Android Development

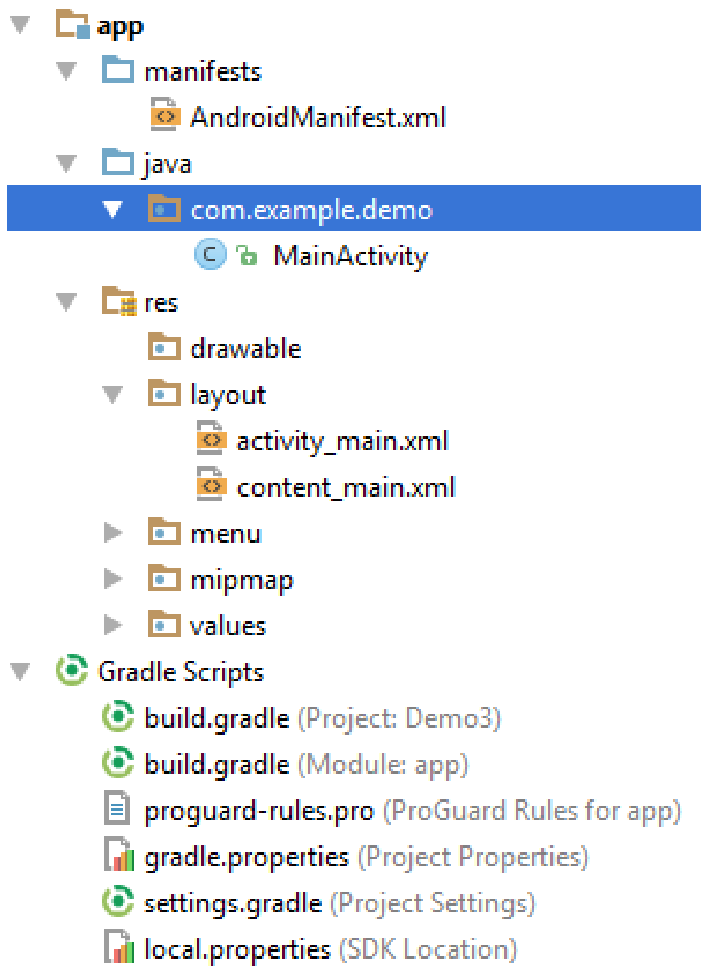
Lean and mean introduction

Based on a presentation by Mihail L. Sichitiu

Applications

- Written in Java (it’s possible to write native code – will not cover that here)
- Good separation (and corresponding security) from other applications:
  - Each application runs in its own process
  - Each process has its own separate VM
  - Each application is assigned a unique Linux user ID – by default files of that application are only visible to that application (can be explicitly exported)

Project structure



```
▼ app
  ▼ manifests
      AndroidManifest.xml
  ▼ java
    ▼ com.example.demo
        MainActivity
  ▼ res
      drawable
    ▼ layout
        activity_main.xml
        content_main.xml
    ▶ menu
    ▶ mipmap
    ▶ values
▼ Gradle Scripts
    build.gradle (Project: Demo3)
    build.gradle (Module: app)
    proguard-rules.pro (ProGuard Rules for app)
    gradle.properties (Project Properties)
    settings.gradle (Project Settings)
    local.properties (SDK Location)
```

App manifest

Java code

Resources

Build scripts

Android Manifest

- Its main purpose in life is to declare the components to the system:

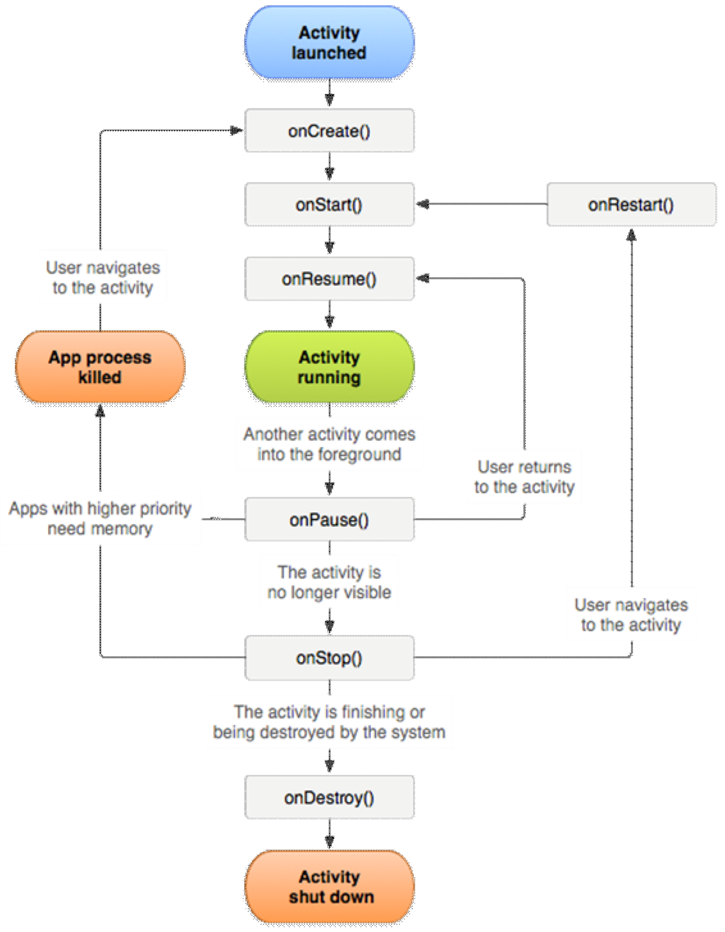
```xml
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.demo">

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:supportsRtl="true"
        android:theme="@style/AppTheme">
        <activity
            android:name="com.example.demo.MainActivity"
            android:label="@string/app_name"
            android:theme="@style/AppTheme.NoActionBar">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

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

</manifest>
```

Activities

- Basic component of most applications
- Most applications have several activities that start each other as needed
- Each is implemented as a subclass of the base Activity class

Activity life cycle

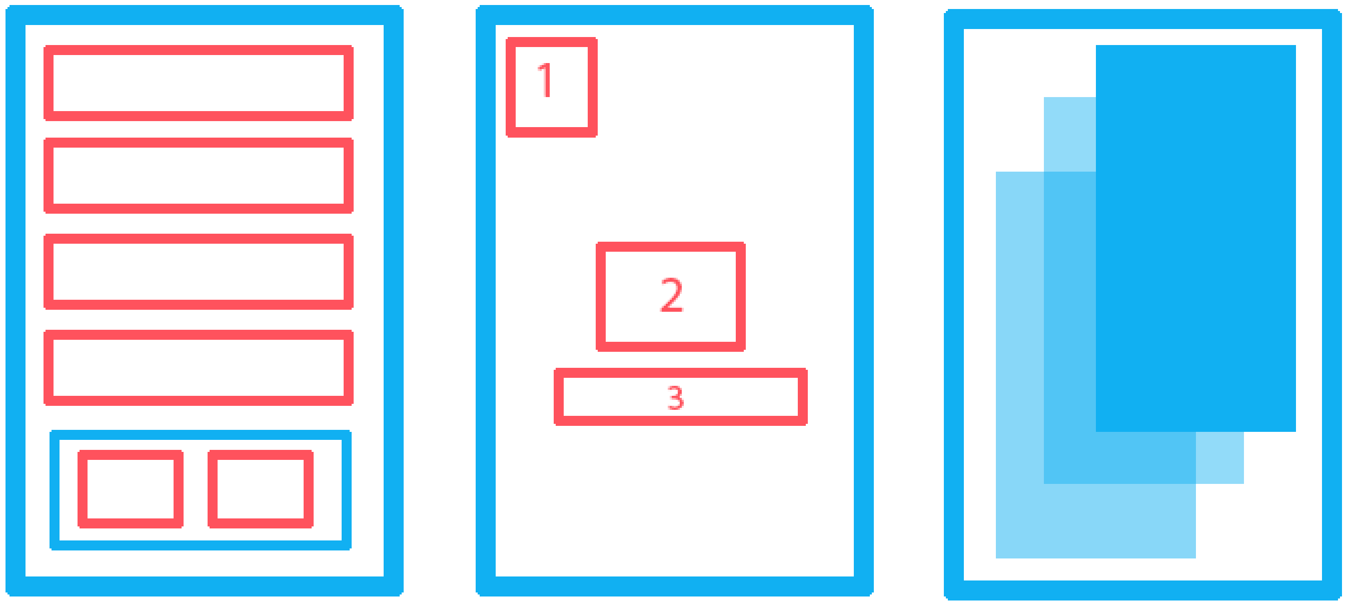
- An Android activity is focused on a single thing a user can do.
- Most applications have multiple activities


Activity launched
onCreate()
onStart()
onRestart()
User navigates to the activity
onResume()
App process killed
Activity running
Another activity comes into the foreground
User returns to the activity
Apps with higher priority need memory
onPause()
The activity is no longer visible
User navigates to the activity
onStop()
The activity is finishing or being destroyed by the system
onDestroy()
Activity shut down

Activities – The View

- Each activity has a default window to draw in (although it may prompt for dialogs or notifications)
- The content of the window is a view or a group of views (derived from View or ViewGroup)
- Example of views: buttons, text fields, scroll bars, menu items, check boxes, etc.
- View(Group) made visible via Activity.setContentView() method.

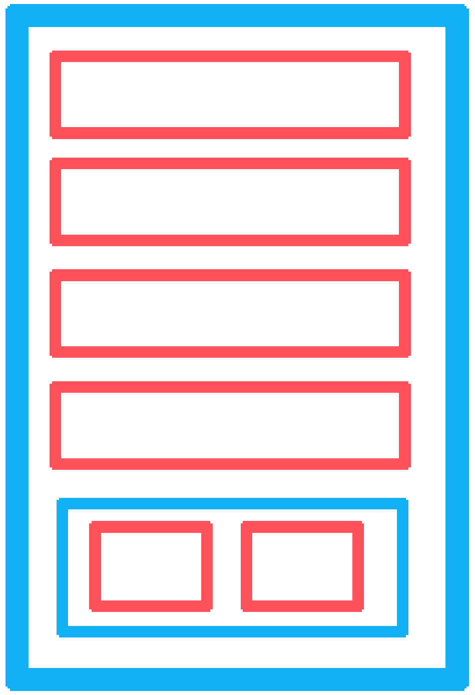
Layouts



LinearLayout

- A Layout that arranges its children in a single column or a single row

```
<LinearLayout
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical">

</LinearLayout>
```



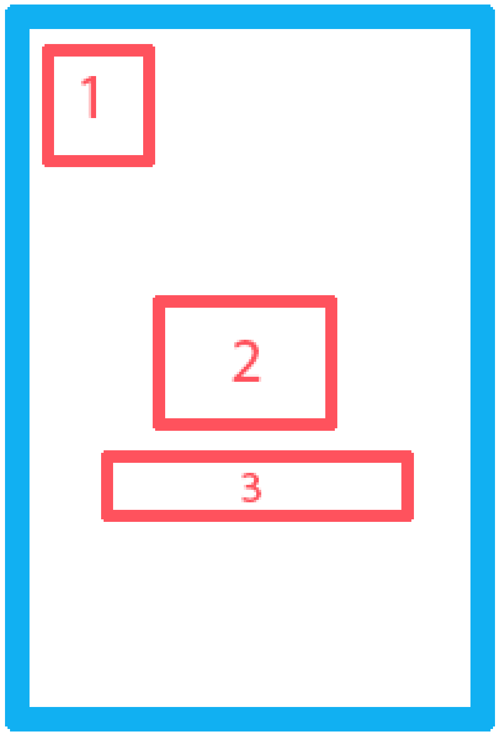
RelativeLayout

- A Layout where the positions of the children can be described in relation to each other or to the parent.

Child 1 is relative to the top left corner of the screen.

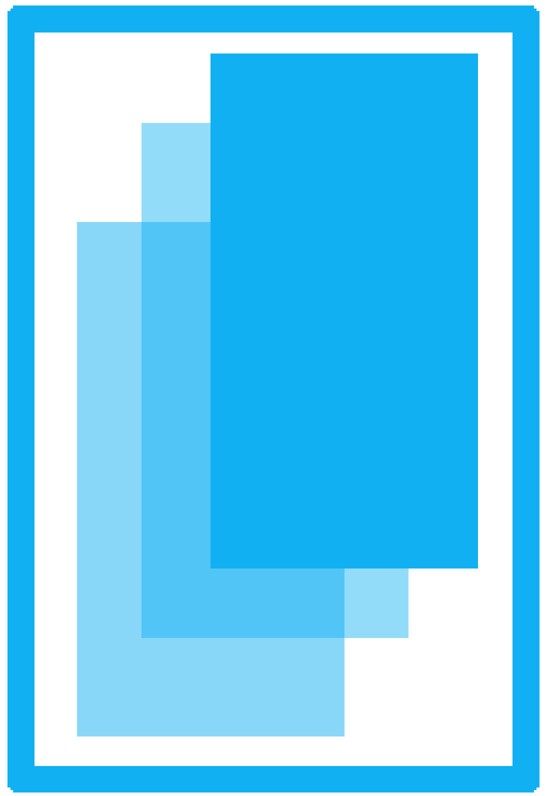
Child 2 is relative to the center of the screen.

Child 3 is positioned below child 2



FrameLayout

- A layout that stacks views along the z-axis.

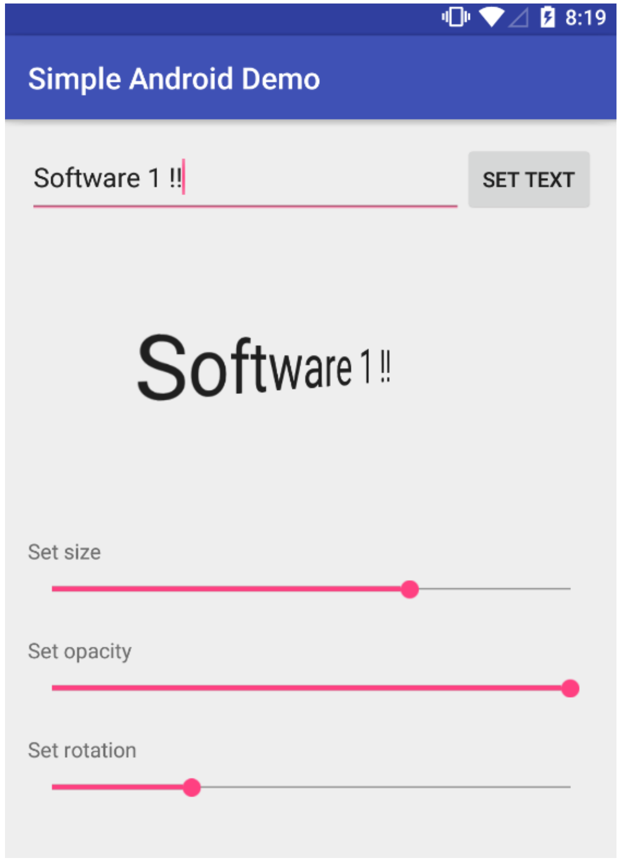


Demo 1 – UI elements

Introduction to basic UI elements.

We will meet and learn to control the TextView, EditText, Button and SeekBar views.

https://github.com/AviranAbady/AndroidDemo1



Demo 2 – Memory Game

Simple memory game using a Grid recycler view.

https://github.com/AviranAbady/AndroidMemoryGameDemo