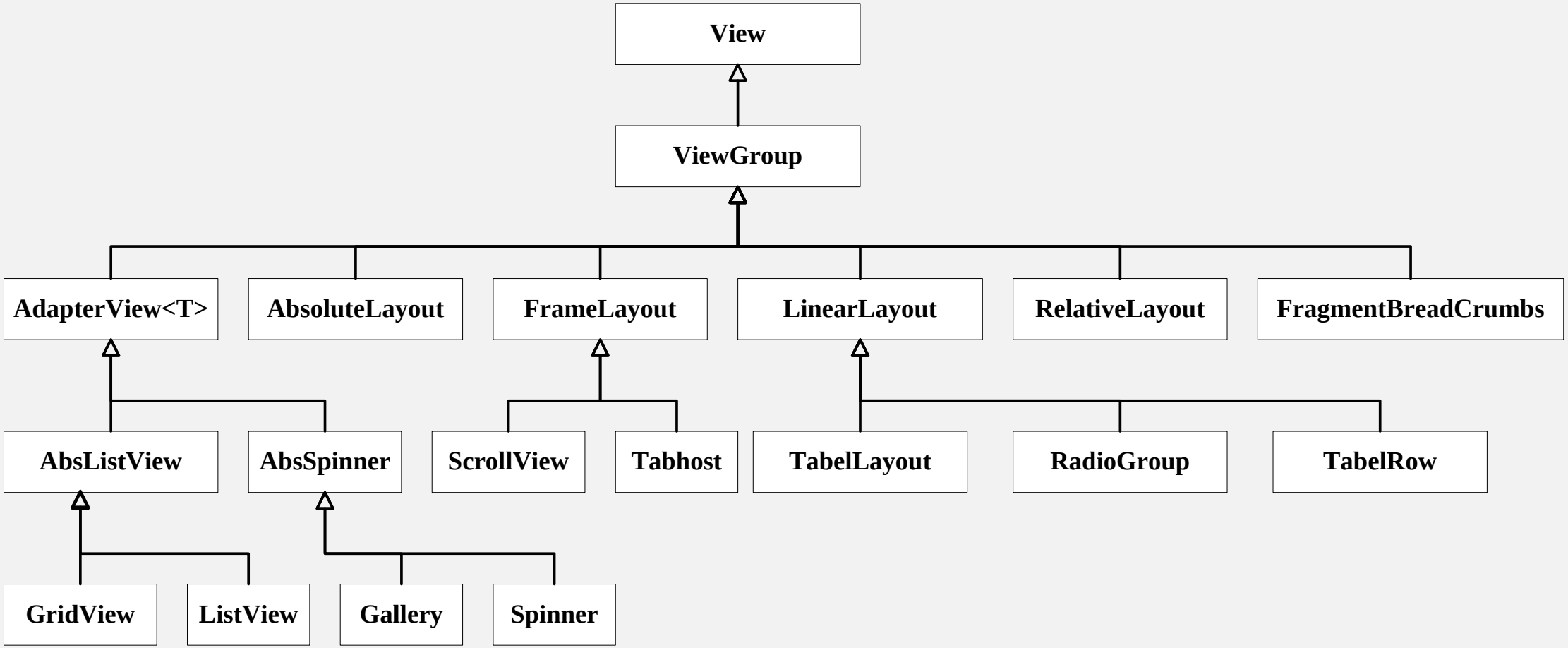
ViewGroup类提供的主要方法

类名	功能描述
ViewGroup()	构造方法
void addView(View child)	用于添加子视图，以View作为参数，将该View增加到视图组中
removeView(View view)	将指定的View从视图组中移除
updateViewLayout(View view, ViewGroup.LayoutParams params)	用于更新某个View的布局
void bringChildToFront(View child)	将参数所指定的视图移动到所有视图之前显示
boolean clearChildFocus(View child)	清除参数所指定的视图的焦点
boolean dispatchKeyEvent(KeyEvent event)	将参数所指定的键盘事件分发给当前焦点路径的视图。
boolean dispatchPopulateAccessibilityEvent(AccessibilityEvent event)	将参数所指定的事件分发给当前焦点路径的视图
boolean dispatchSetSelected(boolean selected)	为所有的子视图调用setSelected()方法

ViewGroup继承了View类，虽然可以当成普通的View来使用，但习惯上将ViewGroup当容器来使用。由于ViewGroup是一个抽象类，在实际应用中通常使用ViewGroup的子类作为容器，例如各种布局管理器。

ViewGroup继承结构

- ViewGroup的继承者大部分位于android.widget包中。



布局参数类

在Android布局文件中，每个组件所能使用的XML属性有三类：

- 组件本身的XML属性；
- 组件祖先类的XML属性；
- 组件所属容器的布局参数。

ViewGroup容器使用两个内部类来控制子组件在其中的分布位置:

- ViewGroup.LayoutParams
- ViewGroup.MarginLayoutParams

XML属性	功能描述
android:layout_width	设定该组件的子组件布局的宽度
android:layout_height	设定该组件的子组件布局的高度

布局参数类

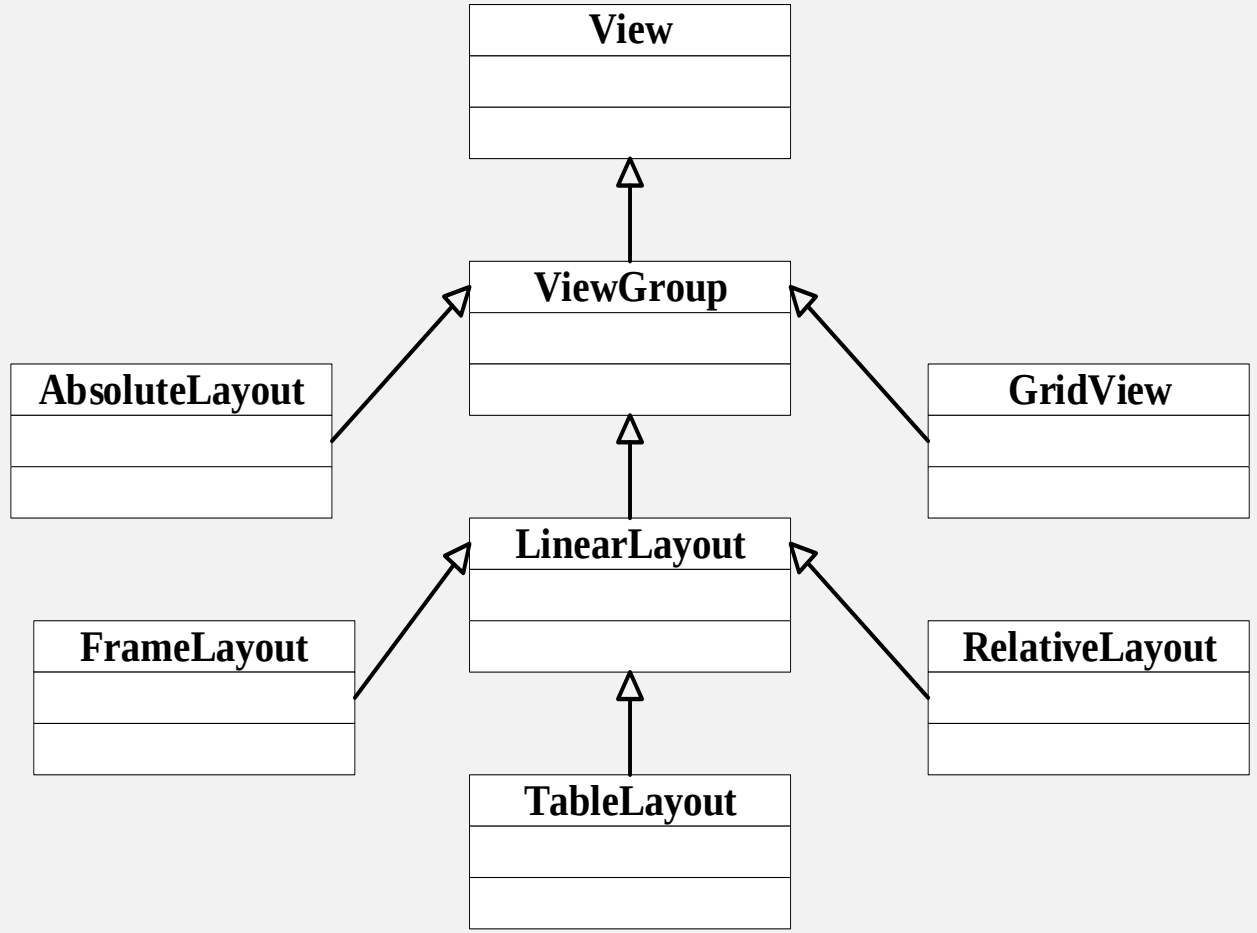
- ViewGroup.MarginLayoutParams用于控制子组件周围的页边距

XML属性	功能描述
android:layout_marginTop	指定该子组件上面的页边距
android:layout_marginRight	指定该子组件右面的页边距
android:layout_marginBottom	指定该子组件下面的页边距
android:layout_marginLeft	指定该子组件左面的页边距

由于LayoutParams也具有继承关系，因此LinearLayout的子类除了可以使用LinearLayout.LayoutParams所提供的XML属性外，还可以使用其祖先类ViewGroup.LayoutParams的XML属性。

布局管理

- 布局管理器可以根据运行平台来调整组件的大小
- 所有的布局管理器都是ViewGroup的子类



Android常用布局

- LinearLayout线性布局：该布局中子元素之间成线性排列
- RelativeLayout相对布局：该布局中子元素之间根据相对位置排列
- TableLayout表格布局：该布局中子元素的位置分配到表格的行或列中
- AbsoluteLayout绝对布局：该布局中子元素按照绝对坐标进行排列

Fragment

- Fragment允许将Activity拆分成多个完全独立的可重用的组件
- 每个Fragment都是一个独立的模块
- 与绑定的Activity紧密的联系在一起
- 一个Fragment可以被多个Activity所共用

界面布局

Android中提供了两种创建布局的方式：

- 在XML布局文件中声明
- 在程序中直接实例化布局及其组件

线性布局

- LinearLayout是线性布局，布局中的组件按照垂直或者水平方向进行排列

XML属性	对应方法	功能描述
android:divider	setDividerDrawable()	设置垂直布局时两个按钮之间的分隔条
android:gravity	setGravity()	设置布局管理器内组件的对齐方式
android:orientation	setOrientation()	设置布局管理器内组件的排列方式

- LinearLayout中子元素的位置都受LinearLayout.LayoutParams控制

XML属性	功能描述
android:layout_gravity	指定子元素在LinearLayout中的对齐方式
android:layout_weight	指定子元素在LinearLayout中所占的比重

表格布局

- TableLayout类似表格形式，以行和列的方式来布局子组件
- 在TableLayout中，可以通过以下3种方式对单元格进行设置：
  - Shrinkable
  - Stretchable
  - Collapsed

XML属性	对应方法	功能描述
android:shrinkColumns	setShrinkAllColumns(boolean)	设置可收缩的列
android:stretchColumns	setStretchAllColumns(boolean)	设置可伸展的列
android:collapseColumns	setColumnCollapsed(int,boolean)	设置要隐藏的列

全局属性的设置

```
<?xml version="1.0" encoding="utf-8"?>
<TableLayout
xmlns:android="http://...d.com/apk/res/android"
    android:layout_wi...nt"
    android:layout...ent"
    android:stretch...
    android:collapseColumns="*"
    android:shrinkColumns="1,2" >
</TableLayout>
```

表示第0列可伸展

表示隐藏所有行

表示第1、2列皆可收缩

注意

列可以同时具备stretchColumns和shrinkColumns属性；当该列的内容较多时，将以“多行”方式显示其内容。

- 使用TableRow.LayoutParams对TableRow的子元素进行修饰

XML属性	功能描述
android:layout_column	指定该单元格在第几列显示
android:layout_span	指定该单元格占据的列数（未指定时，默认为1）

- 对表格属性进行设置

```
<TableLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:stretchColumns="0"
    android:collapseColumns="*"
    android:shrinkCo
   <TableRow>
    <Button  android:
    <Button  android:layout_column="1"/>
   </TableRow>
</TableLayout>
```

表示该控件占据2列

表示该控件显示在第1列

相对布局

- 在相对布局容器中子组件的位置总是相对于兄弟组件或父容器

XML属性	功能描述
android:layout_alignParentLeft	指定该组件是否与布局容器左对齐
android:layout_alignParentTop	指定该组件是否与布局容器顶端对齐
android:layout_alignParentRight	指定该组件是否与布局容器右对齐
android:layout_alignParentBottom	指定该组件是否与布局容器底端对齐
android:layout_centerInParent	指定该组件是否位于布局容器的中央位置
android:layout_centerHorizontal	指定该组件是否位于布局容器的水平居中
android:layout_centerVertical	指定该组件是否位于布局容器的垂直居中

相对布局

XML属性	功能描述
android:layout_toLeftOf	控制该组件位于指定ID组件的左侧
android:layout_toRightOf	控制该组件位于指定ID组件的右侧
android:layout_above	控制该组件位于指定ID组件的上方
android:layout_below	控制该组件位于指定ID组件的下方
android:layout_alignLeft	控制该组件与指定ID组件的左边界进行对齐
android:layout_alignTop	控制该组件与指定ID组件的上边界进行对齐
android:layout_alignRight	控制该组件与指定ID组件的右边界进行对齐

绝对布局

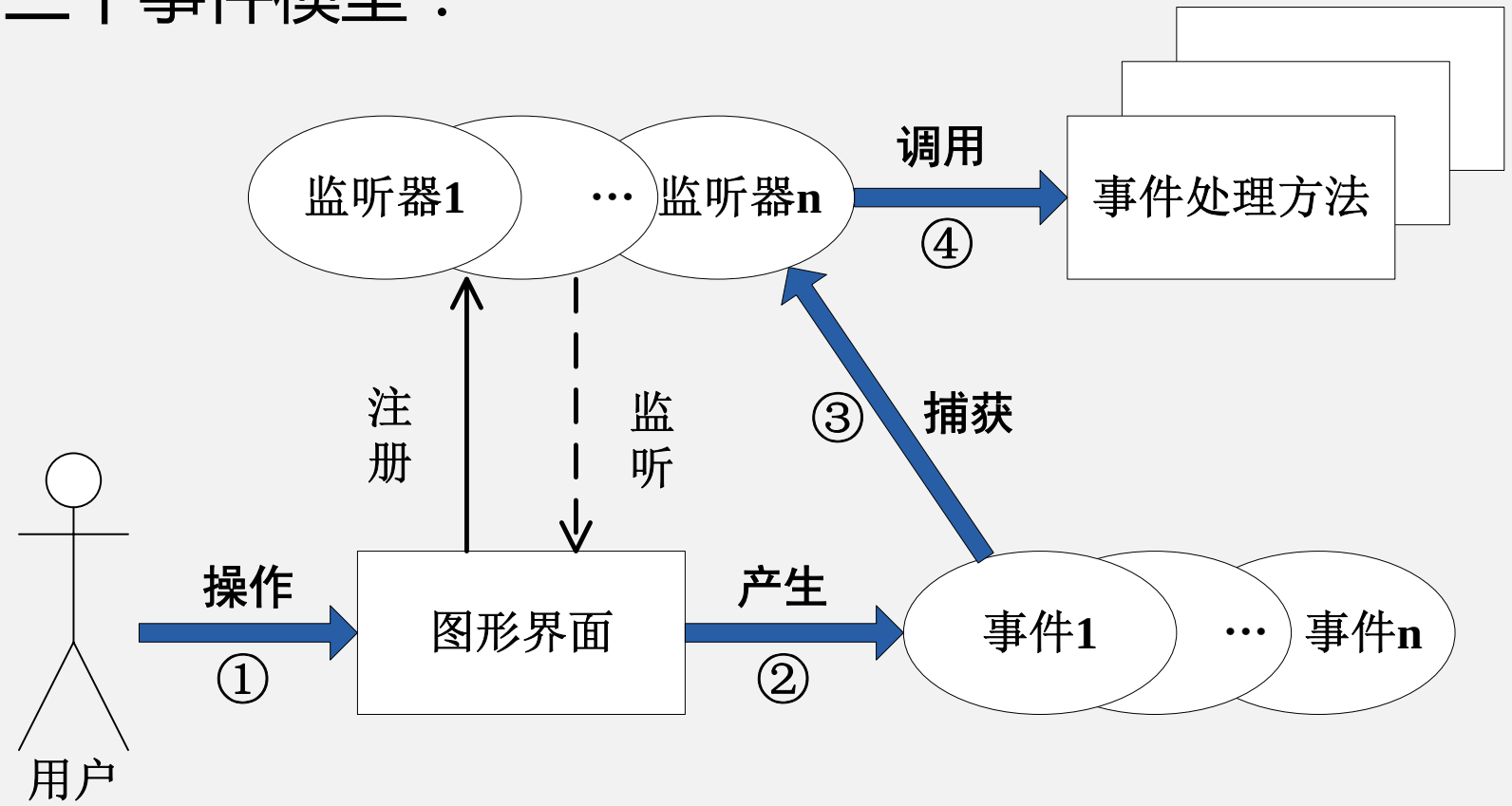
- **AbsoluteLayout通过指定组件的确切X、Y坐标来确定组件的位置**

```
<AbsoluteLayout android:id="@+id/AbsoluteLayout01"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content">
       <Button android:text="A" android:id="@+id/Button01"
       …
       android:layout_x="10dp" android:layout_y="20dp"></Button>
       <Button android:text="B" android:id="@+id/Button02"
       …
       android:layout_x="100dp" android:layout_y="20dp"></Button>
       <Button android:text="C" android:id="@+id/Button03"
       …
       android:layout_x="10dp" android:layout_y="80dp"></Button>
       <Button android:text="D" android:id="@+id/Button04"
       …
       android:layout_x="100dp" android:layout_y="80dp"></Button>
</AbsoluteLayout>
```

事件处理

Android系统中引用了三个事件模型：

- 事件
- 事件源
- 事件监听器



Android中的事件监听器

事件监听器接口	事 件	功能描述
OnClickListener	单击事件	当用户点击某个组件或者方向键触发该事件
OnFocusChangeListener	焦点事件	当组件获得或者失去焦点时触发该事件
OnKeyListener	按键事件	当用户按下或者释放设备上的某个按键触发该事件
OnTouchListener	触摸事件	当触碰屏幕时触发该事件
OnCreateContextMenuListener	创建上下文菜单事件	当创建上下文菜单时触发该事件
OnCheckedChangeListener	选项改变事件	当选择改变时触发该事件

Android中的事件监听器

实现事件监听器有以下四种形式：

- Activity本身作为事件监听器：通过Activity实现监听器接口，并实现事件处理方法
- 匿名内部类形式：使用匿名内部类创建事件监听器对象
- 内部类或外部类形式：将事件监听类定义为当前类的内部类或普通的外部类
- 绑定标签：在布局文件中为指定标签绑定事件处理方法

Android中的事件监听器

实现基于监听的事件处理有三步：

① 实现基于监听的事件处理步骤

② 在事件处理方法中编写事件处理代码

③ 在相应的组件上注册监听器

Activity本身作为事件监听器

- 通过Activity实现监听器接口，并实现该接口中对应的事件处理方法
- 基于监听的事件的处理模型的编程步骤：
  - ① 获取所要触发事件的事件源控件
  - ② 实现事件监听器类
  - ③ 调用事件源的setXxxListener()方法，将事件监听器注册给事件源对象

匿名内部类形式

- 匿名内部类形式的事件监听器更合适事件只是临时使用一次的情况

```
// 使用匿名内部类创建一个监听器
clickBtn.setOnClickListener(new OnClickListener() {

    @Override
    public void onClick(View v) {
        // 实现事件处理方法
        showTxt.setText("btn按钮被单击了！");
    }
});
```

内部类、外部类形式

- 将事件监听器定义成当前类的内部类
- 使用内部类有以下优点：
  - 可以在当前类中复用内部监听器类
  - 可以访问当前类的所有界面组件

绑定标签

- 指在界面布局文件中直接为指定标签绑定事件处理方法

```
<Button
        android:id="@+id/clickBtn"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:onClick="clickMe"
        android:text="单击我" />
```

```
public void clickMe(View v){
    // 实现事件处理方法
    showTxt.setText("btn按钮被单击了！");
}
```

基于回调机制的事件处理

- onKeyDown()
- onKeyUp()
- onTouchEvent()
- onTrackBallEvent()
- onFocusChanged()

onKeyDown()方法

```
public boolean onKeyDown (int keyCode, KeyEvent event)
```

- 用来捕捉手机键盘被按下的事件
  - 参数keyCode表示被按下的键值
  - 参数event用于封装按键事件的对象
  - 返回值为boolean类型

常量名	功能描述
KEYCODE_POWER	电源键
KEYCODE_NOTIFICATION	通知键
KEYCODE_MUTE	话筒静音键
KEYCODE_VOLUME_MUTE	扬声器静音键
KEYCODE_VOLUME_UP	音量增加键
KEYCODE_VOLUME_DOWN	音量减小键
KEYCODE_CALL	拨号键
KEYCODE_ENDCALL	挂机键

onKeyUp()方法

- 用来捕捉手机键盘按键抬起的事件

```
public boolean onKeyUp (int keyCode, KeyEvent event)
```

  - 参数keyCode表示触发事件的按键码
  - 参数event是一个事件封装类的对象
  - 返回值为boolean类型

onTouchEvent()方法

- 用来处理手机屏幕的触摸事件

```
public boolean onTouchEvent (MotionEvent event)
```

  - 参数event是手机屏幕触摸事件封装类的对象，用于封装件的相关信息
  - 返回值为boolean类型

自定义的View并不会自动刷新，所以每次改变数据模型时都需要手动调用postInvalidate()方法进行屏幕的刷新操作。

onTrackBallEvent()方法

- 用来处理手机中轨迹球事件

```
public Boolean onTrackballEvent (MotionEvent event)
```

  - 参数event为手机轨迹球事件封装类的对象
  - 返回值为boolean类型

- 轨迹球与手机键盘有一定的区别
  - » 某些型号的手机设计出的轨迹球会比只有手机键盘时更美观
  - » 轨迹球使用更为简单

在模拟器运行状态下，可以通过F6键打开模拟器的轨迹球，然后通过鼠标的移动来模拟轨迹球事件。

onFocusChanged()方法

- 焦点改变的回调方法

```
protected void onFocusChanged (
    Boolean gainFocus, int direction, Rect previouslyFocusedRect)
```

  - 参数gainFocus表示触发该事件的View是否获得了焦点
  - 参数direction表示焦点移动的方向
  - 参数previouslyFocusedRect表示在触发事件的View的坐标系中，前一个获得焦点的矩形区域

每按下一次按键，会调用两次onFocusChanged()方法，一次是某个按钮失去焦点时调用，第二次是另一个按钮获得焦点时调用。

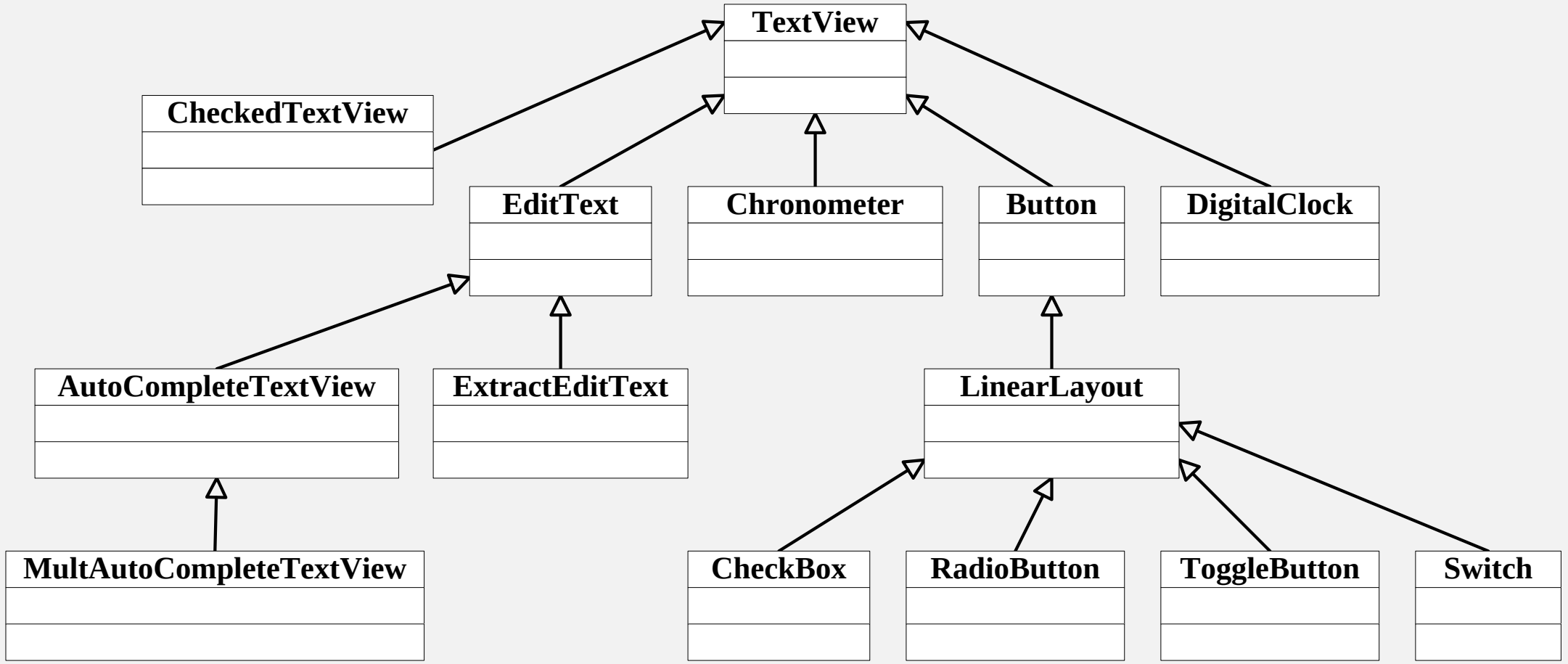
常见的焦点相关方法

方法	功能描述
setFocusable()	用于设置View是否可以拥有焦点
isFocusable()	用于判断View是否可以拥有焦点
setNextFocusDownId()	用于设置View的焦点向下移动后获得焦点View的ID
hasFocus()	用于判断View的父控件是否获得了焦点
requestFocus()	用于尝试让此View获得焦点
isFocusableTouchMode()	用于设置View是否可以在触摸模式下获得焦点，默认情况下不可用

Widget组件通用属性

属性名称	功能描述
android:id	设置控件的索引
android:layout_height	设置布局高度
android:layout_width	设置布局宽度
android:autoLink	设置是否当文本为URL链接时，文本显示为可点击的链接
android:autoText	如果设置，将自动执行输入值的拼写纠正
android:bufferType	指定getText()方式取得的文本类别
android:capitalize	设置英文字母大写类型。需要弹出输入法才能看得到
android:cursorVisible	设定光标为显示/隐藏，默认显示

TextView文本框



EditText编辑框

- EditText可以接受用户的输入
- 在EditText中，常用的inputType属性值

Button按钮

方法	功能描述
onKeyDown()	当用户按键时，该方法被调用
onKeyUp()	当用户按键弹起后，该方法被调用
onKeyLongPress()	当用户保持按键时，该方法被调用
onKeyMultiple()	当用户多次按键时，该方法被调用
invalidateDrawable()	用于刷新Drawable对象
onPreDraw()	用于设置视图显示
setOnKeyListener()	用于设置按键监听器
setOnClickListener()	用于设置点击监听器
onKeyDown()	当用户按键时，该方法被调用

单选按钮和单选按钮组

- RadioButton表示单个圆形单选框，RadioGroup是一个可以容纳多个RadioButton的容器
- 同一个RadioGroup中，只能有一个RadioButton被选中
- 不同的RadioGroup中，RadioButton互不影响
- 一个RadioGroup中至少有2个RadioButton

CheckBox复选框

- CheckBox复选按钮具有选中或者未选中两种状态
- 可以通过点击来改变其状态
- 在CheckBox复选框组中，允许同时选中多个
- CheckBox默认都以矩形表示

注意

Toast是Android中用来显示提示信息的一种机制，与Dialog不同的是：Toast提示没有焦点、且时间有限，在一定的时间后会自动消失。

开关控件

- ToggleButton

XML属性	对应方法	功能描述
android:checked	setChecked(boolean)	设置该按钮是否被选中
android:textOff	setTextOff(CharSequence)	设置按钮的状态关闭时所显示的文本
android:textOn	setTextOn(CharSequence)	设置按钮的状态打开时所显示的文本

开关控件

- Switch

XML属性	对应方法	功能描述
android:checked	setChecked(boolean)	设置该按钮是否被选中
android:textOff	setTextOff(CharSequence)	设置按钮的状态关闭时所显示的文本
android:textOn	setTextOn(CharSequence)	设置按钮的状态打开时所显示的文本
android:switchMinWidth	setSwitchMinWidth(int)	设置开关的最小宽度
android:textStyle	setSwitchTypeface (Typeface , int)	设置开关的文本风格
android:typeface	setSwitchTypeface(Typeface)	设置开关的文本的字体风格
android:switchPadding	setSwitchPadding(int)	设置开关与标题文本之间的空白
android:thumb	setThumbResource(int)	使用自定义的Drawable来绘制开关的开关按钮
android:track	setTrackResource(int)	使用自定义的Drawable来绘制开关的开关轨道

图片视图(ImageView)

- 继承自View组件
- 用于显示图像资源

XML属性	对应方法	功能描述
android:adjustViewBounds	setAdjustViewBounds(boolean)	是否保持宽高比
android:cropToPadding	setCropToPadding(boolean)	截取指定区域是否使用空白代替
android:maxHeight	setMaxHeight(int)	设置View的最大高度
android:maxWidth	setMaxWidth(int)	设置View的最大宽度
android:src	setImageResource(int)	设置ImageView所显示的Drawable对象
android:scaleType	setScaleType(ImageView.ScaleType)	设置所显示的图片如何缩放或移动以适应ImageView的大小

Dialog对话框

Android常用的4种对话框：

- AlertDialog提示对话框
- ProgressDialog进度条对话框
- DatePickerDialog日期对话框
- TimePickerDialog时间对话框

AlertDialog提示对话框

- **AlertDialog继承自Dialog类**
- **使用Builder内部类进行创建**
- **可以包含一个标题、一个内容消息或者一个选择列表以及0至3个按钮。**

方法	功能描述
void create()	根据设置的属性，创建一个AlterDialog
void show()	根据设置的属性，显示已创建的AlterDialog
AlterDialog.Builder setTitle()	设置标题
AlterDialog.Builder setIcon()	设置标题的图标
AlterDialog.Builder setMessage()	设置标题的内容
AlterDialog.Builder setCancelable()	设置是否模态
AlterDialog setPositiveButton()	为对话框添加Yes按钮
AlterDialog setNegativeButton	为对话框添加No按钮

AlterDialog.Builder的大部分设置属性的方法返回是此AlterDialog.Builder对象，所以可以使用链式方式编写代码，这样更方便。

ProgressDialog进度对话框

- ProgressDialog有两种显示方式：
  - 滚动的环状图标
  - 带刻度的进度条
- 通过ProgressDialog.setProgressStyle()方法进行设置：
  - STYLE_HORIZONTAL——刻度滚动
  - STYLE_SPINNER——图标滚动，默认选项

本章总结

- Android应用的绝大部分UI组件都放在android.widget包及其子包中，Android应用程序的所有UI组件都继承了View类
- Android中的界面元素主要由以下几个部分构成：视图、视图容器、Fragment、Activity和布局管理器
- Android的所有UI组件都是建立在View、ViewGroup基础之上的，Android采用了“组合器”模式来设计View和ViewGroup，其中ViewGroup是View的子类
- 布局管理器可以根据运行平台来调整组件的大小，程序员的工作只是为容器选择合适的布局管理器即可

本章总结

- Android提供了多种布局，常用的布局有以下几种：LinearLayout(线性布局)、RelativeLayout(相对布局)、TableLayout(表格布局)和AbsoluteLayout (绝对布局)
- Android提供了两种方式的事件处理：基于回调的事件处理和基于监听的事件处理
- Android系统中引用Java的事件处理机制，包括事件、事件源和事件监听器三个事件模型
- Android的事件处理机制是一种委派式事件处理方式，该处理方式类似于人类社会的分工协作。这种委派式的处理方式将事件源和事件监听器分离，从而提供更好的程序模型，有利于提高程序的可维护性和代码的健壮性
- 对于基于回调的事件处理模型而言，事件源和事件监听器是统一的，当用户在GUI组件上触发某个事件时，组件自身的方法将会负责处理该事件
- 对Widget组件进行UI设计时既可以采用XML布局方式也可以采用编码方式来实现，其中XML布局方式更加简单易用，被广泛使用