2019 级本科

移动应用开发

张昕

zhangxin@cust.edu.cn

2022

课程总体要求

课程特点

- 依托已有的程序设计及软件开发技术基础
- 面向实际开发场景与需求
- 强调处理和解决问题的能力
- 内容线索跨度较大

课程信息

- 理论课时： 32 学时　　实验课时： 16 学时
- 课程学分： 2.5
- 成绩评定
  - 平时成绩： 10%
  - 实验成绩： 20%
  - 考试成绩： 70%

学习要求

- 注重实际开发能力的培养
- 要求独立完成实验课程题目
- 配合必要的课下练习

课程总体要求

- **内容安排**
  - 以基于 Android 移动操作系统的技术开发为主
  - 搭配少量 Web App 开发技术
  - 以技术主题为内容主线
- **技术主题**
  - 用户界面设计及 UI 组件
  - 基本程序单元及事件处理
  - 资源访问
  - 消息、通知、广播与闹钟
  - 多媒体处理
  - 数据存储技术
  - Handler 消息处理
  - Service 应用
  - 传感器与定位服务
  - 网络编程与应用
  - 开发实例

Operating Systems in Smart Devices

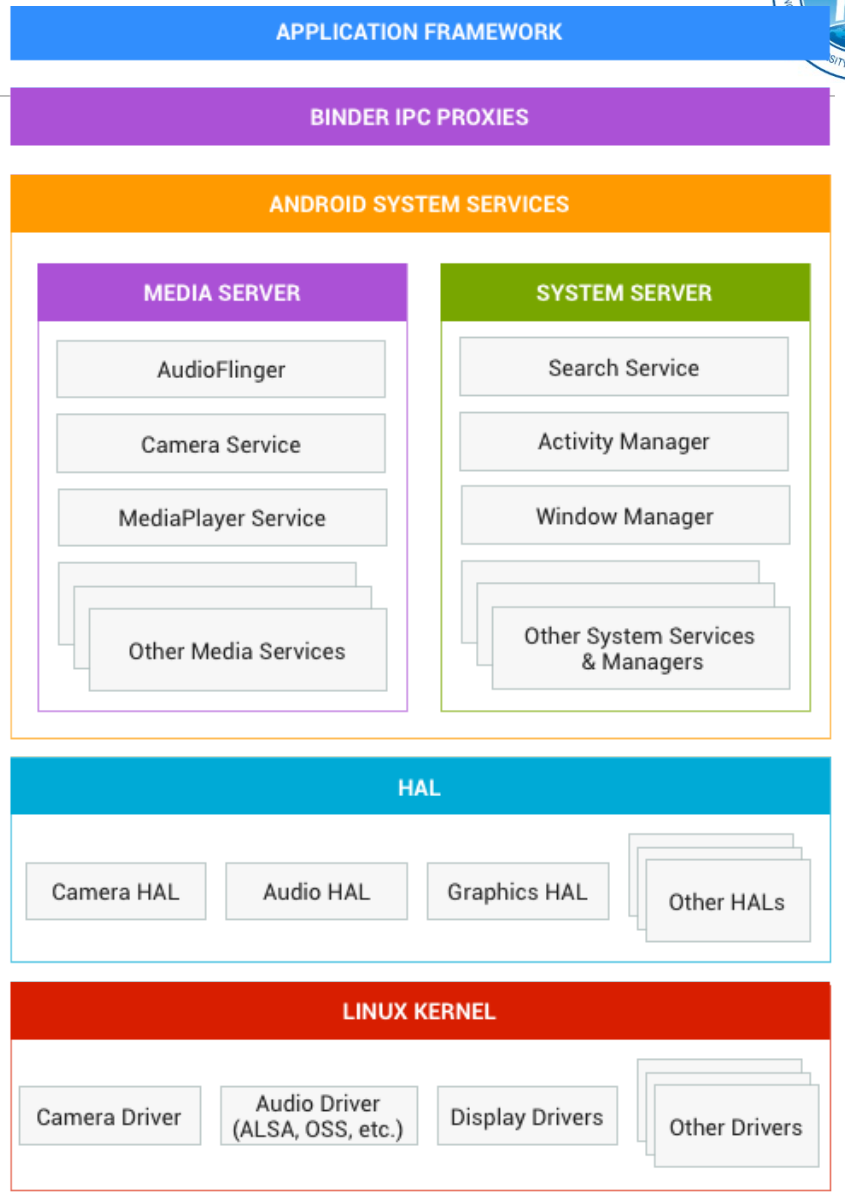
- **Representative Mobile Operating System**
  - Android 、 iOS 、 HarmonyOS
  - Windows Mobile 、 Windows Phone 、 BlackBerry 、 Symbian 、 PalmOS
  - Linux

HarmonyOS

- **Android**
  - Based on Linux kernel (without GNU components)
  - Initially created by Andy Rubin (Android Inc.) and purchased by Google in 2005
  - 1st version (Android 1.0) was published by Google in September 2008
  - Latest version： Android11 （released in Sep 2020）

Android system architecture

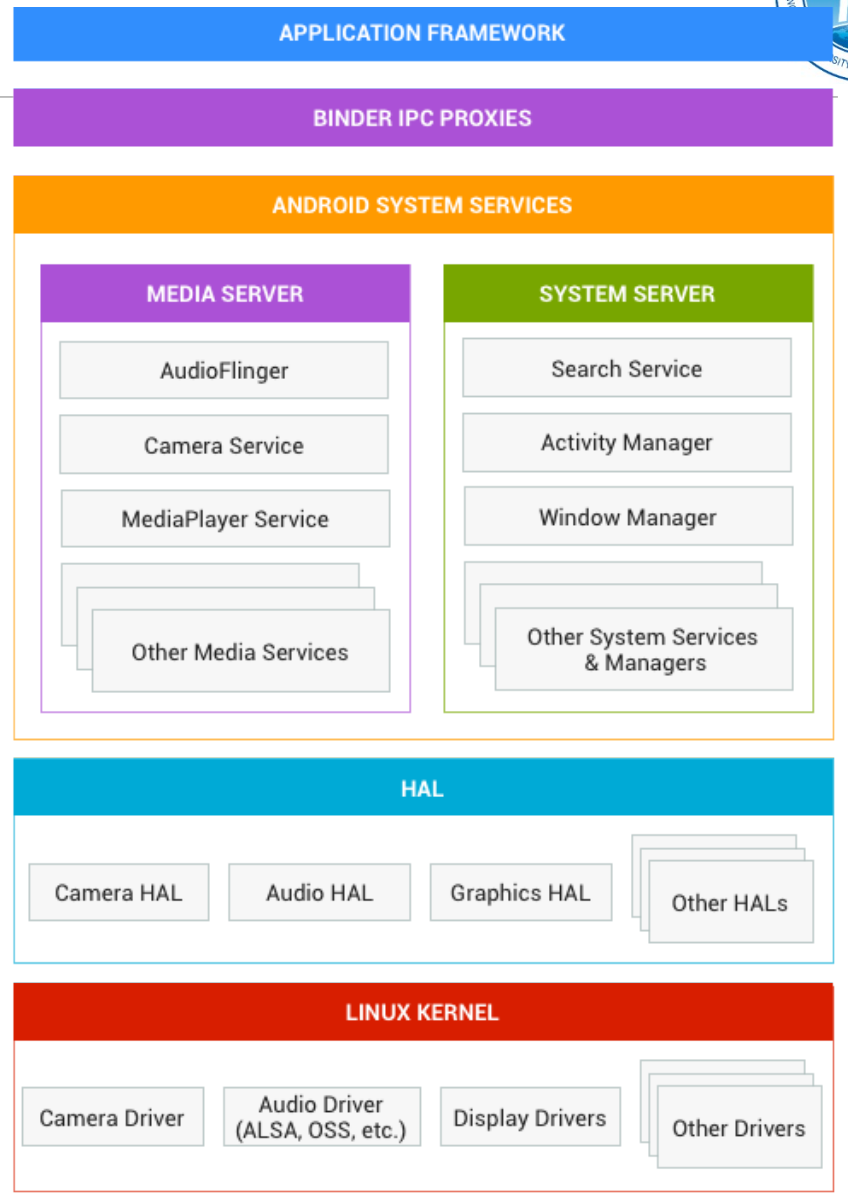
- **Application framework**
  - Often utilized by developers
  - Offers APIs for implementing functions
- **Binder IPC (Inter-Process Communication)**
  - Binder works as a function/delegation for handling process communication to call system services
- **System services**
  - Provides modular components of playing functionalities
    - e.g., window manager, search service
  - Respectively performed by media sever and system server



Source: https://source.android.com/devices/architecture/

Android system architecture

- **HAL (Hardware Abstraction Layer)**
  - Defines a standard interface for hardware venders to implement
  - Using a HAL allows to implement functionality without affecting or modifying the higher level system
  - HAL implementations are packaged into modules and loaded by the Android system at the appropriate time
- **Linux kernel**
  - Android uses a version of the Linux kernel with a few special additions and other features important for a mobile embedded platform
  - These additions are primarily for system functionality and do not affect driver development

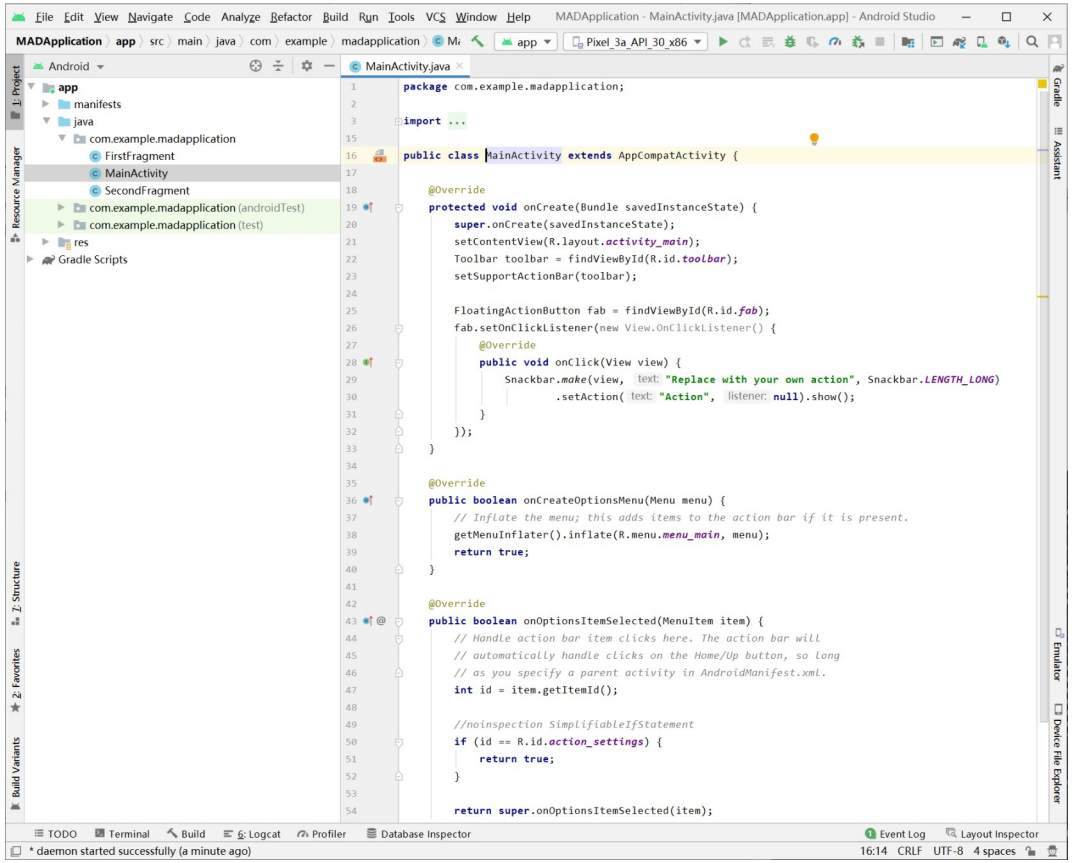


Source: https://source.android.com/devices/architecture/

Android Integrated Development Environment

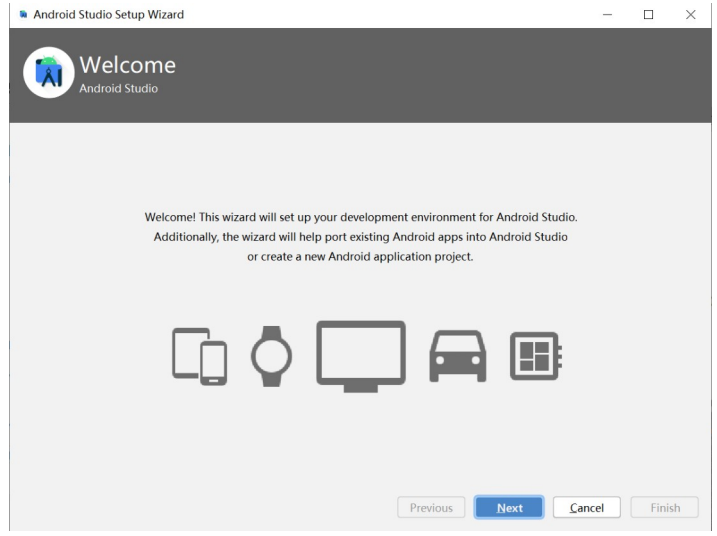
Android studio

Android Studio provides the fastest tools for building apps on every type of Android device



Default main page

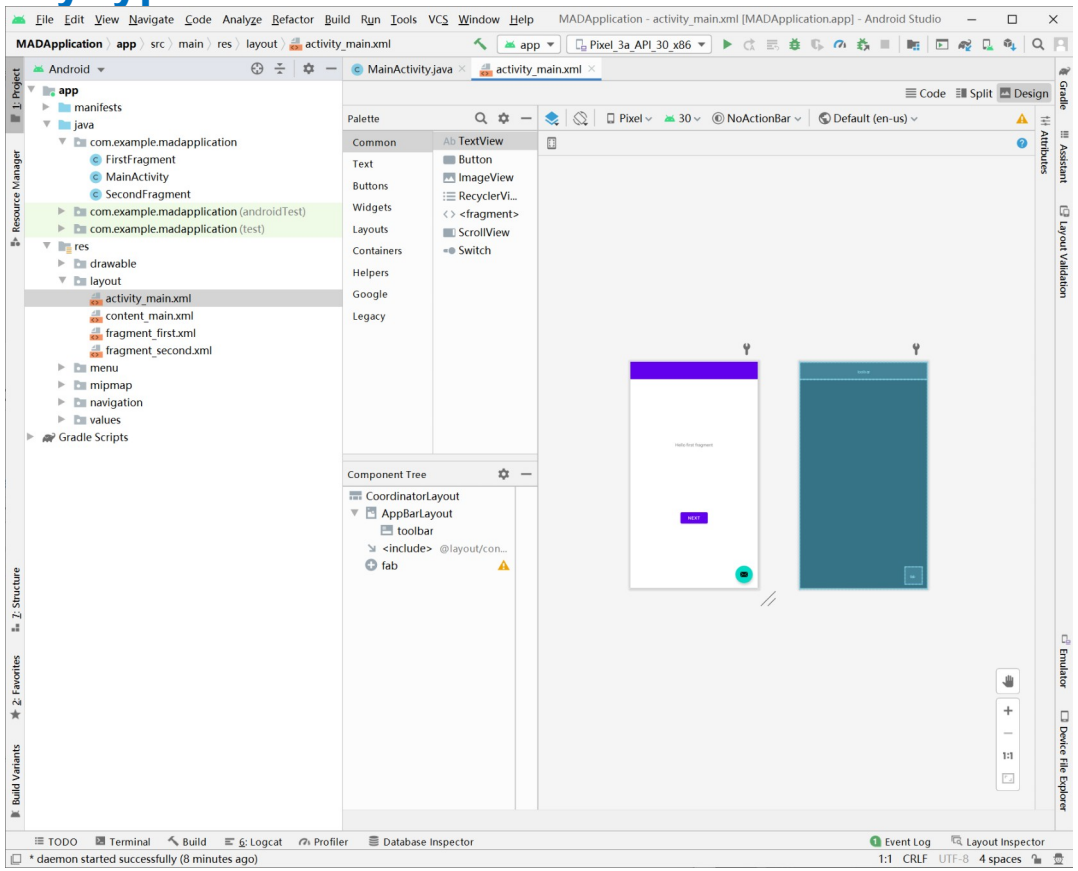
Download Link: https://developer.android.com/studio



Android Integrated Development Environment

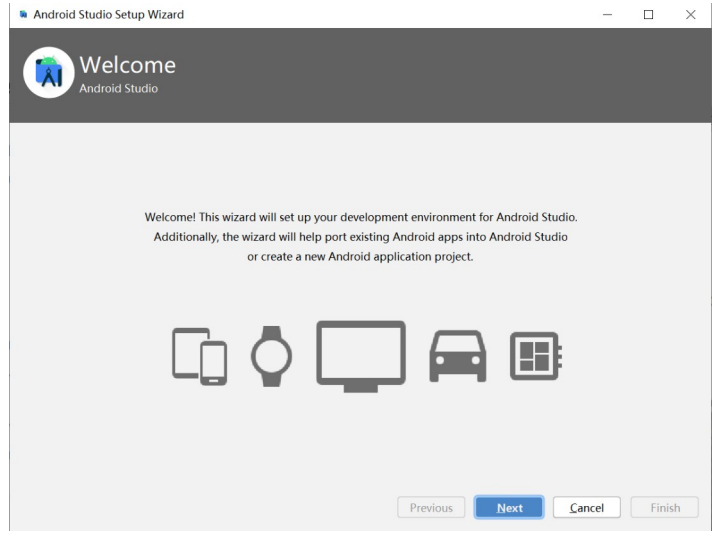
Android studio

Android Studio provides the fastest tools for building apps on every type of Android device



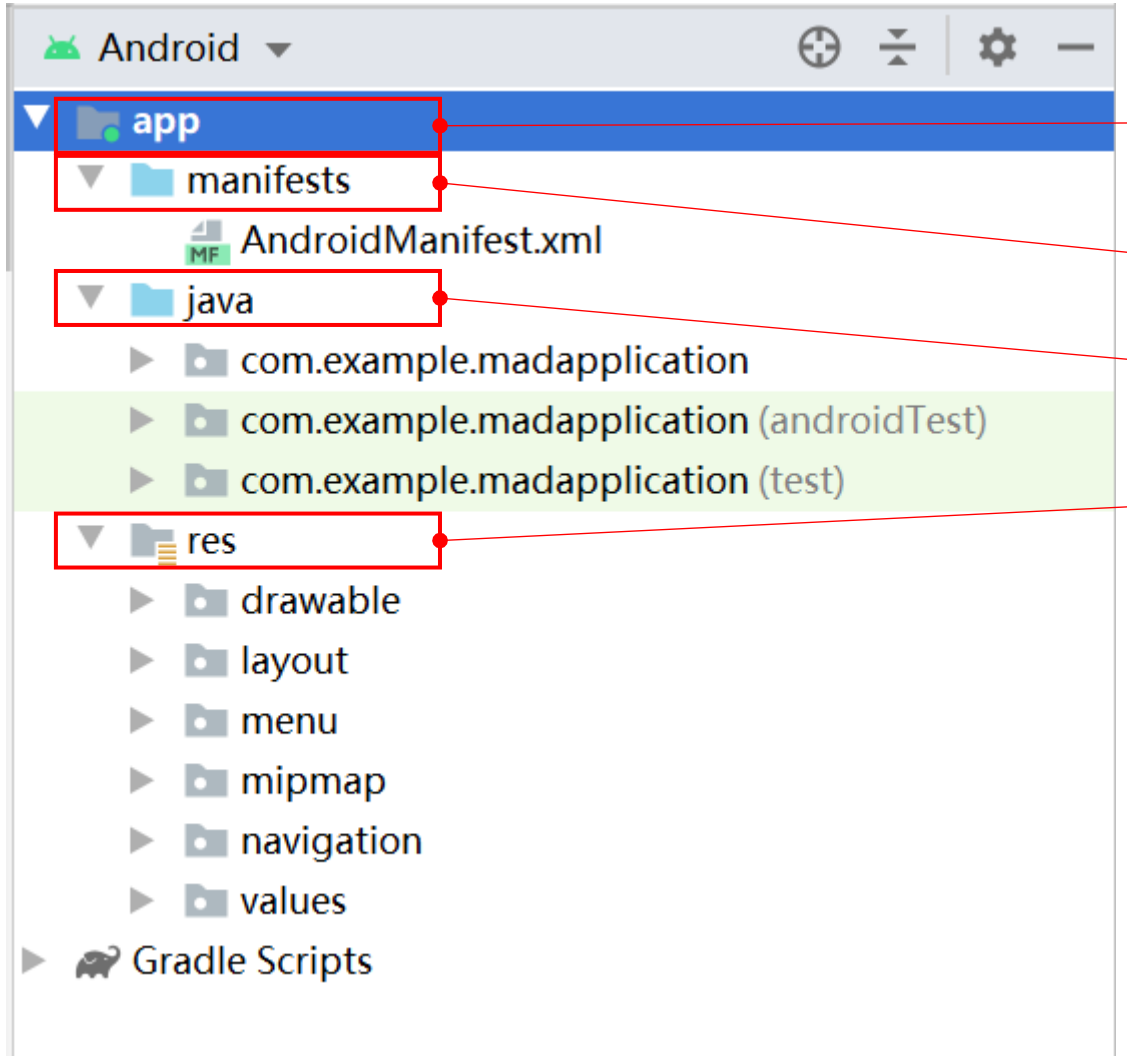
UI editor

Download Link: https://developer.android.com/studio



Android Integrated Development Environment

Android studio



When creating a project, a module named “app” is then established automatically

Folder of configuration files

Folder of java source files

Folder of resources (e.g., pictures, layout files, menu resources, string resources)

Android Integrated Development Environment

- **manifest folder**
  - Contains the configure file to display Android application
  - Every Android application must have one AndroidManifest.xml file under manifest folder
    - AndroidManifest.xml is the global description specifying application name, icon, Acitivity, Service, etc.
    - An application can not run without the corresponding AndroidManifest.xml file

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.madapplication">

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/Theme.MADApplication">
        <activity
            android:name=".MainActivity"
            android:label="@string/app_name"
            android:theme="@style/Theme.MADApplication.NoActionBar">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>

</manifest>
```

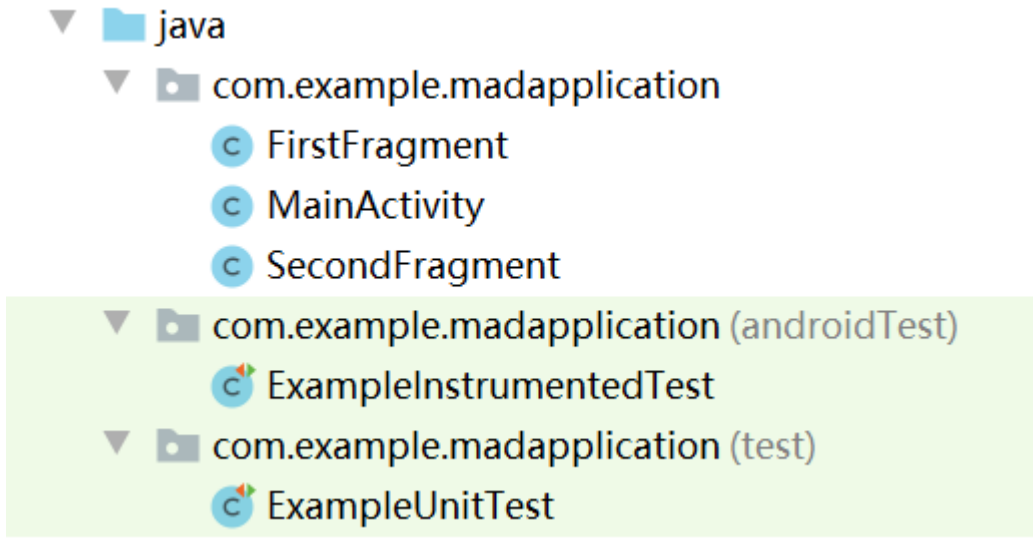
AndroidManifest.xml

Element	Description
manifest	Root element, describing all the content in the package
xmlns:android	Including the declaration of naming space to accept the various standard attributes in the xml file
package	Declaring the application package
application	Including the root element declared by application-level components, one manifest could contain 0 or 1 application element
android:icon	Icon of application
android:label	Program label, i.e., the application name
android:theme	Theme of application, default as @style/AppTheme
activity	Main tool for interacting with users, the initial page when opening app
intent-filter	Configuration of Intent filter
action	Intent Action supported by components
category	Intent Category supported by components, usually used to specify the default invoked activity

- **Every Activity needs to preserve a <activity> element in AndroidManifest.xml**

Android Integrated Development Environment

- **java folder**
  - Contains all the packages and java source files
  - *MainActivity* extends AppCompatActivity
    - The activity is with Action Bar
    - It allows to override *onCreate(…)* method
    - Within *onCreate(…)*, to call *setContentView(R.layout.activity_main)* to configure the layout file in Activity
    - R.layout.activity_main is to obtain layout/activity_main.xml which is layout file

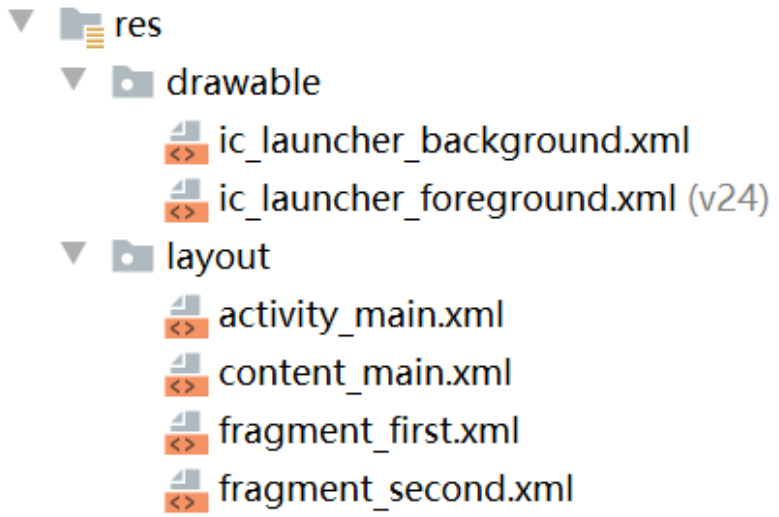


- Every resource would generate one index in R.java file, which allows developers to call the resource files in Android application

R.java is located at <application name>build\generated\source\r\debug\<package path>, and it is a read-only file and automatically maintained when new resources are import

Android Integrated Development Environment

- **res folder**
  - Displays the source files in res folder
  - When resources are newly changed, R.java will be maintained automatically
- **drawable sub-folder**
  - Preserves picture resouces
- **layout sub-folder**
  - Preserves layout files
  - activity_main.xml layout file will be generated by default when creating Android application

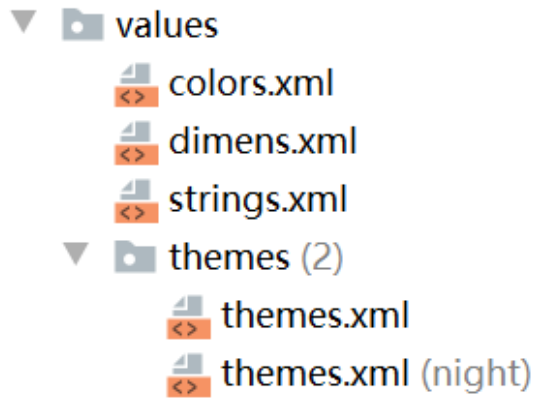


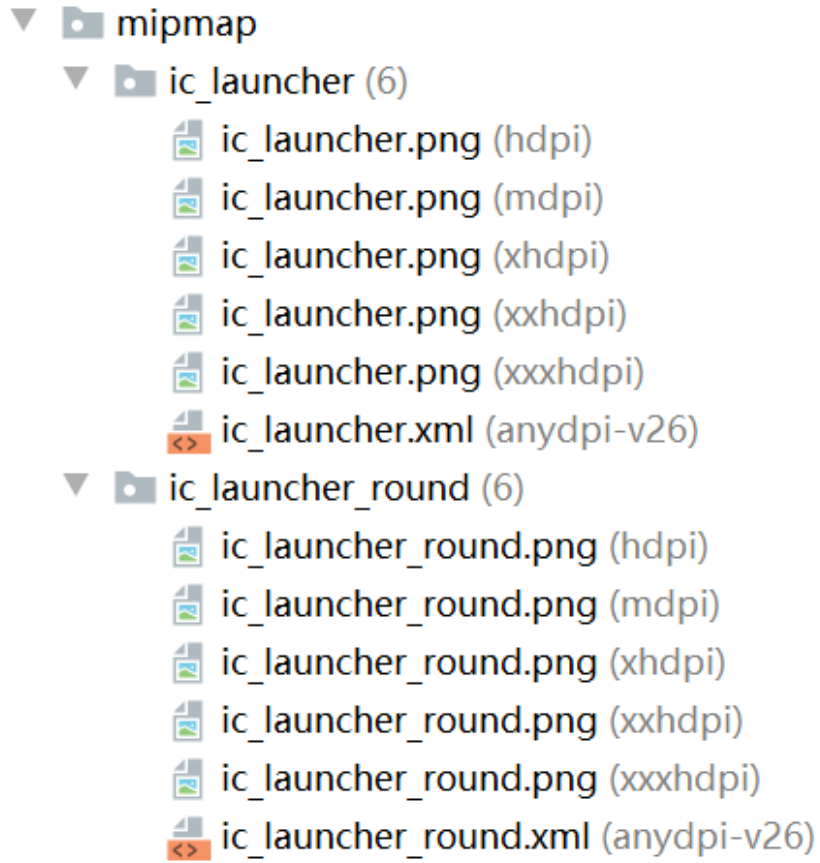
activity_main.xml

Element	Description
CoordinatorLayout	Layout manager
xmlns:android	Including the declaration of naming space to accept the various standard attributes in the xml file
xmlns:tools	Default tool for layout
android:layout_width	Width of the view in screen
android:layout_height	Height of the view in screen
TextView	Text field component to display text
android:text	Text displayed in the text field component

Android Integrated Development Environment

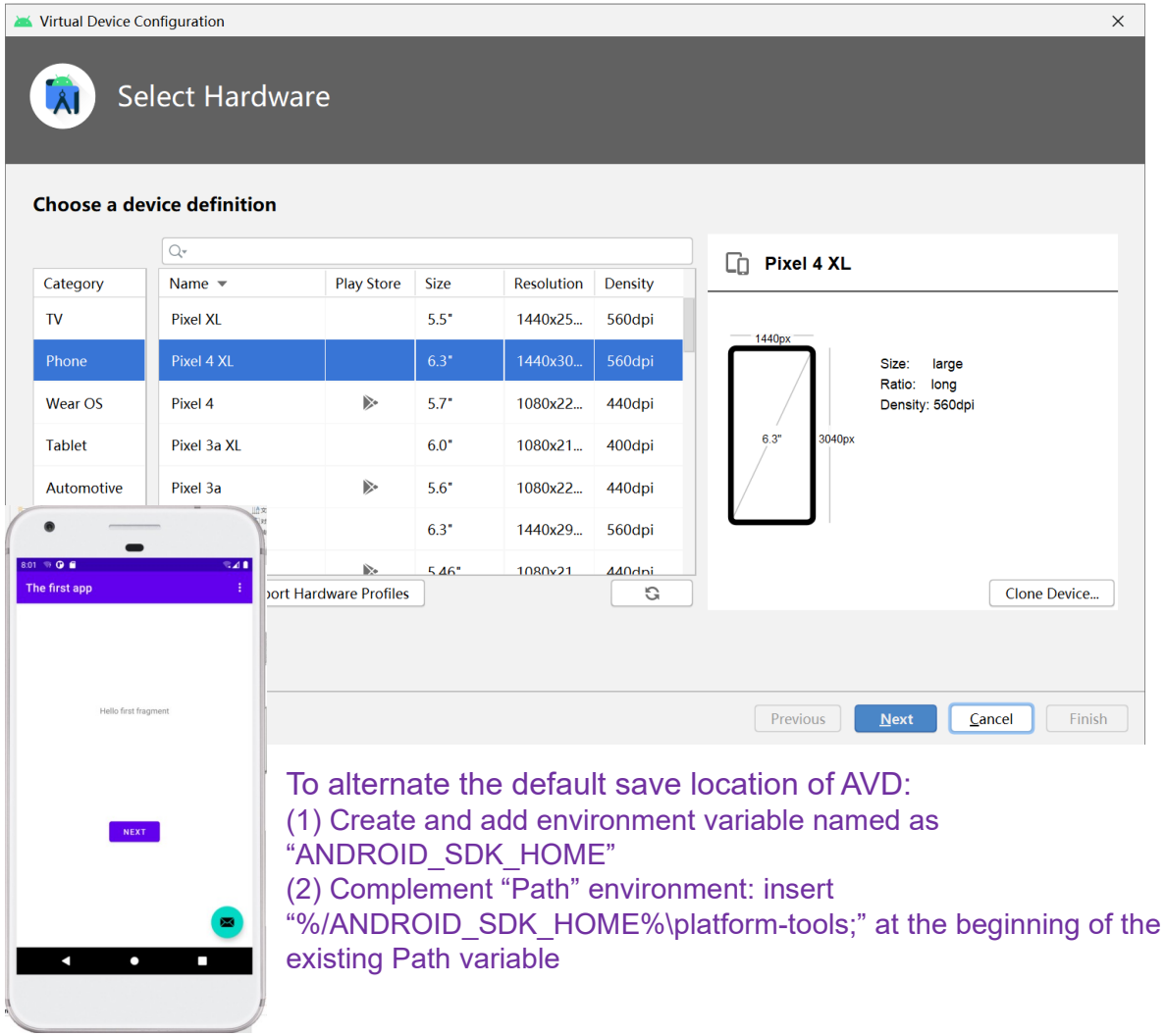
- **mipmap sub-folder**
  - Preserves application icon
  - Should prepare various version for different conditions
- **values sub-folder**
  - Preserves the strings, styles and color resources





Execution of Android apps

- **Two means of executing Android apps**
  - Android Virtual Device (AVD)
    - Create a virtual device in advance
    - Flexible for loading different SDK
    - Agile for alternating frameworks
    - Might not workable on real phone
    - High storage cost
  - Real phone hardware
    - Require to activate developer mode
    - Connect hardware and load apps
    - Hardware-sensitive



To alternate the default save location of AVD:
(1) Create and add environment variable named as "ANDROID_SDK_HOME"
(2) Complement "Path" environment: insert "%/ANDROID_SDK_HOME%\platform-tools;" at the beginning of the existing Path variable

User Interface Design

View

- **A rectangular area on the screen**
  - Responsible for drawng components and event handling
  - Is the base class for *widgets* to create interactive UI components
    - e.g., buttons, text fields, etc.
    - Located in androd.view package
  - All of the views in a window are arranged in a single tree
    - It allows to add views either from code or by specifying a tree of views in one or more XML layout files

View

XML attributes and corresponding methods

Attribute	Methods	Description
android:background	setBackgroundResource(int)	Set background color with Drawable resource or color value
android:clickable	setClickable(Boolean)	Defines whether this view reacts to click events.
android:elevation	setElevation(float)	base z depth of the view. Newly added in Android API 21.
android:id	setId(int)	Set identifier name for this view, retrieved by *findViewById()*
android:longClickable	setLongClickable(Boolean)	Defines whether this view reacts to long click events.
android:minHeight	setMinimumHeight(int)	Defines the minimum height of the view.
android:minWidth	setMinimumWidth(int)	Defines the minimum width of the view.

To be continued…

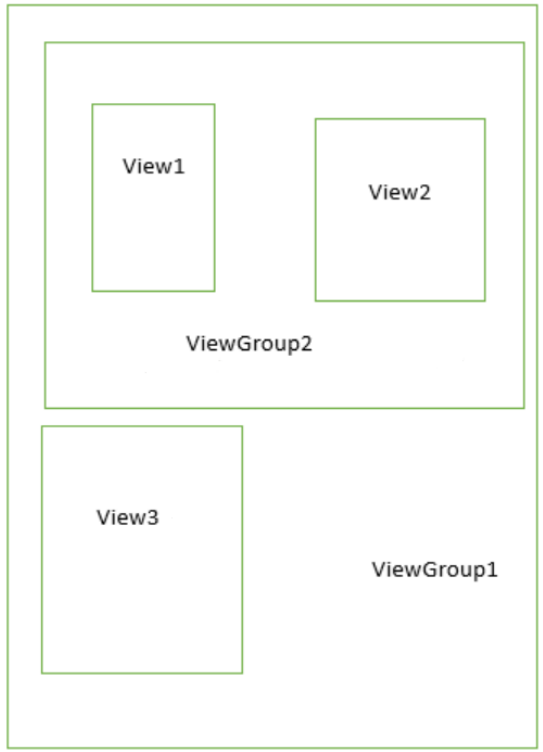
View

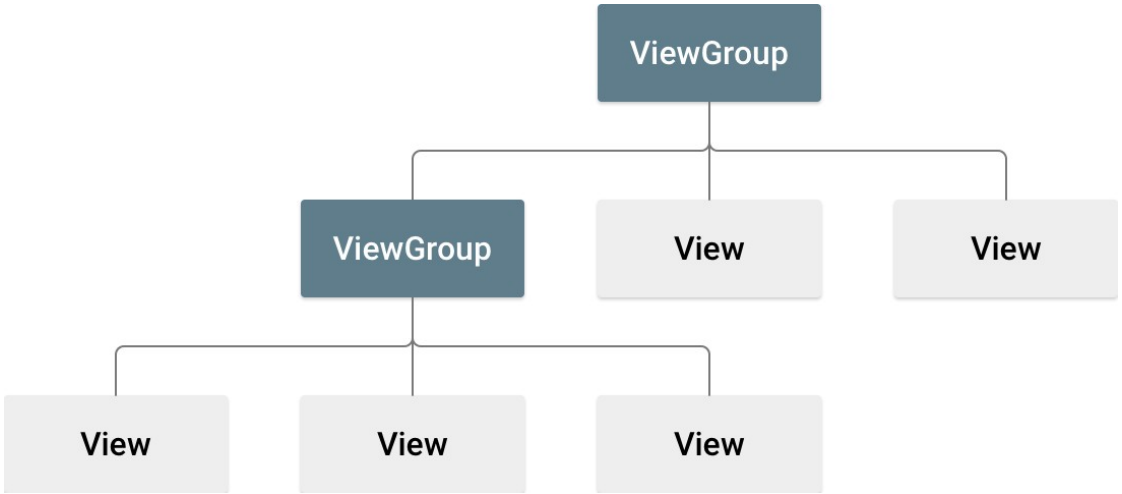
XML attributes and corresponding methods

Attribute	Methods	Description
android:onClick		Name of the method in this View's context to invoke when the view is clicked.
android:padding	setPaddingRelative(int, int, int, int)	Sets the padding, in pixels, of all four edges
android:paddingBottom	setPaddingRelative(int, int, int, int)	Sets the padding, in pixels, of the bottom edge
android:paddingEnd	setPaddingRelative(int, int, int, int)	Sets the padding, in pixels, of the end edge under horizonal alignment
android:paddingLeft	setPadding(int, int, int, int)	Sets the padding, in pixels, of the left edge
android:paddingRight	setPadding(int, int, int, int)	Sets the padding, in pixels, of the right edge
android:paddingStart	setPaddingRelative(int, int, int, int)	Sets the padding, in pixels, of the start edge under horizonal alignment
android:paddingTop	setPaddingRelative(int, int, int, int)	Sets the padding, in pixels, of the top edge
android:visibility	setVisibility(int)	Controls the initial visibility of the view.

ViewGroup

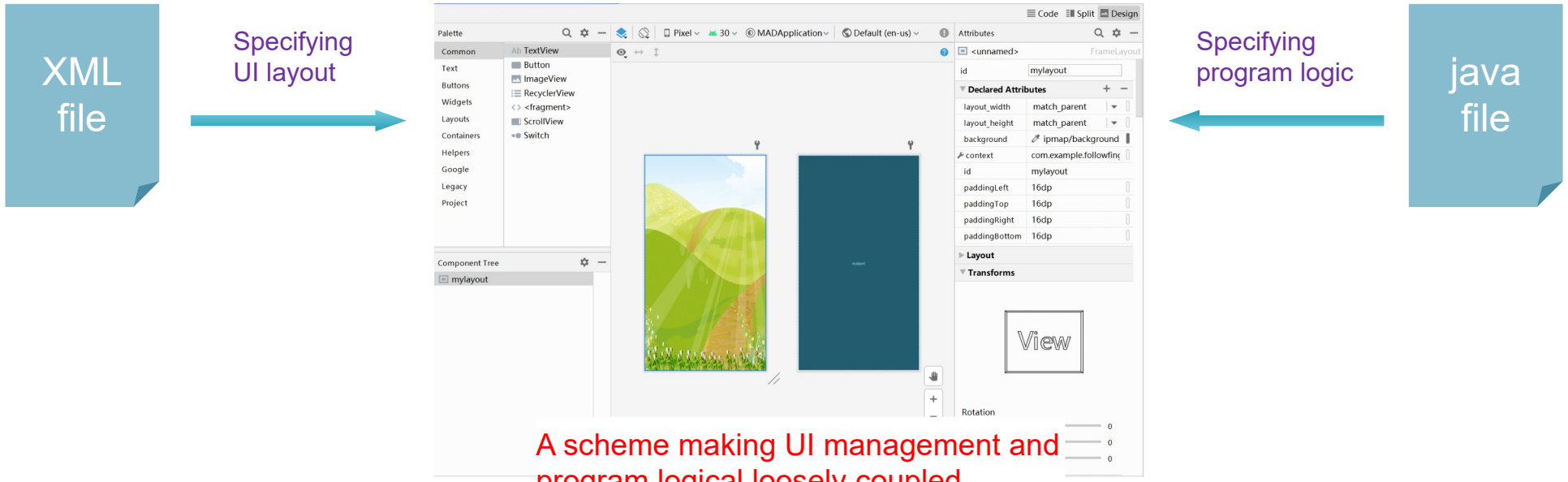
- **is an invisible container that defines the layout structure for *View* and other *ViewGroup* objects**
  - **usually called "layouts" can be one of many types that provide a different layout structure**
  - **is an abstract class and usually inherited as containers/layout**





- **However, the actual development seldomly employs View and ViewGroup classes to design UI, it prefers to select the sub-classes of View and ViewGroup classes.**
- **The developers often choose customizing UI through extending View class.**

Controlling UI with XML layout file



A scheme making UI management and program logical loosely coupled

- **Procedures**
  - Create XML layout file under res\layout folder, and the file name should accord with Java naming rules
  - Display and connect XML file content with Java source code in Activity
    - setContentView(R.layout.activity_main);

Small demo

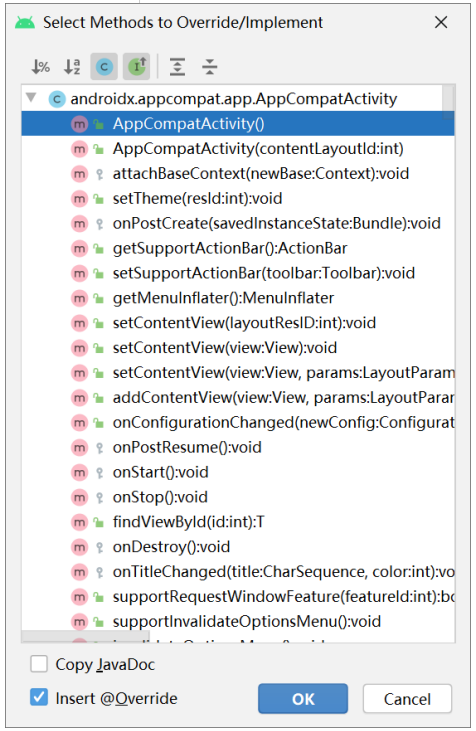
- **Import photo resource**
  - Location:…\AndroidStudioProjects\(project name)\(module name)\src\main\res\mipmap-xhdpi
    - Mipmap-xhdpi for short
  - There are several folders corresponding to various dpi(s)
  - Directly copy-paste or import photo
- **Create necessary constant**
  - Location: res\values\strings.xml
  - One <string> element specifies a string constant
    - Name attribute of the element declares constant name
    - Value of the elements specifies constant value

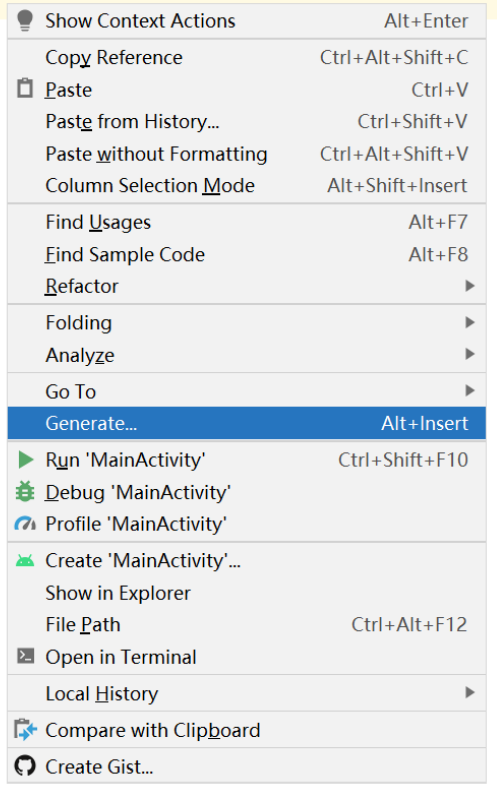
```
<resources>
    <string name="app_name">FirstAppDemo</string>
    <string name="start">Start Game</string>
</resources>
```

Customizing View components

- **Procedures**
  - Create a Java class extending android.view.View class and override the constructor
  - Override other member method(s) according to the requirements
    - Select the necessary methods from the list
  - In Activity, create and instantiate the customized View class, then add it within the layout manager

```
package com.example.firstappdemo;

import ...

public class MainActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }

}
```





Small demo

- Load necessary photo resources
- Edit res\layout\activity_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<FrameLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="@mipmap/background"
    android:id="@+id/mylayout"
    android:paddingBottom="16dp"
    android:paddingLeft="16dp"
    android:paddingRight="16dp"
    android:paddingTop="16dp"
    tools:context="com.example.followfingerbunny.MainActivity">
</FrameLayout>
```

背景图片

布局 id

Small demo

- Create a Java class extending android.view.View class
  - override the constructor and onDraw() method
    - Select the necessary methods from the list

```
public class RabbitView extends android.view.View{
    public float bitmapX; //X position of rabbit
    public float bitmapY; //Y position of rabbit
    public RabbitView(Context context){
        super(context);
        bitmapX = 210; //default X position of rabbit
        bitmapY = 130; //default Y position of rabbit
    }

    @Override
    protected void onDraw(Canvas canvas) {
        super.onDraw(canvas);
        Paint paint = new Paint(); //create and instantiate Paint object
        Bitmap bitmap = BitmapFactory.decodeResource(this.getResources(), rabbit); //generate bitmap object according to
photo
        canvas.drawBitmap(bitmap,bitmapX,bitmapY,paint); //draw the rabbit on the canvas
        if(!bitmap.isRecycled()){
            bitmap.recycle();
        }
    }
}
```

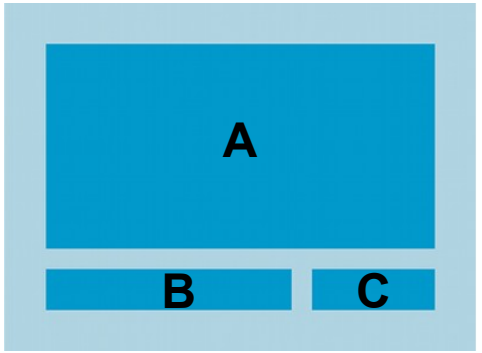
Small demo

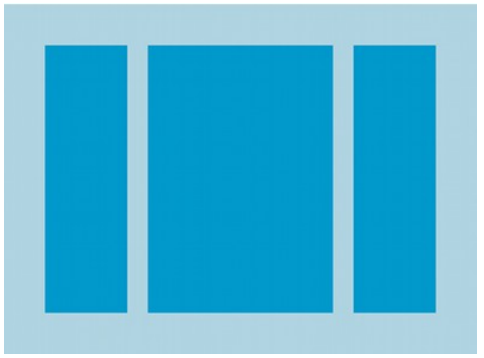
- **In Activity, create and instantiate the customized View class, then add it within the layout manager**

```
public class MainActivity extends AppCompatActivity {
    private static final String TestApp = "DefaultMessage";
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        FrameLayout frameLayout = (FrameLayout)findViewById(R.id.mylayout); //Obtain the layout manager of UI

        final RabbitView rabbit = new RabbitView(this);          //create and instantiate Rabbit class
        rabbit.setOnTouchListener(new View.OnTouchListener(){
            @Override
            public boolean onTouch(View v, MotionEvent event) {
                rabbit.bitmapX = event.getX(); //display the rabbit according to the touch
                rabbit.bitmapY = event.getY();
                rabbit.invalidate();
                Log.d(TestApp,"within the event handler");
                return true;
            }
        });
        frameLayout.addView(rabbit);
    }
}
```

Layout manager

- Manage the position and size
- Suggest to manage layouts with XML files
- Five common layouts you would meet with
  - RelativeLayout
    - specify the location of child objects relative to each other (child A to the left of child B) or to the parent (aligned to the top of the parent)
  - LinearLayout
    - Organizes its children into a single horizontal or vertical row
    - It creates a scrollbar if the length of the window exceeds the length of the screen.
  - FrameLayout
    - Organizes all the components on the top-left corner layer by layer





Layout manager

- Manage the position and size
- Suggest to manage layouts with XML files
- Five common layouts you would meet with
  - TableLayout
    - Arranges its children into rows and columns
  - AbsoluteLayout
    - This class was deprecated in API level 3
    - Lets developers specify exact locations
  - WebView
    - Displays web pages

RelativeLayout

```
<RelativeLayout xmlns:android=http://schemas.android.com/apk/res/android
        Attribute list
>
</RelativeLayout>
```

Attribute	Methods
android:gravity	Specifies how an object should position its content, on both the X and Y axes, within its own bounds.
android:ignoreGravity	Indicates what view should not be affected by gravity.

- **RelativeLayout has an inner class named as RelativeLayout.LayoutParams**
  - **LayoutParams enables to manage the locations of components**

Attribute	Methods
android:layout_above	The value is the given anchor view ID. It positions the bottom edge of this view above the given anchor view ID.
android:layout_alignBottom	The value is the given anchor view ID. It makes the bottom edge of this view match the bottom edge of the given anchor view ID.
android:layout_alignLeft	It makes the left edge of this view match the left edge of the given anchor view ID.
android:layout_below	The value is the given anchor view ID. Positions the top edge of this view below the given anchor view ID.

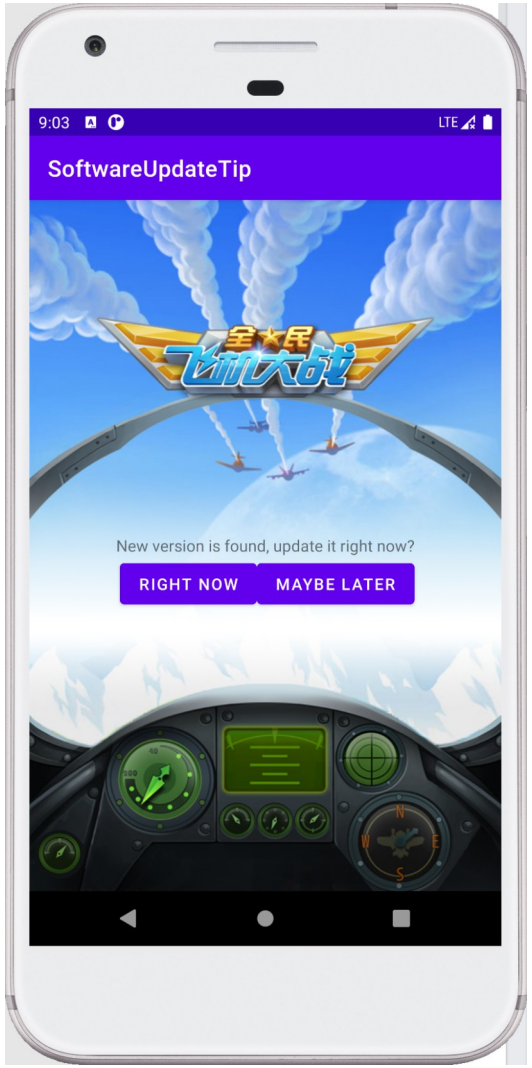
RelativeLayout

- **RelativeLayout has an inner class named as RelativeLayout.LayoutParams**
  - **LayoutParams enables to manage the locations of components**

Attribute	Methods
android:layout_alignParentBottom	If true, makes the bottom edge of this view match the bottom edge of the parent.
android:layout_alignParentLeft	If true, makes the left edge of this view match the left edge of the parent.
android:layout_below	The value is the given anchor view ID. It positions the top edge of this view below the given anchor view ID.
android:layout_centerHorizontal	If true, centers this child horizontally within its parent.
android:layout_centerInParent	If true, centers this child horizontally and vertically within its parent.
android:layout_toLeftOf	The value is the given anchor view ID. Positions the right edge of this view to the left of the given anchor view ID.

Small Demo

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingBottom="16dp"
    android:paddingLeft="16dp"
    android:paddingRight="16dp"
    android:paddingTop="16dp"
    android:background="@mipmap/bg"
    tools:context=".MainActivity">
    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="New version is found, update it right now?"
        android:id="@+id/textViewSUT"
        android:layout_centerInParent="true"/>
    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Maybe later"
        android:id="@+id/button_later"
        android:layout_alignRight="@id/textViewSUT"
        android:layout_below="@id/textViewSUT"/>
    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Right now"
        android:id="@+id/button_now"
        android:layout_below="@id/textViewSUT"
        android:layout_toLeftOf="@id/button_later"/>
</RelativeLayout>
```



LinearLayout

```
<LinearLayout xmlns:android=http://schemas.android.com/apk/res/android

        Attribute list
>
</LinearLayout>
```

Each row/column only displays one component till the boundary of the view, and the more components will not be displayed

Attribute	Methods
android:orientation	Determines the arrangement of components. The attribute value could be either horizontal or vertical (default)
android:gravity	Sets the displaying positions of components within the view. The attribute values: top, bottom, left, right, center_vertical, fill_vertical, center_horizontal, fill_horizontal, center, fill, clip_vertical, clip_horizontal. The values could be simultaneously set with "|" (no space around "|")
android:layout_width android:layout_height	Specifies the basic width/height of components. The attribute values: fill_parent, match_parent, wrap_content.
android:id	Set identifier name for this view, retrieved by *findViewById().*

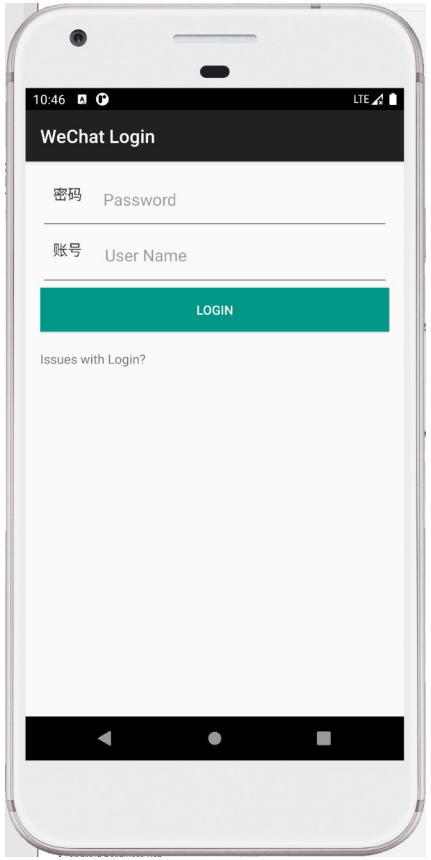
Under horizontal linear layout, the contained component's android:layout_width should not be set as match_parent or fill_parent since there will be only one component displayed in one row. In the same way, under vertical linear layout, the android:layout_height should be match_parent or fill_parent for the same reason.

LinearLayout

The common attributes of components under LinearLayout

Attribute	Methods
android:layout_gravity	Used to set the positions of the components within the located container. The attribute values: top, bottom, left, right, center_vertical, fill_vertical, center_horizontal, fill_horizontal, center, fill, clip_vertical, clip_horizontal. The values could be simultaneously set with “|” (no space around “|”)
android:layout_weight	Used to assign the weights to the components for **consuming the remaining space** within the located container.

Small Demo




```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:orientation="vertical"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingBottom="16dp"
    android:paddingLeft="16dp"
    android:paddingRight="16dp"
    android:paddingTop="16dp"
    tools:context=".MainActivity">
    <EditText
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:paddingBottom="20dp"
        android:hint="Password"
        android:drawableLeft="@mipmap/mima"/>
    <EditText
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:paddingBottom="20dp"
        android:hint="User Name"
        android:drawableLeft="@mipmap/zhanghao"/>
    <Button
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Login"
        android:textColor="#FFFFFF"
        android:background="#FF009688"/>
    <TextView
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Issues with Login?"
        android:layout_gravity="center_horizontal"
        android:paddingTop="20dp"/>
</LinearLayout>
```

```
<manifest
xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.wechatlogin">
    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/Theme.AppCompat.Light.DarkActionBar">
        <activity android:name=".MainActivity">
            <intent-filter>
                <action
android:name="android.intent.action.MAIN" />
                <category
android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>
</manifest>
```

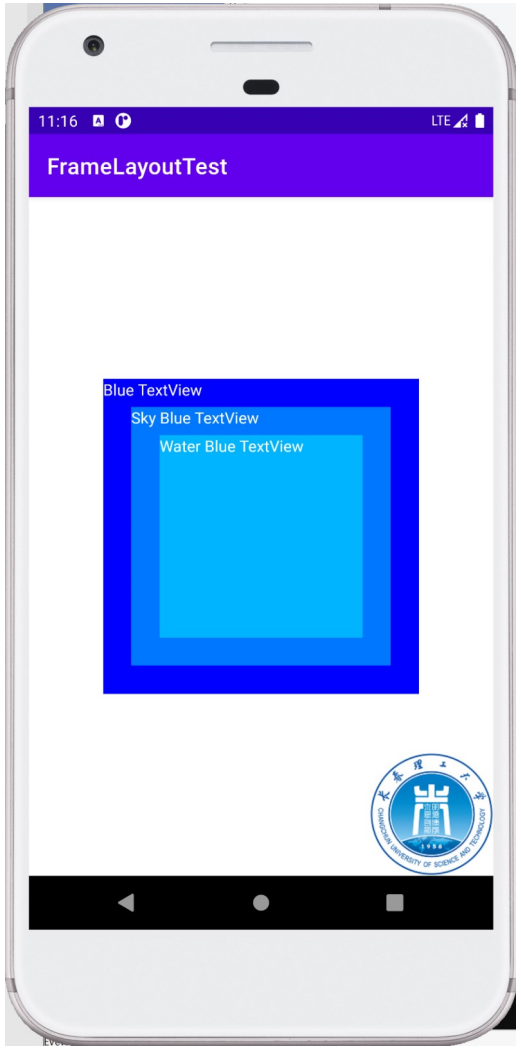
FrameLayout

```
<FrameLayout xmlns:android=http://schemas.android.com/apk/res/android

        Attribute list
>
</FrameLayout>
```

- Corresponding to each added component, a blank area (aka. frame) is created.
- All the frames are located at the top-left corner, i.e., (0,0).
- All the frames are overlaid.

Attribute	Methods
android:foreground	Determines the foreground image
android:foregroundGravity	Defines the gravity to apply to the foreground drawable.

Small Demo




```
<FrameLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:foreground="@mipmap/logo"
    android:foregroundGravity="bottom|right"
    tools:context=".MainActivity">
    <TextView
        android:id="@+id/textView1_FrameLayoutTest"
        android:layout_width="280dp"
        android:layout_height="280dp"
        android:layout_gravity="center"
        android:background="#FF0000FF"
        android:textColor="#FFFFFF"
        android:text="Blue TextView"/>
    <TextView
        android:id="@+id/textView2_FrameLayoutTest"
        android:layout_width="230dp"
        android:layout_height="230dp"
        android:layout_gravity="center"
        android:background="#FF0077FF"
        android:textColor="#FFFFFF"
        android:text="Sky Blue TextView"/>
    <TextView
        android:layout_width="180dp"
        android:layout_height="180dp"
        android:layout_gravity="center"
        android:background="#FF00B4FF"
        android:textColor="#FFFFFF"
        android:text="Water Blue TextView"/>
</FrameLayout>
```

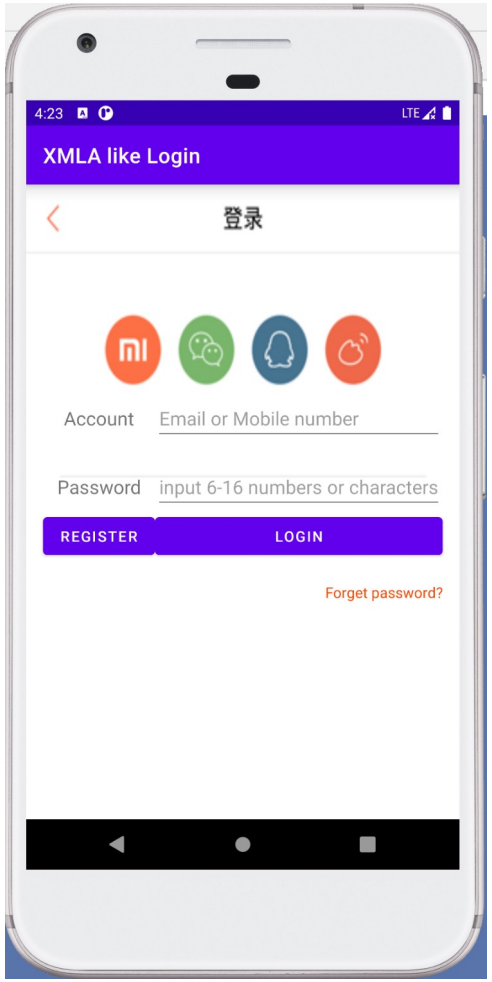
TableLayout

```
<TableLayout xmlns:android=http://schemas.android.com/apk/res/android
        Attribute list
>
<TableRow attribute list> UI components to add </TableRow>
More <TableRow>
</TableLayout>
```

- **TableLayout inherits LinearLayout, and it completely support all the XML attributes of LinearLayout.**

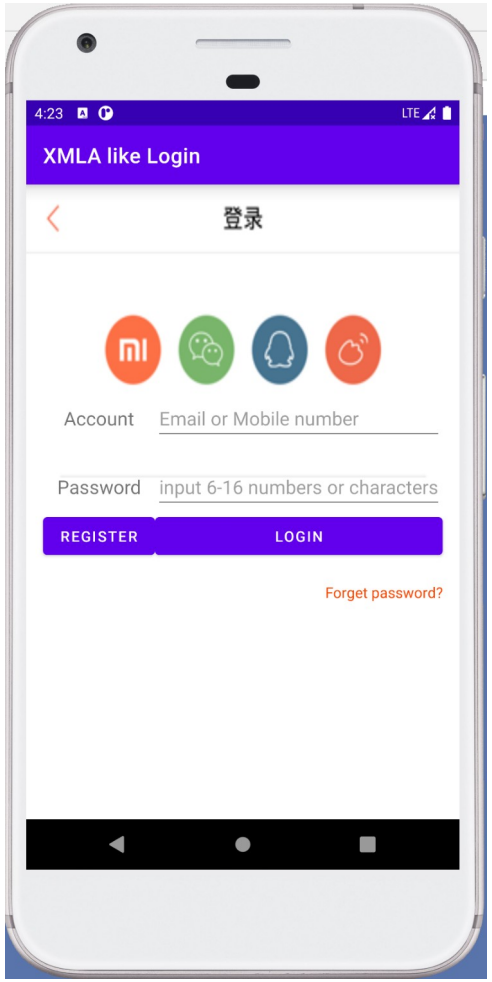
Attribute	Methods
android:collapseColumns	Set the number (from 0) of the column to be collapsed (hide), multiple numbers separated by “,”
android:shrinkColumns	Set the number (from 0) of the column to be shrunk, multiple numbers separated by “,”
android:stretchColumns	Set the number (from 0) of the column to be stretched, multiple numbers separated by “,”

Small Demo



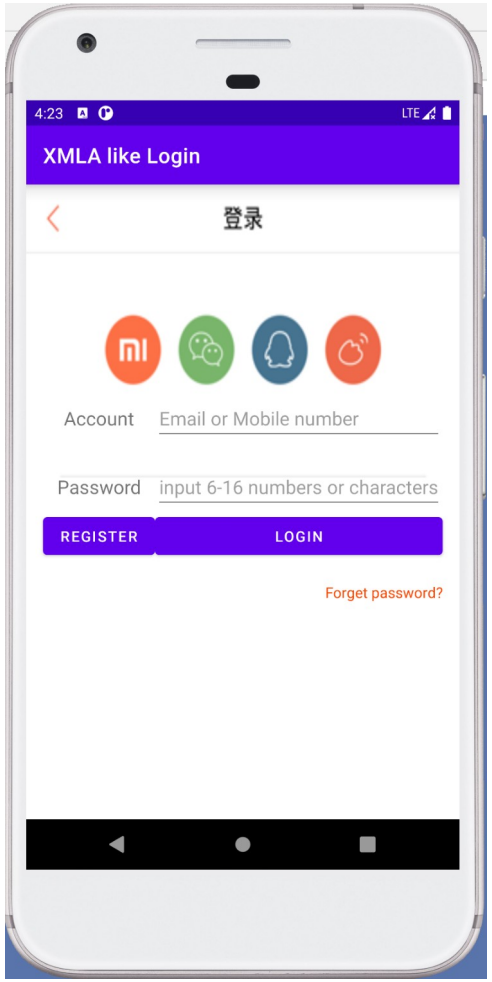

```
<TableLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="@mipmap/biaoge"
    android:stretchColumns="0,3"
    tools:context="com.example.xmlalikelogin.MainActivity">
    <TableRow
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:paddingTop="200dp">
        <TextView />
        <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:textSize="18sp"
            android:text="Account"
            android:gravity="center_horizontal"/>
        <EditText
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:hint="Email or Mobile number"/>
        <TextView />
    </TableRow>
```

Small Demo




```
<TableRow
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:paddingTop="20dp">
    <TextView />
    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textSize="18sp"
        android:text="Password"
        android:gravity="center_horizontal"/>
    <EditText
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:hint="input 6-16 numbers or characters"/>
    <TextView />
</TableRow>
<TableRow
    android:layout_width="wrap_content"
    android:layout_height="wrap_content">
    <TextView />
    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Register"/>
    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:background="#FF8247"
        android:text="Login"/>
    <TextView />
</TableRow>
```

Small Demo




```
        <TableRow
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:paddingTop="20dp">
            <TextView />
            <TextView />
            <TextView
                android:text="Forget password?"
                android:textColor="#FF4500"
                android:gravity="right"/>
            <TextView />
        </TableRow>
    </TableLayout>
```

GridLayout

```
<GridLayout xmlns:android=http://schemas.android.com/apk/res/android
        Attribute list
>
</GridLayout>
```

- It allows one component to take multiple rows/columns, which cannot be achieved by TableLayout.

Attribute	Methods
android:columnCount android:rowCount	The maximum number of the columns/rows
android:orientation	To control the 'direction' in which default row/column indices are generated when they are not specified in a component's layout parameters.

- GridLayout.LayoutParams controls the arrangement of the contained components.

Attribute	Methods
android:layout_column	Determines the specific column of the current component
android:layout_columnSpan	Determines the occupied columns (horizontal direction) by index (from 0)
android:columnWeight	Determines the weight in horizontal direction of the current component, i.e., the ratio of taking the remaining space

Small Demo


```
<GridLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingBottom="16dp"
    android:paddingLeft="16dp"
    android:paddingRight="16dp"
    android:paddingTop="16dp"
    android:background="@mipmap/bg"
    android:columnCount="6"
    tools:context=".MainActivity">
    <ImageView
        android:id="@+id/imageView1"
        android:src="@mipmap/a1"
        android:layout_gravity="end"
        android:layout_columnSpan="4"
        android:layout_column="1"
        android:layout_row="0"
        android:layout_marginRight="5dp"
        android:layout_marginBottom="20dp"/>
    <ImageView
        android:id="@+id/imageView2"
        android:src="@mipmap/ico2"
        android:layout_column="5"
        android:layout_row="0"/>
    <ImageView
        android:id="@+id/imageView3"
        android:src="@mipmap/ico1"
        android:layout_column="0"
        android:layout_row="1"/>
    <ImageView
        android:id="@+id/imageView4"
        android:src="@mipmap/b1"
        android:layout_row="1"
        android:layout_marginBottom="20dp"/>
```

Small Demo


```
        <ImageView
            android:id="@+id/imageView5"
            android:src="@mipmap/a2"
            android:layout_gravity="end"
            android:layout_columnSpan="4"
            android:layout_column="1"
            android:layout_row="2"
            android:layout_marginRight="5dp"
            android:layout_marginBottom="20dp"/>
        <ImageView
            android:id="@+id/imageView6"
            android:src="@mipmap/ico2"
            android:layout_column="5"
            android:layout_row="2"/>
        <ImageView
            android:id="@+id/imageView7"
            android:src="@mipmap/ico1"
            android:layout_column="0"
            android:layout_row="3"/>
        <ImageView
            android:id="@+id/imageView8"
            android:src="@mipmap/b2"
            android:layout_row="3"
            android:layout_marginBottom="20dp"/>
    </GridLayout>
```

Embedding Layouts

- **Principles**
  - Root layout must contain xmlns attribute
  - Within one layout file, there is at most one root layout
    - If there are more than one layouts, a root layout should be employed to hold all the other layouts
  - The number of layers contain the embedded layouts should not be too high to prevent decreasing loading speed