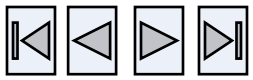
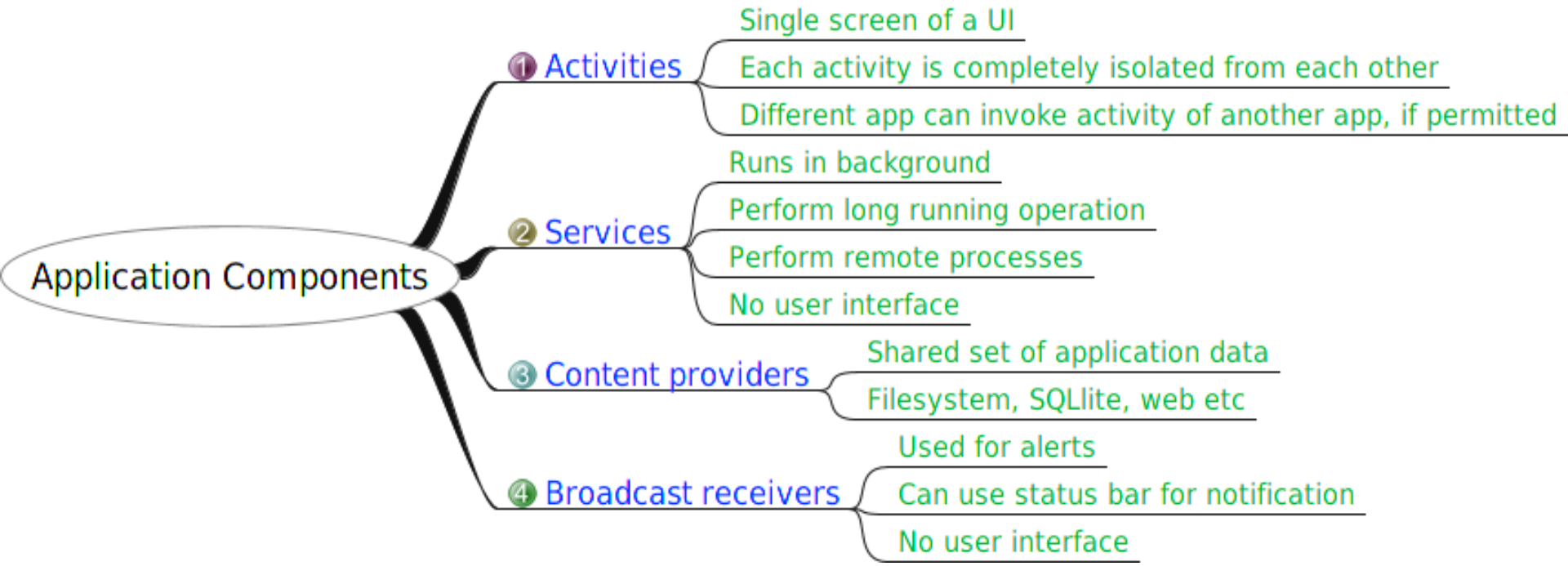
Android App Components

CSCI3310 Mobile Computing & Application Development



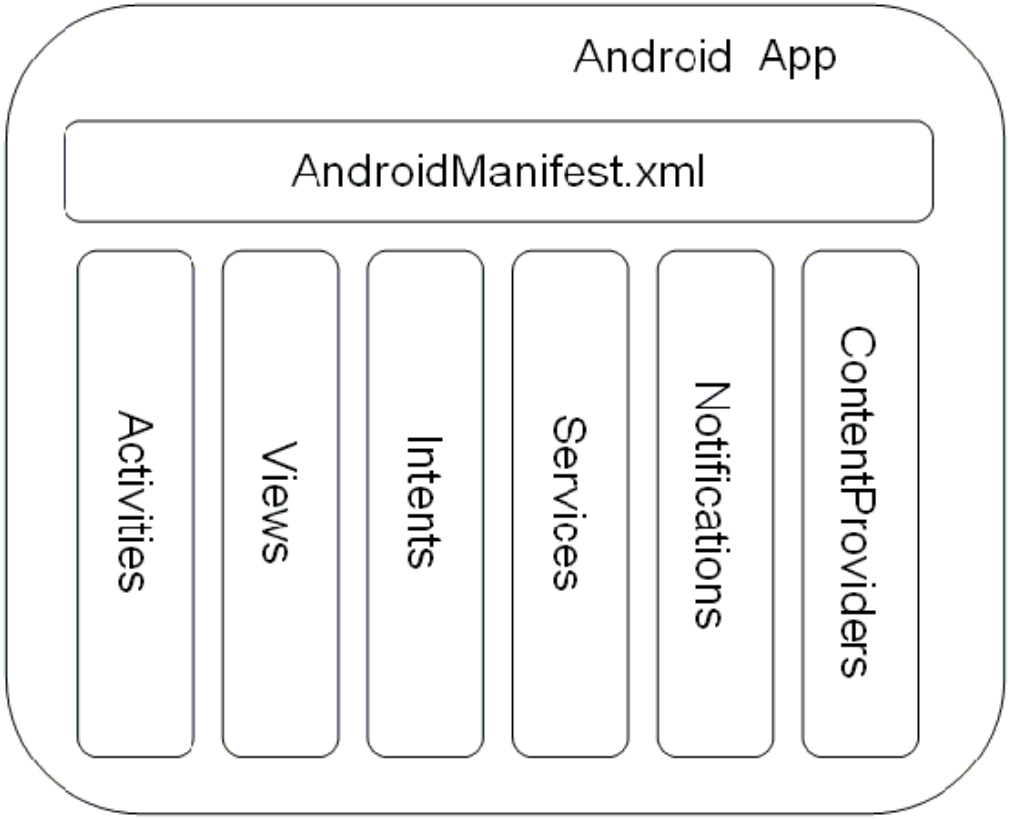
Android App Anatomy

Android application is component-based



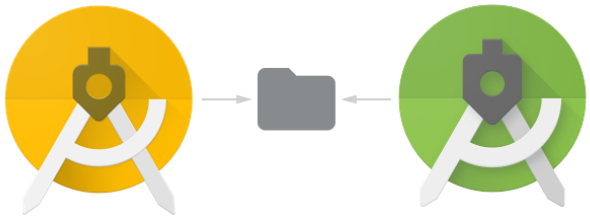
Android App Anatomy

- Android App is **components** based



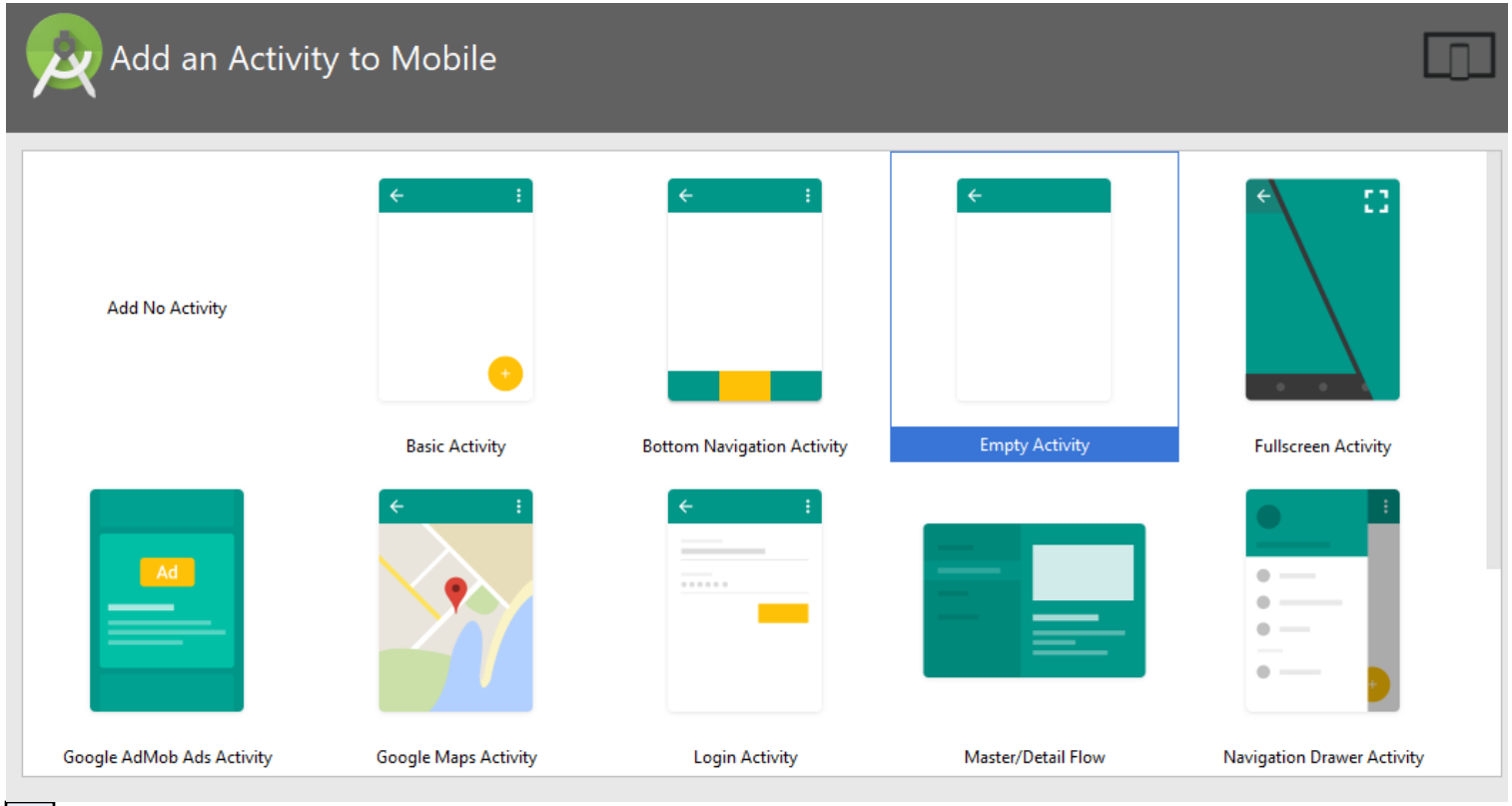
Development platform

- Once supports several different IDEs e.g. Eclipse, Visual Studio
- **Android Studio(AS) is now the official environment**
- For all platforms, the following are needed in addition to AS
  - Sun's Java Development Kit (JDK)
  - Phone driver
  - Additional emulator system images
- Windows, Mac, Linux are all supported



App Module

Collection of Activity templates provided with different sources files, resource files and build setting



Covered in Lab

Android Project Structure

Project structure in Android Studio

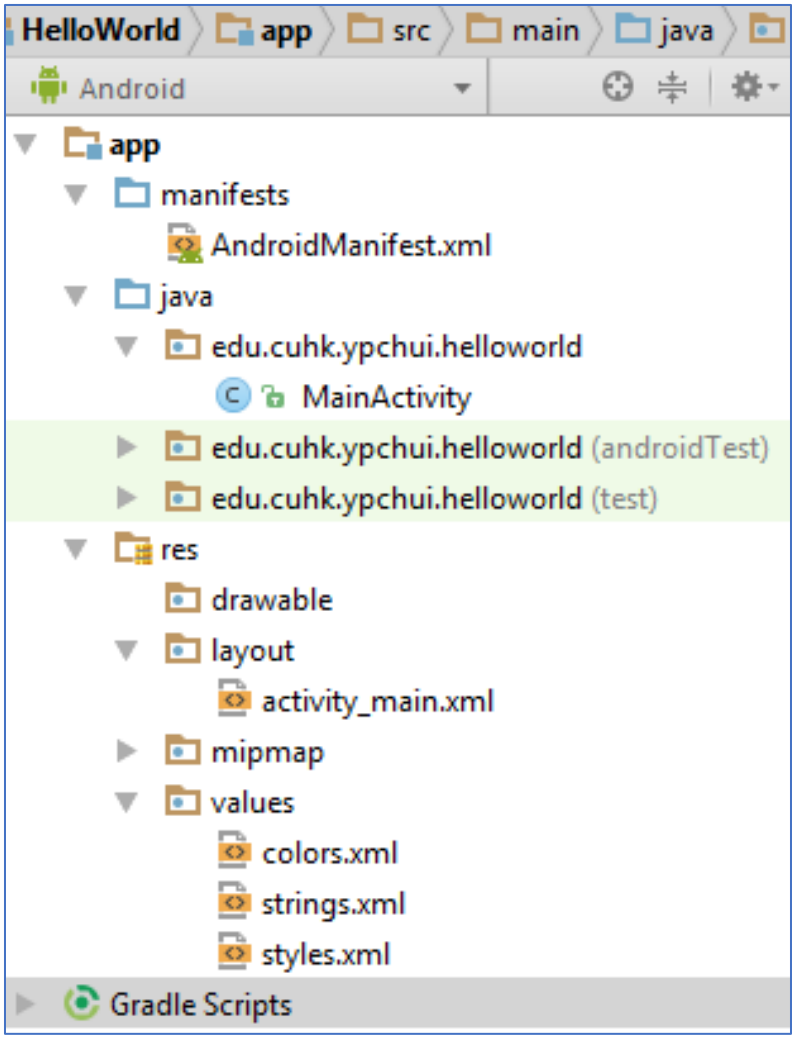
- AndroidManifest.xml
- java
  - MainActivity.java

    *[the main **Controller** entry]*
- res
  - activity_main.xml

    *[the **Views** are here]*
  - strings.xml

    [for different languages]
  - styles.xml

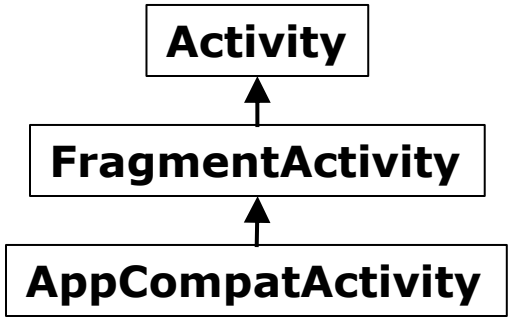
    [for different UI styles]



Covered in Lab

Android Java (java/)

- Contains the source, separated by package names
- All the Activity class are in java/ e.g. the default empty activity MainActivity.java looks like this:




```
package edu.cuhk.csci3310.helloworld;

import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;

public class MainActivity extends AppCompatActivity {

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

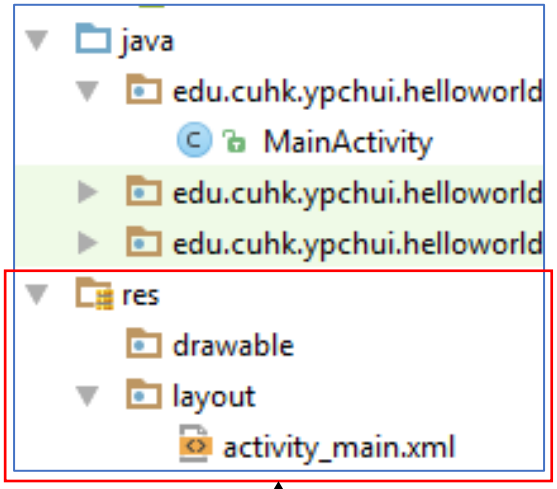
The support library will be superseded by AndroidX, e.g.

```
import androidx.appcompat.app.AppCompatActivity;
```

Covered in Lab

Android Java (java/)

- Contains the source, separated by package names
- All the Activity class are in java/ e.g. the default empty activity MainActivity.java looks like this:



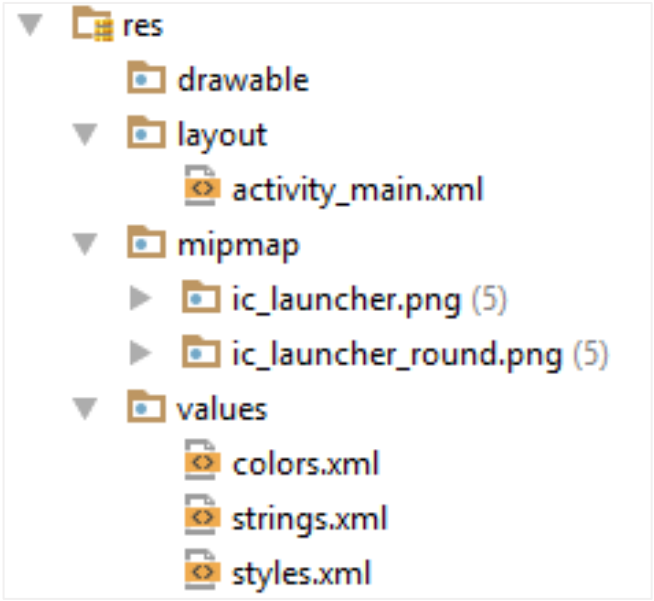

```
package edu.cuhk.csci3310.helloworld;

import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;

public class MainActivity extends AppCompatActivity {

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

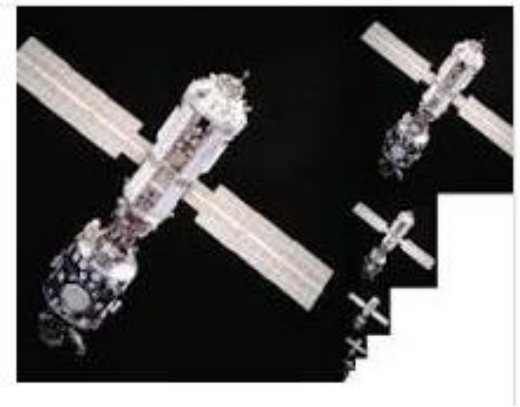
link to res/

Covered in Lab

Android Resource (res/)

- Non-code resources, e.g. XML, layouts, UI Strings, mipmap images
- Accessed in java through the "R" class, e.g.
  - res/drawable/ accessed from **R.drawable** class
  - res/layout/ accessed from **R.layout** class





Covered in Lab

Android Resource (res/)

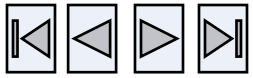
- Android Strings.xml
  - Define strings (and their formatting/styling)
  - Provide a mean for easy language translating, e.g.:

in values/strings.xml

```
<resources>
    <string name="app_name">Hello World</string>
</resources>
```

in values-fr/strings.xml

```
<resources>
    <string name="app_name">Bonjour le monde</string>
</resources>
```



Covered in Lab

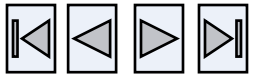
Android Resource (res/)

- Android Styles.xml, like .css in web design
  - specify properties e.g. height, padding, font size/color etc.

```
<resources>
<style name="MyTextStyle" parent="Theme.AppCompat.Light.DarkActionBar">
        <!-- Customize your theme here. -->
        <item name="colorPrimary">@color/colorPrimary</item>
        <item name="colorPrimaryDark">@color/colorPrimaryDark</item>
        <item name="colorAccent">@color/colorAccent</item>
    </style>
</resources>
```

  - and being used in **res/layout/:**

```
<TextView
    style:layout_width="@style/MyTextStyle"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:text="Hello World!"/>
```

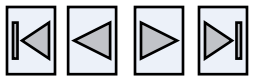


Covered in Lab

AndroidManifest.xml

- Describes the fundamental characteristics of an app and each of its **components**, the default looks like this:

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="edu.cuhk.csci3310.helloworld">
    <uses-sdk
        android:minSdkVersion="20"
        android:targetSdkVersion="26" />
    <application
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:theme="@style/AppTheme">
        <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>
```



AndroidManifest.xml

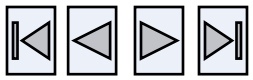
- Works as an ***interface*** between Android OS and our app, we need to declare **components** required or the OS will ignore.

E.g. to give your application access to the "contacts" information, we add a **provider**:

```
<manifest .... >
  <application android:icon="@drawable/icon"
    android:label="@string/app_name">
      <provider android:name=".PersonContentProvider"
        android:authorities="com.examples.provider.personprovider"/>
  </application>
</manifest>
```

- user permission tag is also needed

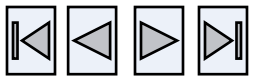
```
<uses-permission android:name="android.permission.READ_CONTACTS" />
```



Permissions

- When the application is installed, user can choose whether to allow the requested permissions and proceed with installation, or to reject installation

```
<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />

<uses-permission android:name="android.permission.INTERNET" />

<uses-permission android:name="android.permission.CALL_PHONE" />
```



Activity

Activity Manager

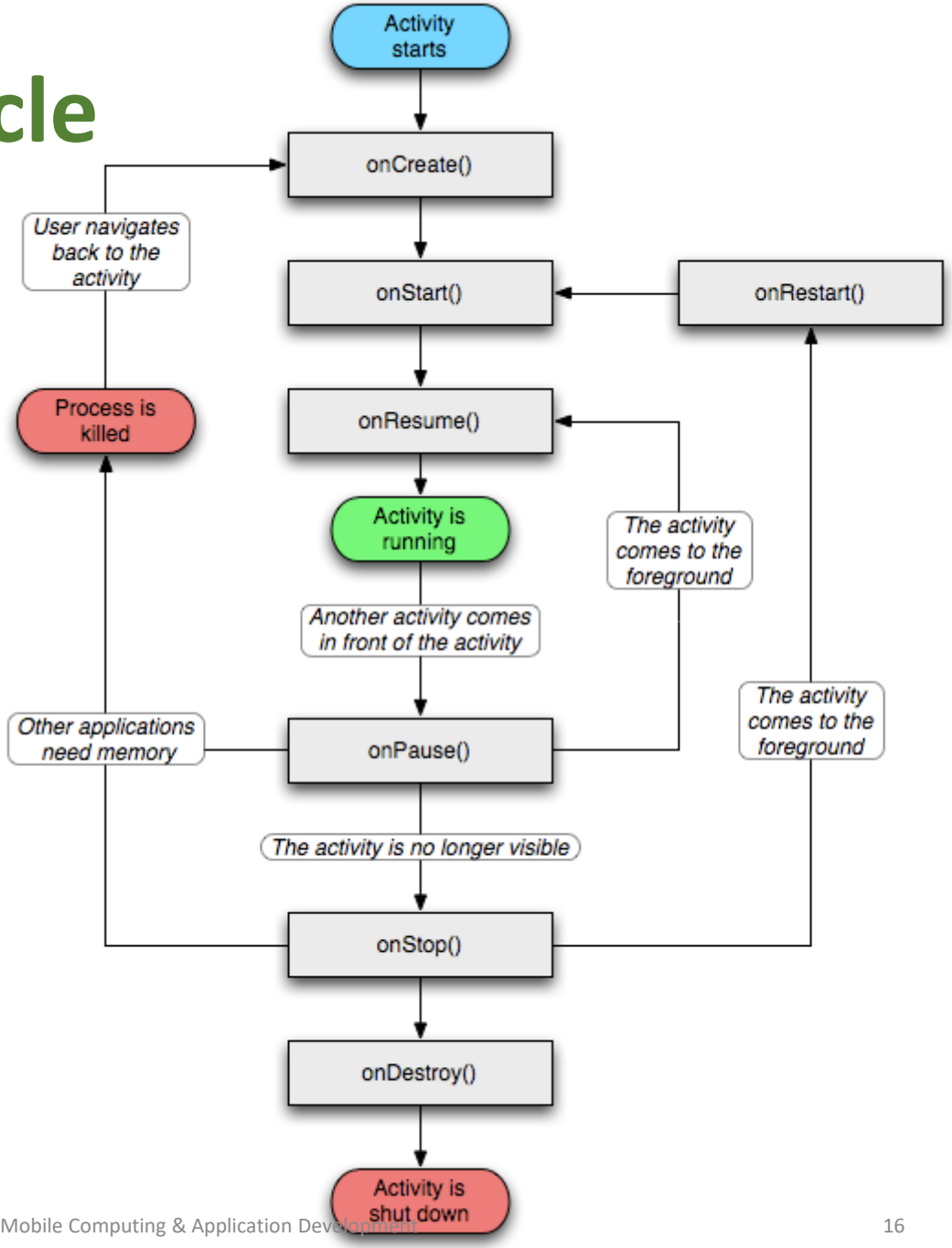
View System

- The **presentation** layer of Android application
- Every screen or window is an extension of the *android.app.Activity* class
- Activities use *Views* to form GUI
  - All UI controls are derived from *android.view.View*
    - *android.widget.Button*, *TextView*, *ListView*, *CheckBox*, …
- Activity Manager controls the lifecycle of activities

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

`Activity` "inflates" layout as part of being created

Activity Lifecycle



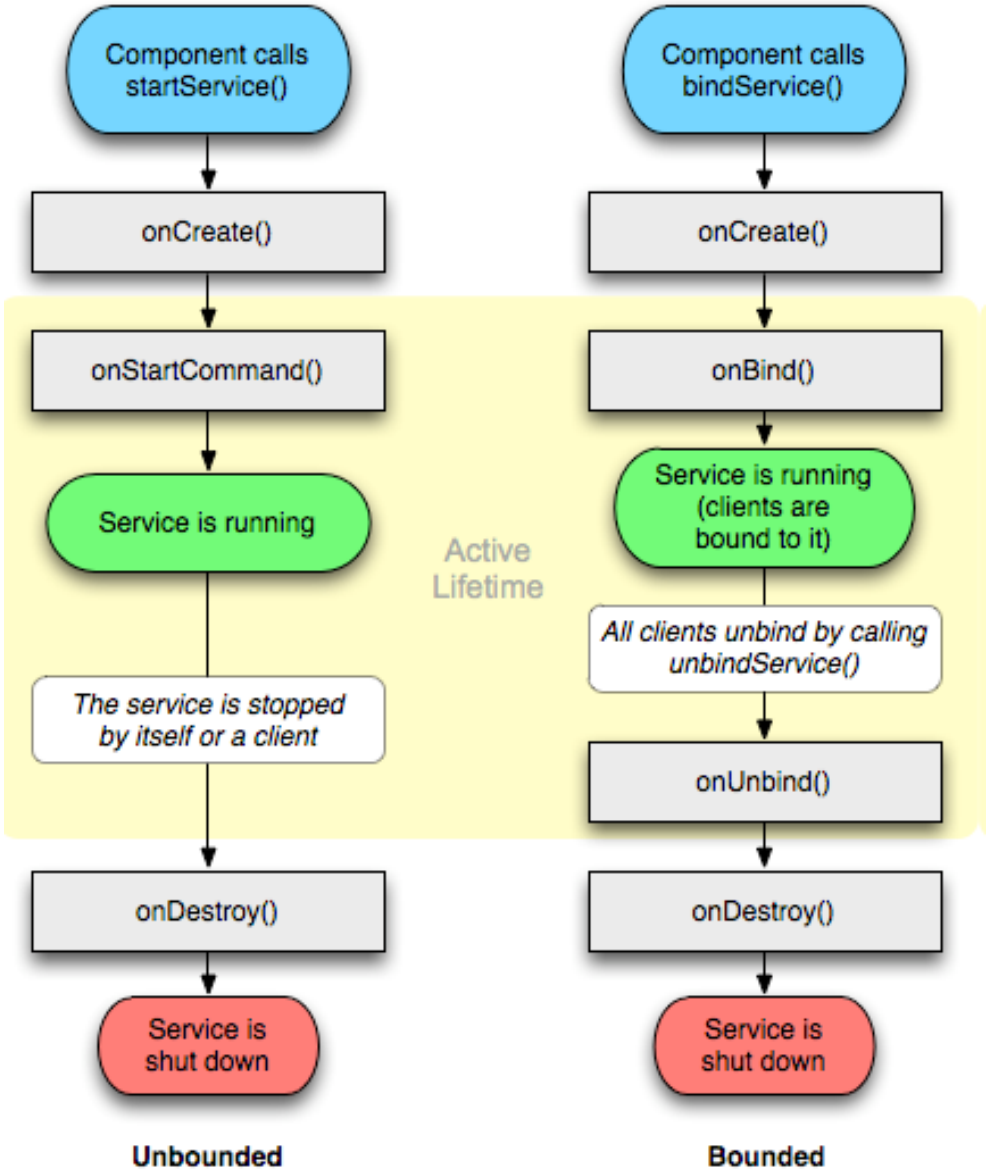
Activities

- Executable code that instantiated by either the user or the operating system
- Interact with the user and request data or services from other activities or services via queries or ***Intents***.
- Usually correspond to display screens: each Activity shows one screen to the user.
- Can be **killed by the operating system** during idle to conserve memory.

Services

- Analogous to services or daemons in desktop operating systems.
- Usually **run in the background** until shut down
- Generally don't expose a user interface.
- Classic example : an MP3 player keep playing queued files, even while the user has gone on to use other applications.

Service Lifecycle



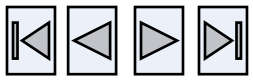
Broadcast Receiver

- **Responds to** a system-wide **announcement** of an event. e.g., battery low.

- Or an Activity or Service provides other applications with access to its functionality by executing an ***Intent Receiver.***

Content providers

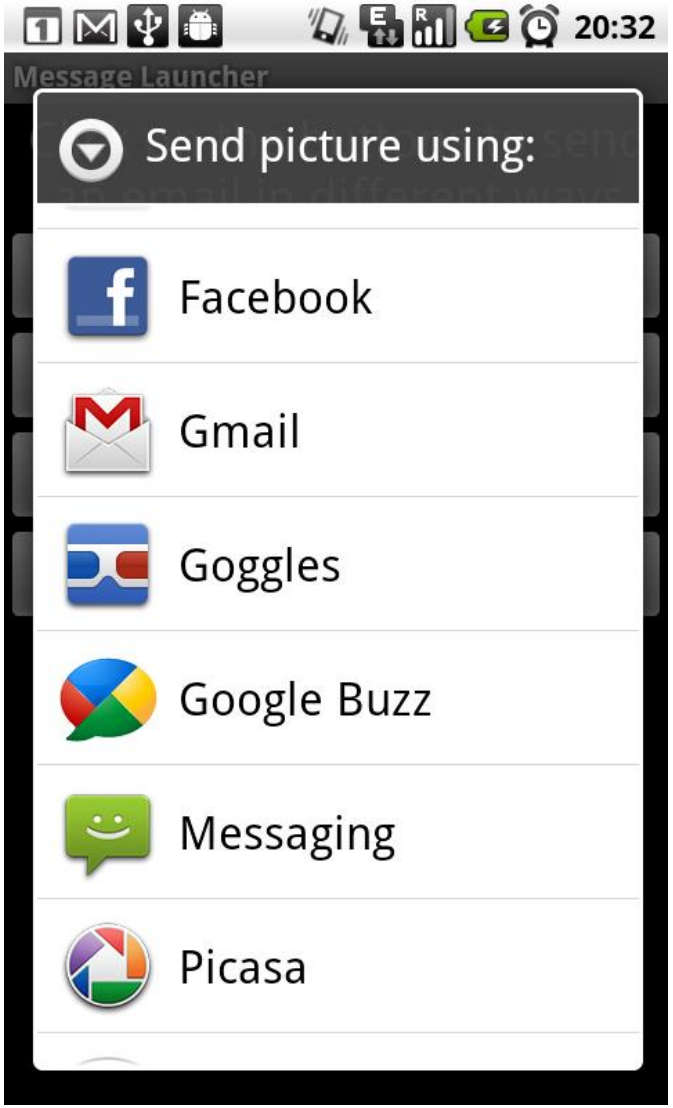
- created to **share data with other activities or services**.
- uses a standard interface in the form of a *URI* to fulfill requests for data from other applications.
- OS **checks which applications have registered** as content providers for the given *URI*, and sends the request to the appropriate application
- If there is more than one suitable content provider, OS asks the user which one he wants to use.



Android

- any application can start another application's component e.g. an app can start a camera capture activity and expect another app to finish it and send back the photo

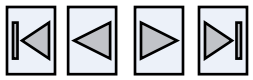
- To the user, the above case may seems that the camera capture is just a part of your app



Intent Receivers

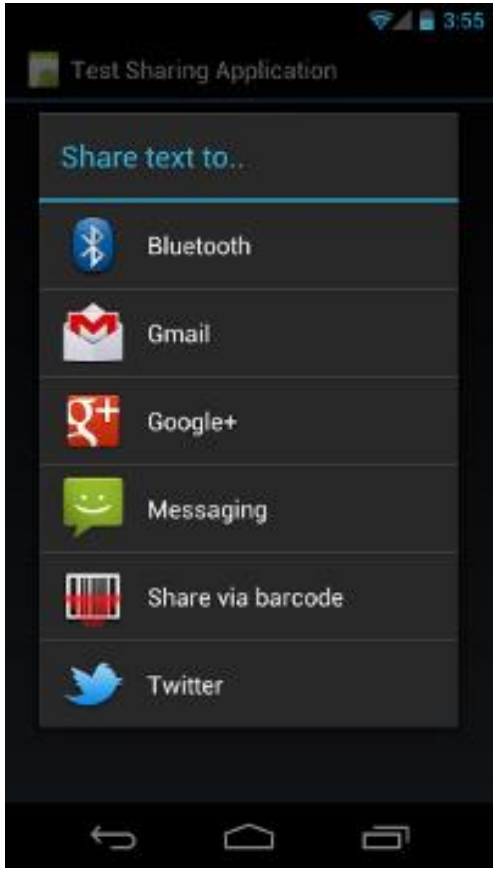
- Requesting (client) activity issues an Intent,
  - Android framework will figure out which application should receive and act on it.

- One of the key architectural elements in Android that facilitate the **creation of new** applications **from existing** applications.



Intents

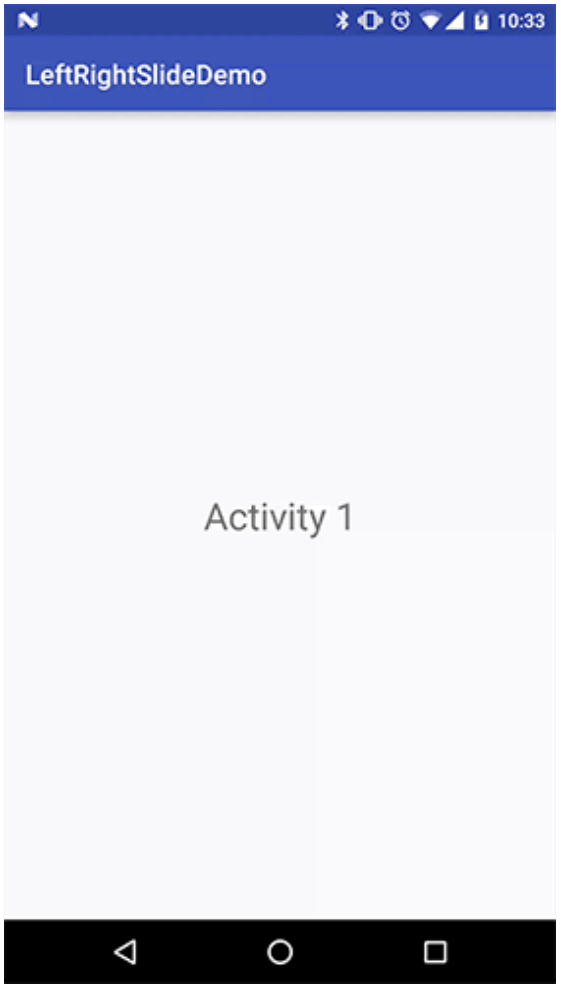
- Intent messaging is a facility for late **run-time binding** between **components** in the **same** or **different** applications.
  - an Intent object is a passive data structure holding an **abstract description of an operation** to be performed
- Activities, services, and broadcast receivers are activated through intents.
  - In broadcasts, a description of something that has happened and is being announced.



Firing of simple text intent will initiate a chooser dialog

Intents

- These **enable an application to select** an Activity based on the action you want to invoke and the data on which they operate.
- Thus **don't need a hardcoded path** to an application to use its functions and exchange data with it.
- Data can be **passed in both directions** using Intent objects, and this enables a convenient, high-level system of inter-process communication.



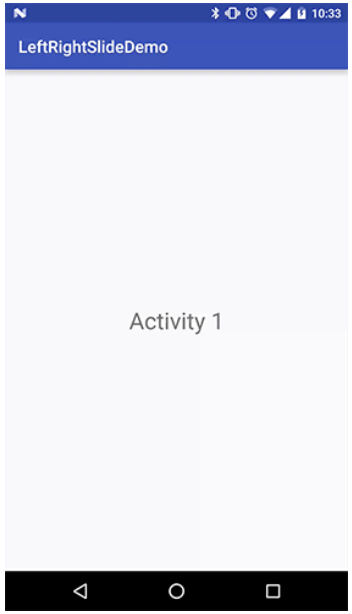
http://developer.android.com/reference/android/content/Intent.html

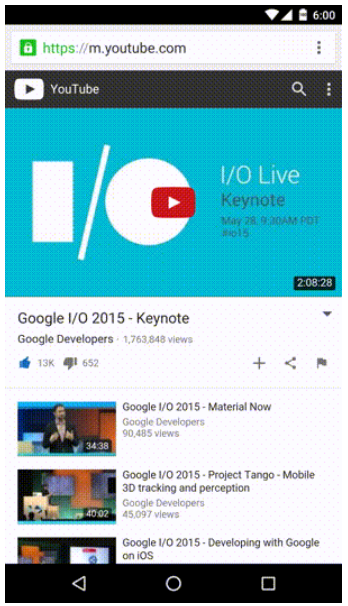
Intent Resolution

- Intents can be divided into two groups:
  1. **Explicit intents** : designate target component by its name, usually **for application internal** use e.g. launching a sister activity

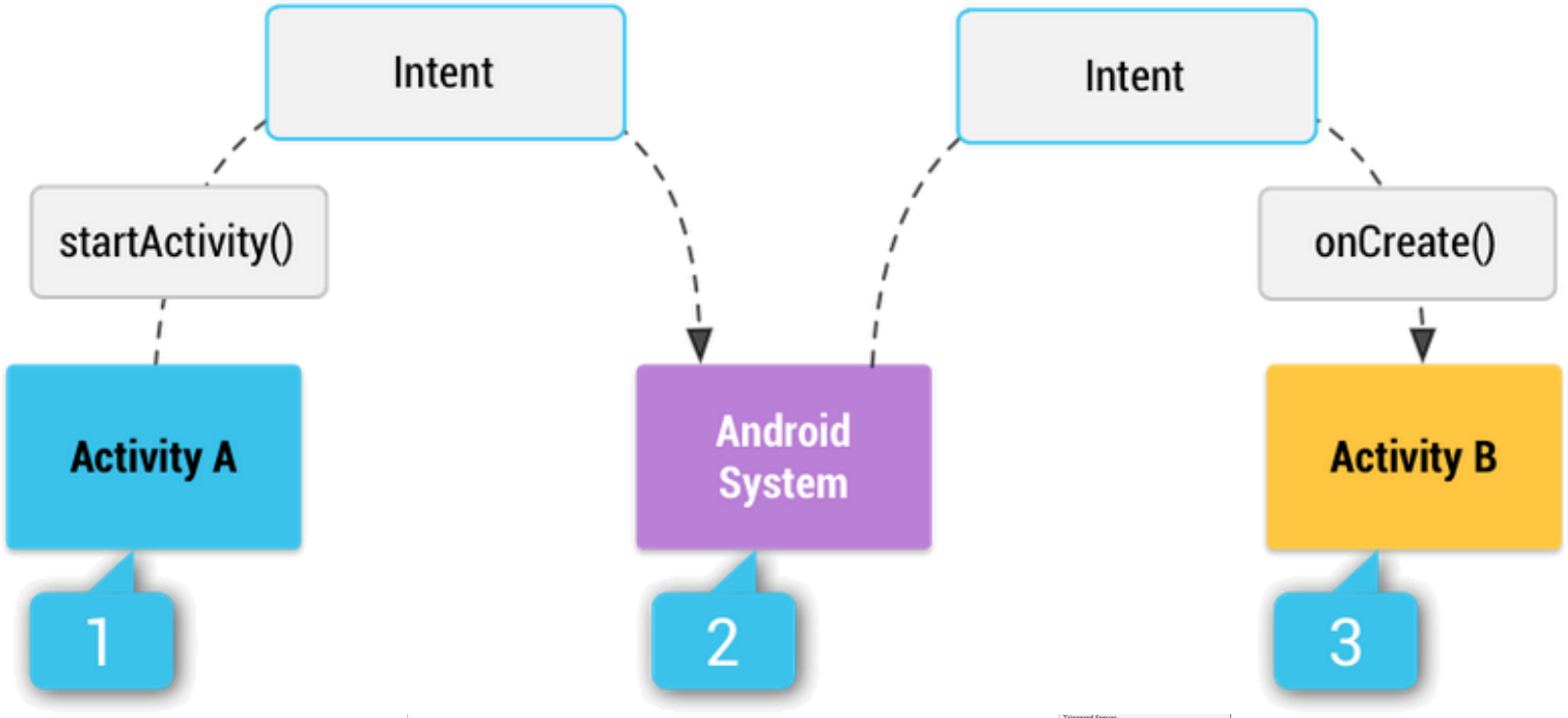
  2. **Implicit intents** : no name, used for activating components in **other applications**

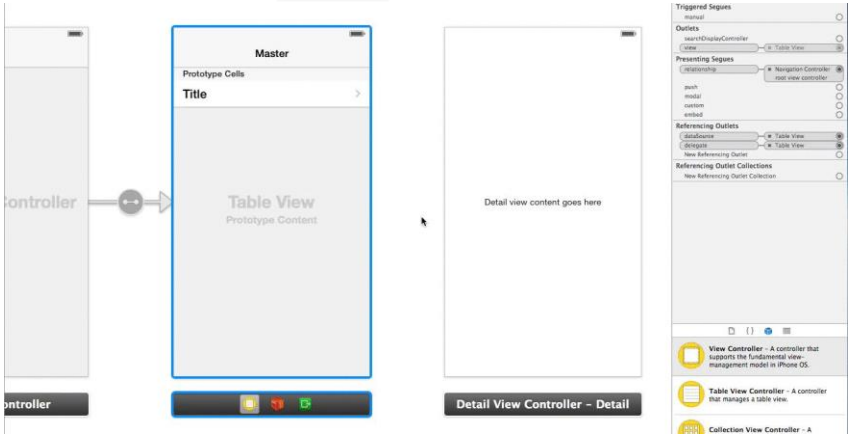
     Android system must find the best component to handle the intent Through comparing intent object to intent filter





Intent Resolution





IntentFilter

- To inform the system which implicit intents they can handle

```
…
        <activity android:name=".MainActivity">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
…
```

- IntentFilter is defined in AndroidManifest.xml
  - An intent filter is an instance of the IntentFilter class. the application's manifest file (AndroidManifest.xml) as <intent-filter> elements.

IntentFilter

- By matching intent filter through three aspects of the Intent object
  - action
  - Data (both URI and data type)
  - category.
- Target component will be identified (if more than one, the user will be given the choice)

- This mechanism provides two key benefits:
  - Activities can be **reused from other components** in the form of Intent generated by a request;
  - Activities may at any time called with the same IntentFilter, but with **a new component capable of same requirement replaced**.

Reference

- Android App Components
  https://developer.android.com/guide/components/index.html

- Intents and Intent Filter
  https://developer.android.com/guide/components/intents-filters.html