## 开发环境

1. 虚拟机name首字母大写
2. Res文件夹放的是应用所用的全部资源，AAPT会为项目自动生成R.java（资源清单类）文件，R类为每份资源定义一个内部类，每个资源项对应内部类的Filed
3. 在Java代码中用R.string.app\_name应用字符串变量

在XML中用@<资源对应的内部类名>/<资源项名称> 如：@string/app\_name

在XML中使用标识符，不需要专门定义，直接分配即可@+id/<标识符代号>

如：android：id=”@+id/ok”

在XML中调用该组件 @id/<标识符代号>

## UI

1. 布局管理器
2. 线性布局 LinearLayout

main.xml，string.xml，Android\_01.java，AndroidManifest.xml

1. 表格布局 TableLayout

列编号从0开始

1. 帧布局 FrameLayout

先定义的位于底层

1. 相对布局 RelativeLayout
2. 网格布局 GirdLayout
3. 绝对布局 AbsoluteLayout
4. UI组件
5. 文本框 TextView 不允许编辑文本内容
6. 编辑框 EditText 允许编辑文本
7. 内部类最后大括号后加分号
8. 按钮
9. 按钮 Button
10. 单选按钮 RadioButton
11. 复选框 CheckBox
12. 状态开关按钮 ToggleButton
13. 开关 Switch

new OnCheckedChangeListener()

1. 时钟
2. 模拟时钟 AnalogClock
3. 数字时钟 DigitalClock

new OnClickListener()

各个组件都需要设置宽和高

1. 单位
2. dp也就是dip，dp与sp基本类似，dp是与密度无关，sp除了与密度无关外，还与scale无关，不依赖像素，使用dp和sp，系统会根据屏幕密度的变化自动进行转换。
3. px：表示屏幕实际的像素。例如，320\*480的屏幕在横向有320个像素，在纵向有480个像素。
4. in：表示英寸，是屏幕的物理尺寸。每英寸等于2.54厘米。例如，形容手机屏幕大 小，经常说，3.2（英）寸、3.5（英）寸、4（英）寸就是指这个单位。这些尺寸是屏幕的对角线长度。如果手机的屏幕是3.2英寸，表示手机的屏幕（可 视区域）对角线长度是3.2\*2.54 = 8.128厘米。
5. mm：表示毫米，是屏幕的物理尺寸。
6. pt：表示一个点，是屏幕的物理尺寸。大小为1英寸的1/72。
7. imageView
8. imagebutton 图片按钮
9. ZoomButton 放大/缩小按钮



new OnTouchListener()

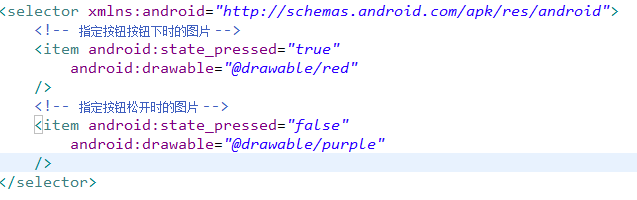
1. ZoomControls放大缩小按钮



1. android:layout\_gravity与android:gravity的区别

android:gravity：是对view控件本身来说的，是用来设置view本身的文本应该显示在view的什么位置,默认值是左侧

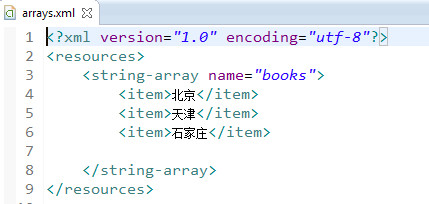
android:layout\_gravity：是相对于包含该元素的父元素来说的，设置该元素在父元素的什么位置



1. QuickContactBadge继承了ImageView，点击图片关联联系人
2. AdapterVIew 列表视图（ListView）和ListActivity
3. 基于数组的ListView



定义数组



1. 用ArrayAdapter创建ListView

android:padding="10px" 在组件的四边设置填充区域

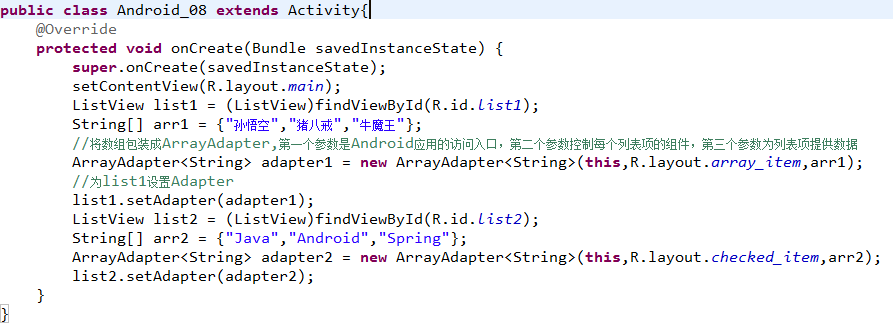
android:shadowRadius="2" 设置阴影的模糊程度

1. 定义两个ListView

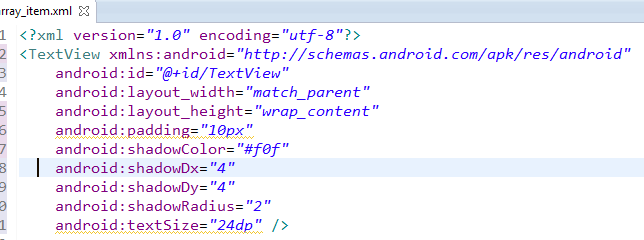


1. 为ListView指定ArrayAdapter

Bundle类用作携带数据，它类似于Map，用于存放key-value名值对形式的值，相对于Map，它提供了各种常用类型的putXxx()和getXxx()方法。

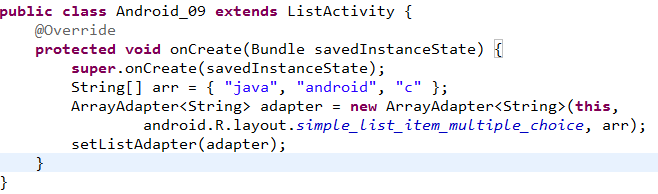


1. 定义控制列表项的组件



每显示一个列表项，都会调用一次定义的\*\_item.xml布局文件，也就是“孙悟空”会调用一次，“猪八戒”也会调用一次……

1. 基于ListActivity实现列表，不需要指定布局文件，相当于布局文件中只有一个ListView



1. 使用SimpleAdapter创建ListView

List对象的元素是Map

SetOnItemClickListener()

setOnItemSelectListener()

1. 扩展BaseAdapter实现不同存储列表项的ListView

获得Adapter最大的控制权

1. AutoCompleteTextView自动完成文本框

当用户输入一定字符之后，自动完成文本框会显示一个下拉菜单，供用户从中选择，当用户选择某个菜单项之后，AutoCompleteTextView按用户选择自动填写该文本框

1. GridView 网格视图

GridView与ListView的区别在于，ListView只显示一列，而GridView显示多列

1. ExpandableListView可展开的列表组件
2. Spinner 列表选择框，相当于弹出一个菜单供用户选择
3. 用android:entries指定Spinner组件
4. 用Adapter提供Spanner组件
5. Gallery画廊视图

水平列表选择框，与Spinner的区别是：Spinner作用是提供用户选择，而Gallery用户通过拖动查看上一个、下一个列表项

1. AdapterViewFlipper

每次只显示一个View组件，可以通过showPrevious()和showNext()控制AdapterViewFlipper显示上一个和下一个组件

1. StackView

以堆叠的形式显示多个列表项

可以用两种方式控制，一是拖动的形式，而是点击按钮的形式

1. ProgressBar 进度条
2. SeekBar 拖动条

通常用于对系统的某种数值进行调节，比如音量调节

1. RatingBar星级评分条
2. ViewSwitcher 视图切换组件

每次只显示一个组件，当程序控制从一个人view切换到另一个view，ViewSwitcher支持指定动画效果

1. ImageSwitcher 图像切换器

比ViewSwitcher效果更炫，可以指定图片切换时的动画效果

1. TextSwitcher 文本切换器

与TextView的区别在于TextSwitcher效果更炫，可以指定文本切换时的动画效果

1. ViewFlipper

调用addView添加多个组件，使用动画控制多个组件之间的切换效果

ViewFlipper需要开发者通过addView添加多个view，AdapterViewFlipper用传入的Adapter提供多个View

1. Toast 显示提示信息框

Toast提示信息不会获得焦点

Toast提示信息过一段时间自动消失

1. CalendarView 日历视图

用于显示和选择日期

1. DatePicker 日期选择器
2. TimePicker 时间选择器
3. NumberPicker 数值选择器
4. TabHost 选项卡

在一个窗口放置多个标签页，如：一个窗口显示包括“未接来电”，“已接来电”

1. ScrollView 垂直滚动条

里面最多只能包含一个组件

1. HorizontalScrollView 水平滚动条

里面最多只能包含一个组件

1. Notification 显示在手机状态栏的通知

Notification所代表的是一种具有全局效果的通知，程序一般通过NotificationManager服务来发送Notification。

1. AlertDialog对话框

可以生成各种内容的对话框

生成的对话框分为4个区域：标题区、图标区、内容区、按钮区

AlertDialog提供6种不同内容的对话框：简单文本内容、简单列表项、单选列表项、多选列表项、自定义列表项、自定义View

1. 把普通窗口设置成“对话框风格”也可以实现对话框效果
2. PopupWindow

可以创建类似对话框风格的窗口

1. DatePickerDialog 日期选择对话框

是对DatePicker的包装

1. TimePickerDialog 时间选择对话框

是对TimePicker的包装

1. ProgressDialog进度对话框
2. OptionMenu选项菜单
3. SubMenu子菜单

使用监听器来监听菜单事件

1. 复选菜单项和单选菜单项
2. ContextMenu 上下文菜单
3. 使用XML文件定义菜单
4. 使用PopupMenu创建弹出式菜单
5. ActionBar 活动条

显示选项菜单

程序图标导航Tab导航

下拉式导航

## Android的事件处理

1. 基于监听的事件处理

* 监听的处理模型：

Event Source(事件源) 事件发生的场所，通常是各个组件

Event(事件) 事件封装了界面组件上发生的特定事情，通过Event类获取组件上发生的事件

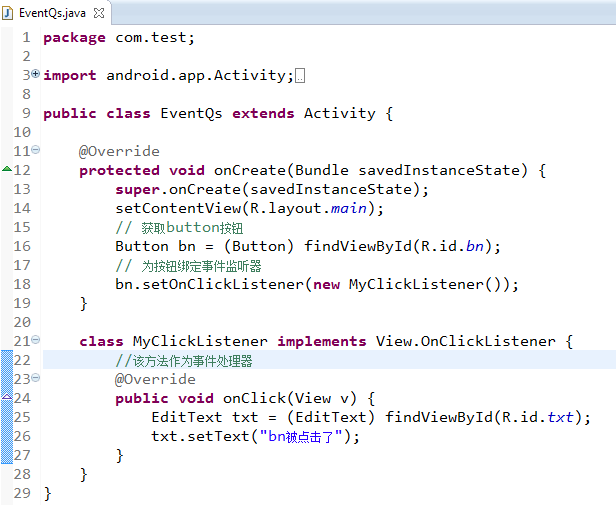
EventListener(事件监听器) 监听事件源发生的事件，并对各种事情做出相应的响应

* 事件监听器以类的形式存在，其中包含的方法就是事件处理器
* 这种事件处理机制是一种委托式事件处理方式
* 每个组件均可以针对特定的事件指定一个事件监听器，每个事件监听器也可以监听一个或多个事件源
* 基于监听的事件处理模型的编程步骤：
* 获取普通界面组件（事件源），也就是被监听的对象
* 实现事件监听器类，必须实现一个XxxListener接口
* 调用事件源的setXxxListener方法注册事件监听器

界面布局文件main.xml



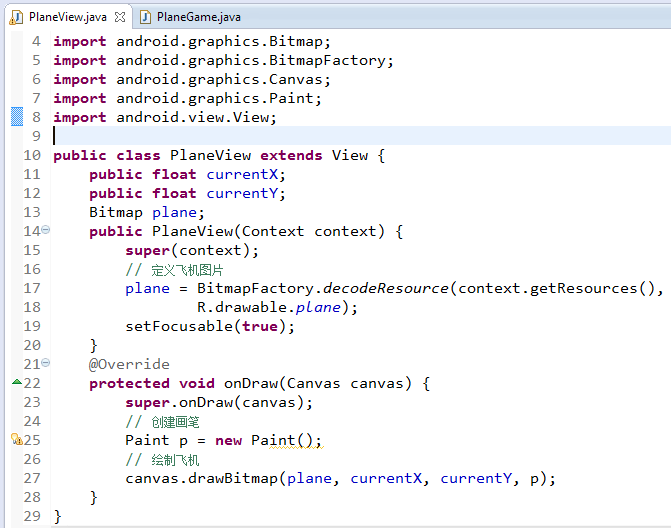
实现类



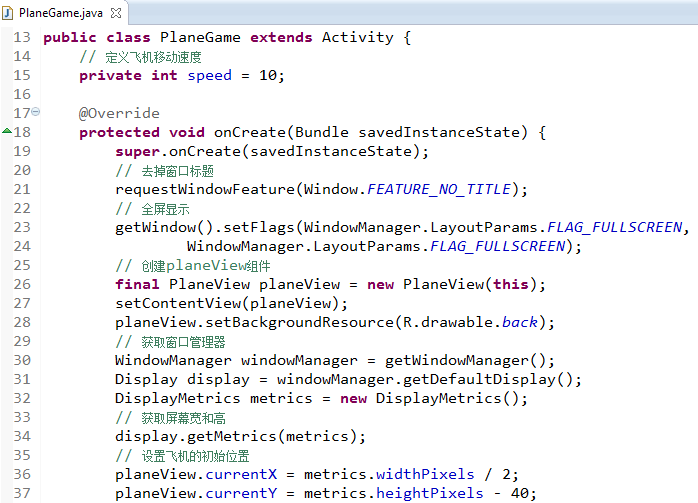
* 事件和事件监听器

如果事件源触发的事件足够简单、事件里封装的信息比较有限，那就无须封装事件对象、无须将事件对象传入事件监听器；对于包含更多信息的事件，Android会把事件信息封装成XxxEvent对象，并把该对象作为参数传入事件处理器

绘制飞机的View



Activity代码





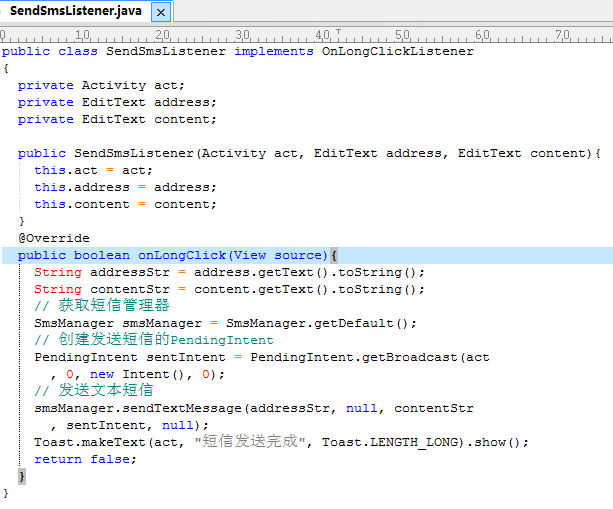
* 在程序中实现事件监听器的几种形式：
* 内部类形式：将事件监听器类定义成当前类的内部类

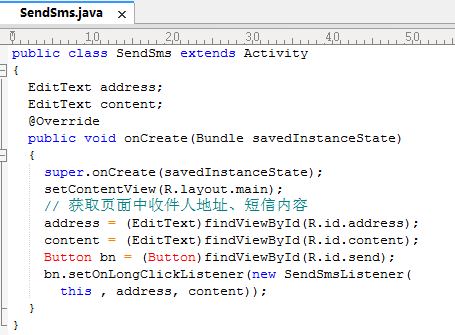
使用内部类可以在当前类中复用该监听器类；

可以自有访问外部类的所有界面组件；

* 外部类形式：将事件监听器定义成一个外部类

如果某个事件监听器确实需要被多个GUI（图形用户界面Graphical User Interface）界面所共享，而且主要是完成某种业务逻辑的实现，则可以考虑使用外部类的形式来定义事件监听器类





* Activity本身作为事件监听器类：让Activity实现监听器接口，并实现事件处理方法

缺点：

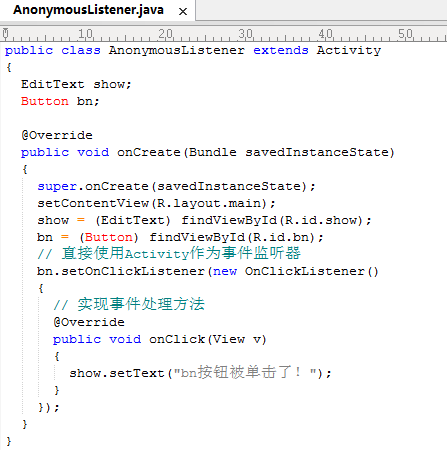
造成程序结构混乱，Activity的主要职责是完成界面初始化工作，但此时还需要包含事件处理方法，引起混乱

Activity实现监听器接口让人感觉怪异



* 匿名内部类形式：使用匿名内部类创建事件监听器对象

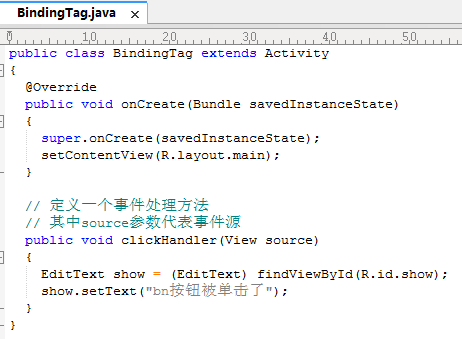
“new 监听器接口”或“new 事件适配器”的形式就是用于创建匿名内部类形式的事件监听器



* 标签绑定监听器

直接在界面布局文件中指定标签绑定事件处理方法





1. 基于回调的事件处理

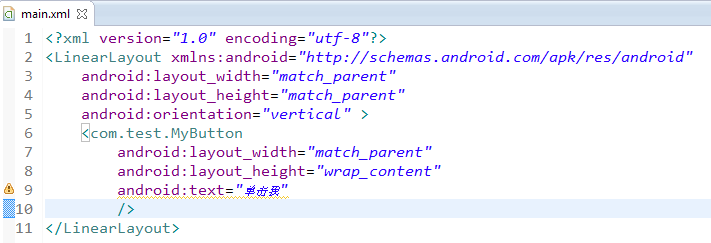
* 原理

对于基于回调的事件处理模型来说，事件源与事件监听器是统一的，或者说事件监听器完全消失了。当用户在GUI组件上激发某个事件时，组件自己特定的方法将会负责处理该事件。

* 实现方式

继承GUI组件类，并重写该类的事件处理方法。基于回调的事件处理机制可通过自定义View来实现，自定义View时重写该View的事件处理方法即可





* 返回值的意义

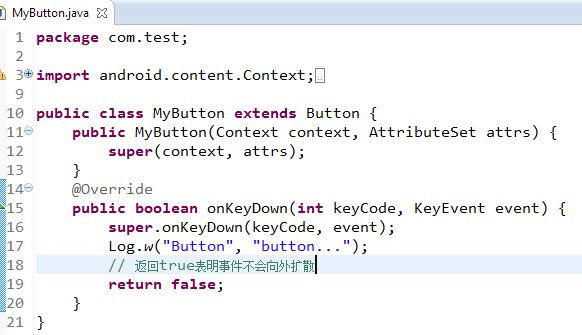
为true时，表明该处理方法已完全处理该事件，事件不会传播出去

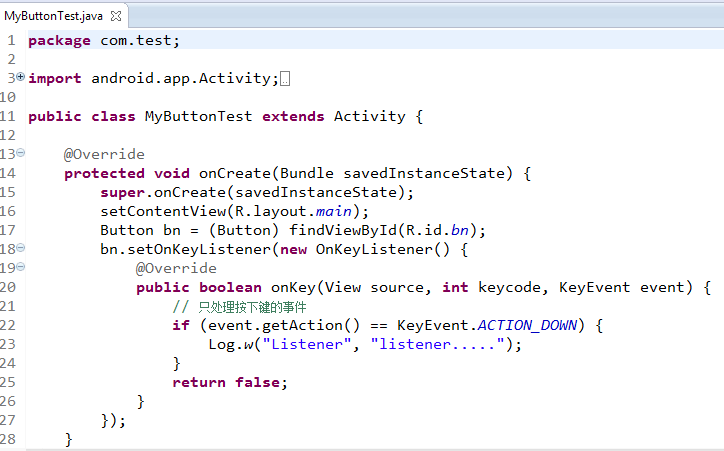
为false时，表明该处理方法未完全处理该事件，事件会向外传播

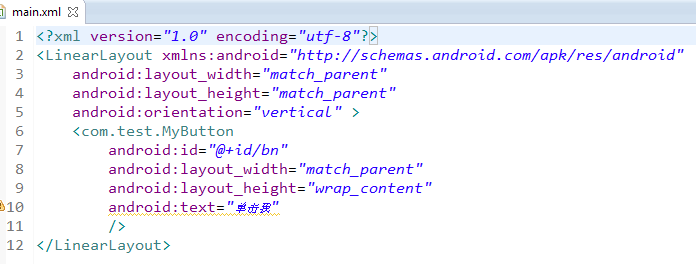
组件上发生的事件不仅激发该组件上的回调方法，也会触发该组件所在Activity的回调方法，只要事件能传播到Activity

* 事件传播

Android系统最先触发的是该按键上绑定的事件监听器，接着触发该组件提供的事件回调方法，然后才会传播到该组件所在的Activity

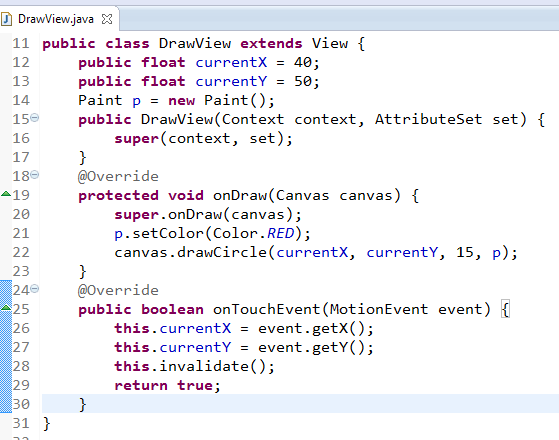


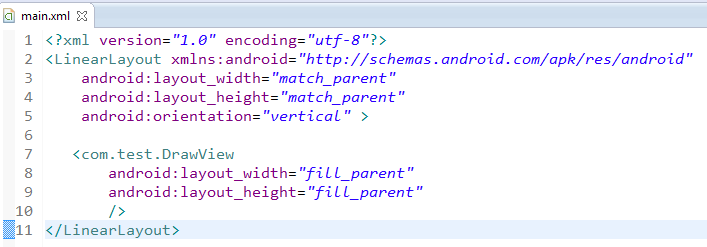




1. 两种事件处理方式对比

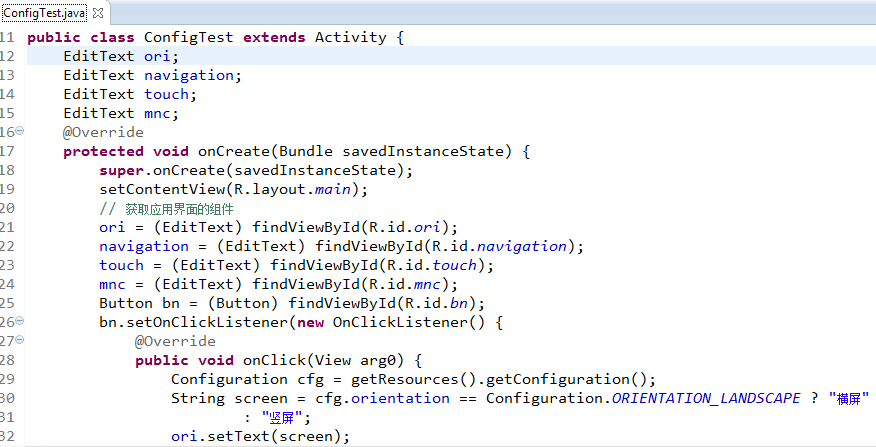
* 基于监听的事件处理模型具有更大的优势，基于监听的事件模型分工更明确，事件源、事件监听由两个类分开实现，更易于维护，Android处理机制保证基于监听的事件监听器被优先触发；基于回调的事件处理更合适应对事件处理逻辑比较固定的View

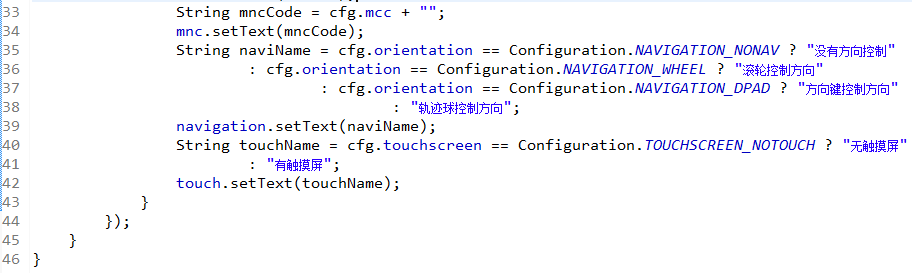




1. Configuration类简介

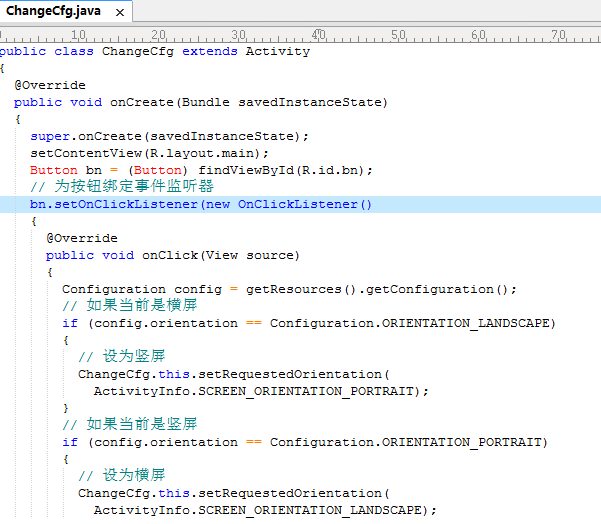
* Configuration类专门用于描述手机设备上的配置信息，这些配置信息既包括用户特定的配置项也包括系统的动态设备配置



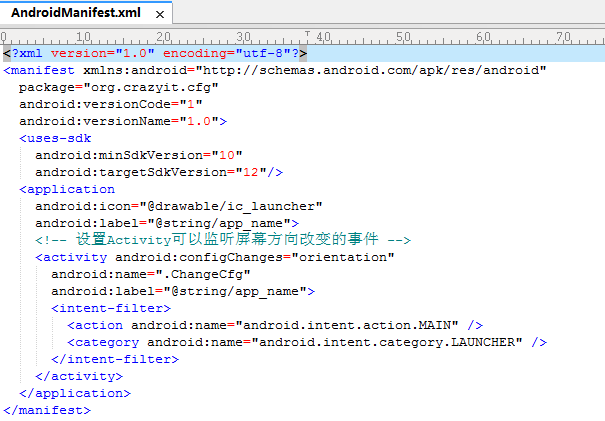


* 重写onConfigurationChanged响应系统设置更改

如果程序需要监听系统设置的更改，重写Activity的onConfigurationChanged()方法，这事一个基于回调的事件处理方法，当系统发生更改时，该方法会被自动触发







1. Handler消息传递机制

* Android的UI操作是线程不安全的，只允许UI线程修改Activity的UI组件
* 当一个程序第一次启动时，Android会同时启动一条主线程，又叫做UI线程
* Handler类的主要作用有两个：在新启动的线程中发送消息，在主线程中获取、处理消息
* 重写Handler类中处理消息的方法，当新线程发送消息时，消息会发送到与之关联的MessageQueue，而Handler会不断从MessgeQueue中获取并处理消息，导致Handler类中处理消息的方法被回调。



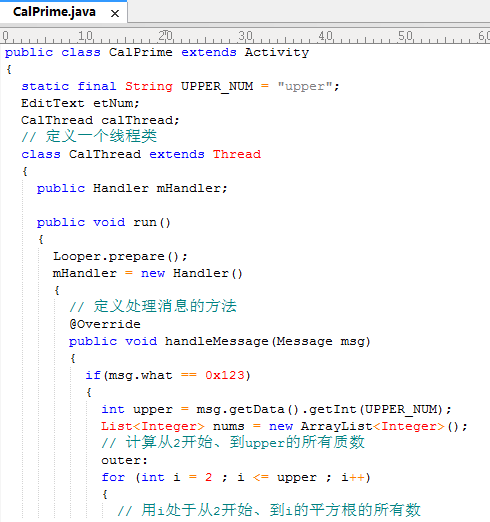


* 每个线程只能拥有一个Looper，每个Looper管理一个MessageQueue

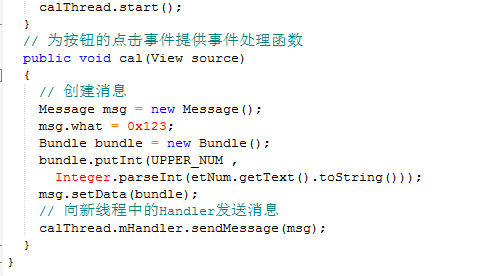
主UI线程中，系统已经初始化了一个Looper对象，直接创建Handler即可，然后就可通过Handler来发送消息、处理消息

程序员自己启动的子线程，需要自己创建一个Looper对象，并启动它，创Looper对象调用它的prepare()方法即可。

* 在线程中使用Handler的步骤：
* 调用Looper的prepared()方法为当前线程创建Looper对象，自动构建MessageQueue
* 创建Handler子类的实例，重写handleMessage()方法，该方法负责处理来自其他线程的消息
* 调用Looper的loop()方法启动Looper。



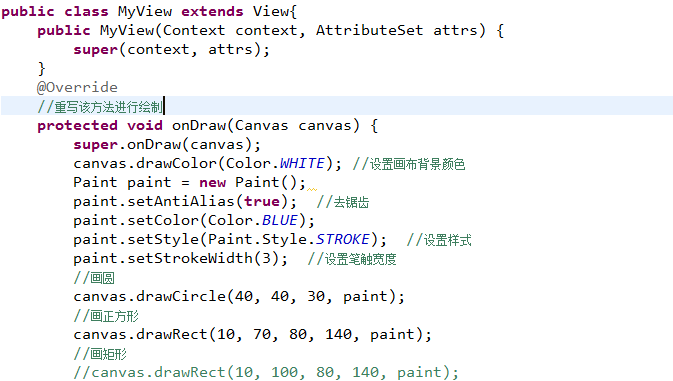


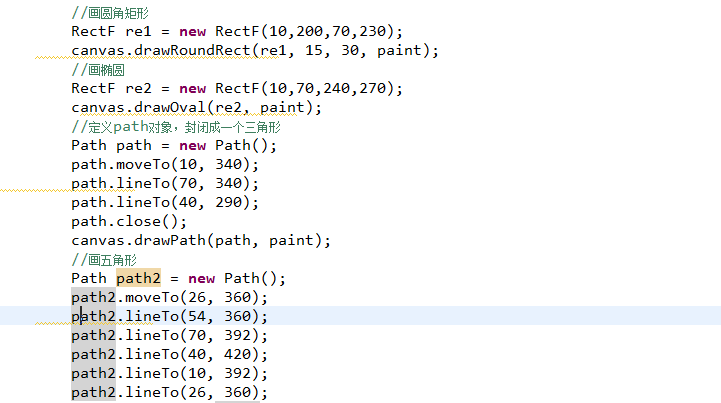


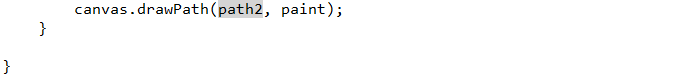
* 尽量避免在UI线程中执行耗时操作，否则会导致ANR异常（Android应用程序无法响应输入事件和Broadcast）

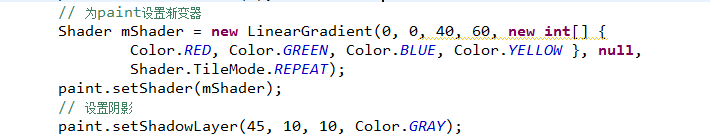
## 图形图像处理

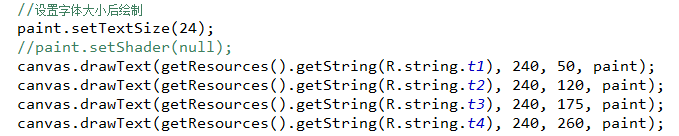
1. Canvas画布，Canvas代表了依附于指定View的画布，Paint代表了Canvas上的画笔



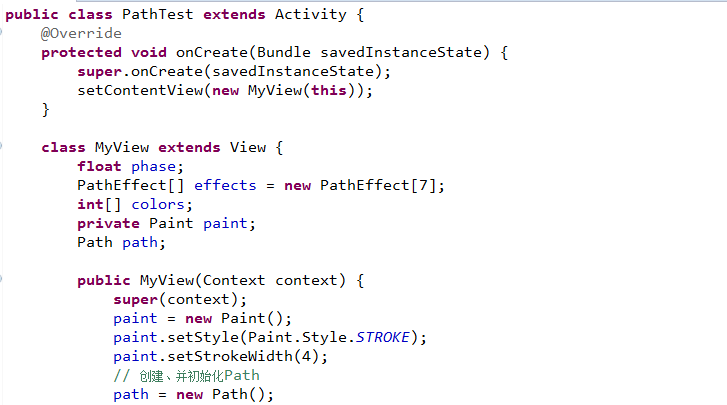


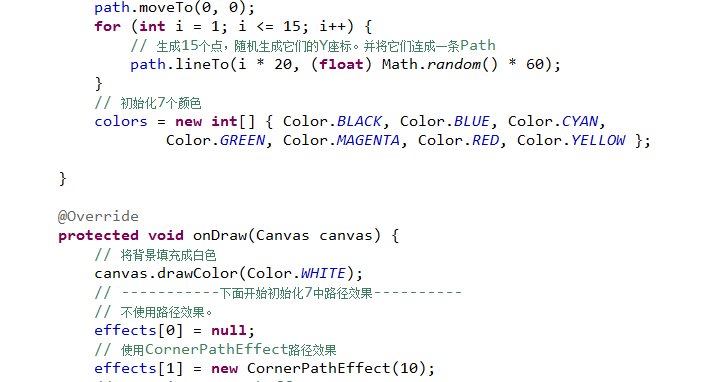


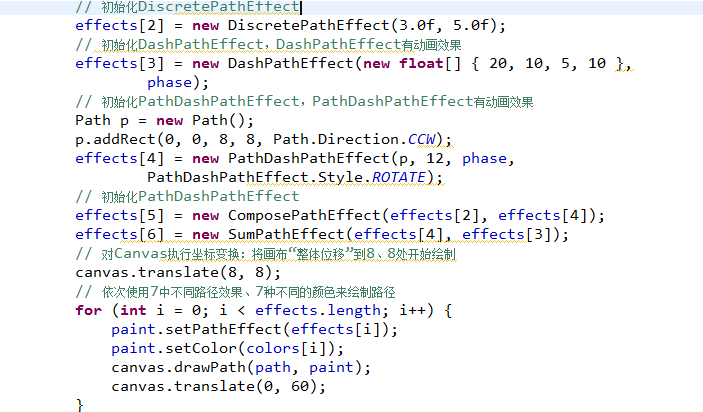




1. Path ，代表任意多条直线连接而成的任意图形







invalidate()作用

