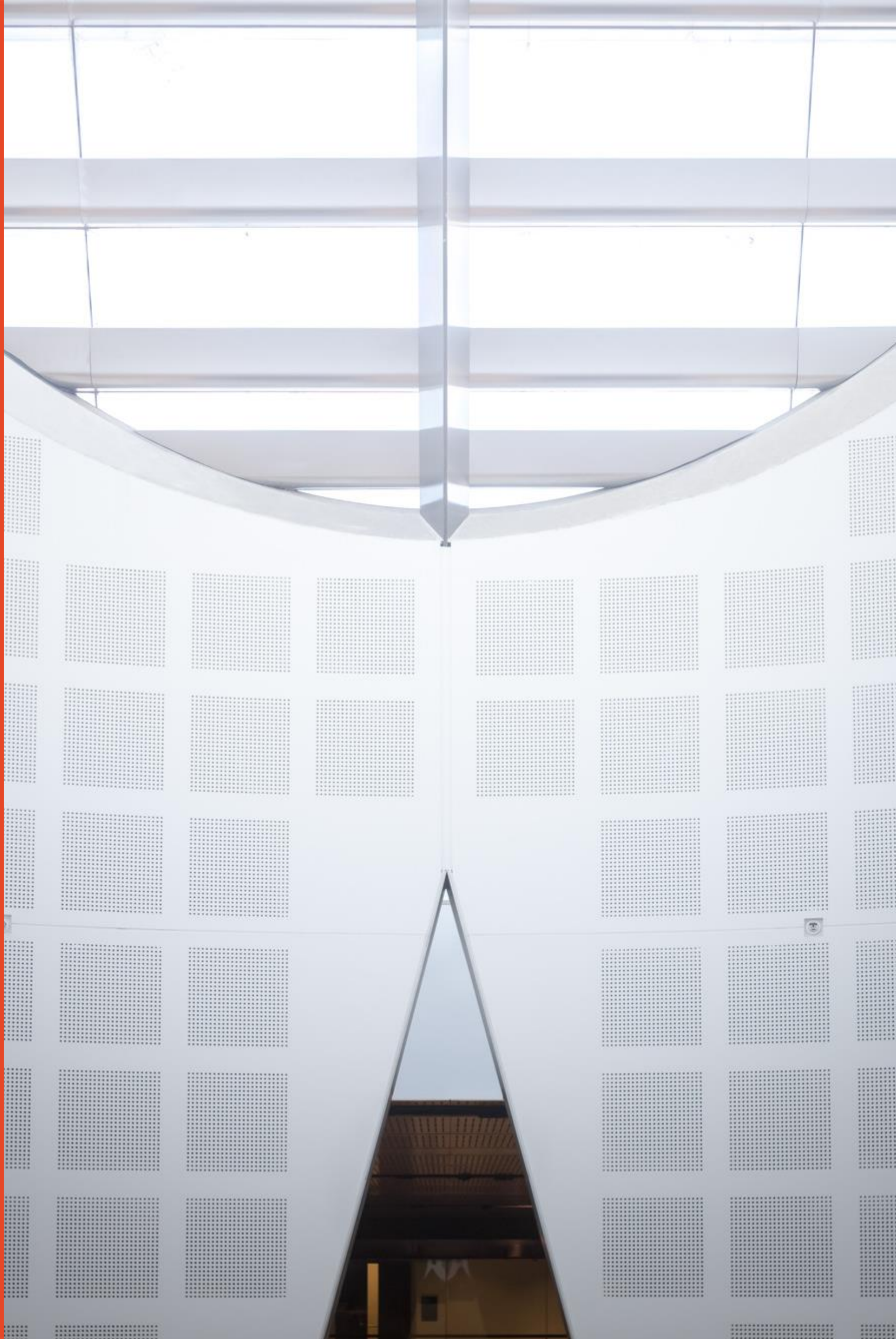
Mobile Computing
COMP5216/COMP4216

Week 03

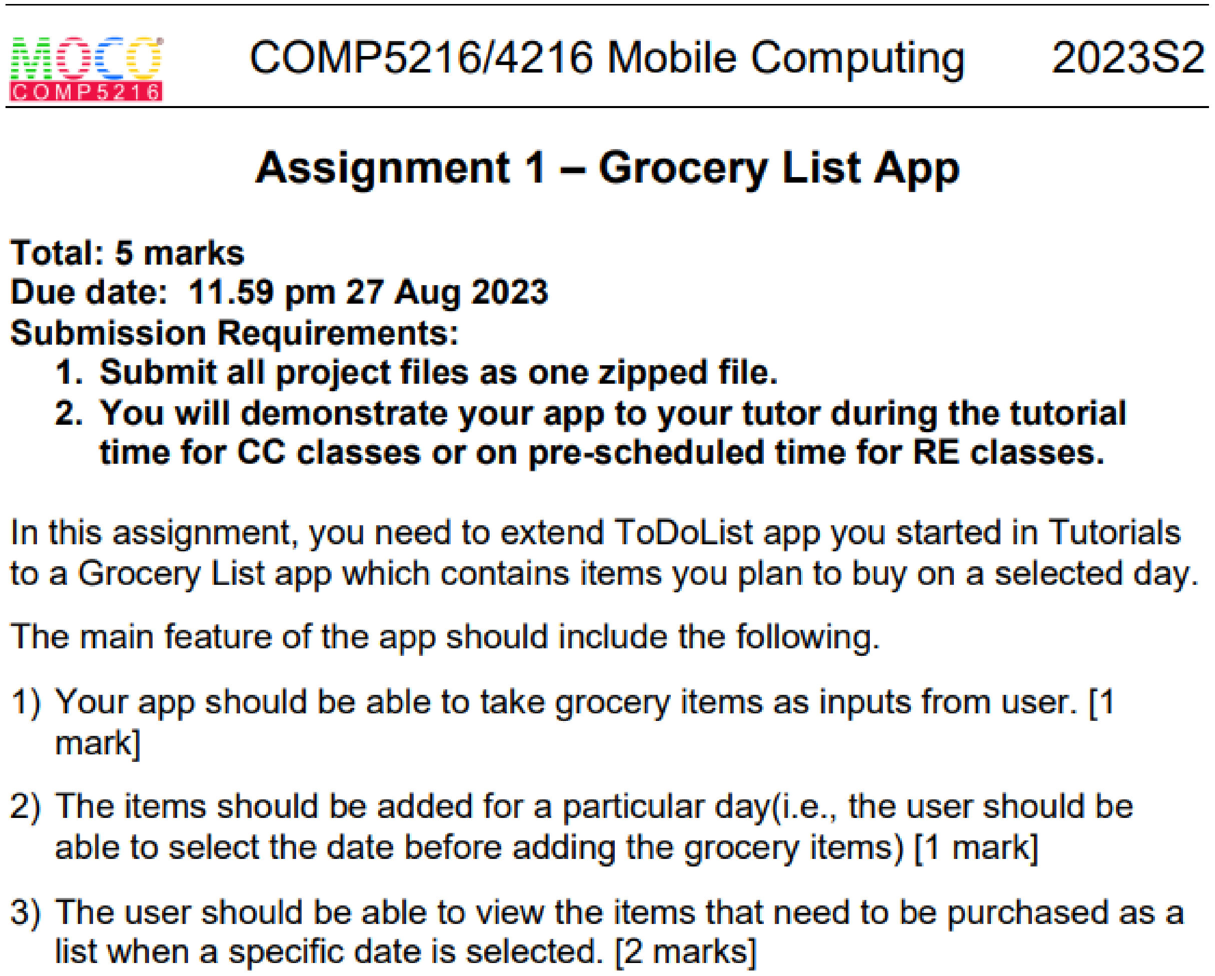
Semester 2, 2023

Dr Thilina Halloluwa
School of Computer Science



Announcements

- Assignment 1 submission.



COMP5216/4216 Mobile Computing 2023S2

Assignment 1 – Grocery List App

Total: 5 marks
Due date: 11.59 pm 27 Aug 2023
Submission Requirements:

1. **Submit all project files as one zipped file.**
2. **You will demonstrate your app to your tutor during the tutorial time for CC classes or on pre-scheduled time for RE classes.**

In this assignment, you need to extend ToDoList app you started in Tutorials to a Grocery List app which contains items you plan to buy on a selected day.

The main feature of the app should include the following.

1) Your app should be able to take grocery items as inputs from user. [1 mark]
2) The items should be added for a particular day(i.e., the user should be able to select the date before adding the grocery items) [1 mark]
3) The user should be able to view the items that need to be purchased as a list when a specific date is selected. [2 marks]

Group Project

- Refer to the Project Guidelines document on Canvas
- Two Phases – Proposal and Final
- Minimum feature set: This is essential !!!
  - Graphical user interface (GUI) to effectively interact with the user.
  - At least one form of data communication using either Cellular, WiFi, Bluetooth, etc.
  - At least one technique to save network bandwidth usage, computation resource usage and device battery usage.
  - At least one method to secure the communication and data storage, or strategy to protect user privacy in handling user data.

- **Come and test/discuss your idea with me !**

App development workflow

Six Steps

1. Define Goals
2. Analyse Requirements
3. Design Workflow: wireframe or storyboard
4. Design project structure
5. Implement codes
6. Test, debug, and release

Proposal Phase

Final Phase

Group Project submission

- Refer to the Project Guidelines document on Canvas
- Proposal Phase: Report (hard & electronic)
- Final Phase:
  - Report (hard & electronic)
  - Presentation slides
  - Video
  - Source Code
  - Presentation and Demo

Deliverables		Due Time
Proposal	Electronic submission	11.59pm, 03/09/2023 (Week 06)
Final	Electronic submission	11:59pm, 01/10/2023 (Week 11)
	Presentation & Demo	11:59pm, 08/10/2023 (Week 12)

Group enrolment

- Maximum Group size is **SEVEN,** Minimum Group Size is **FIVE**
- **Enroll to groups via Canvas.**
- Pick a group number attached to the tutorial of most number of group members.

Project Assessment – Proposal [10 marks]

- (3 marks) App: justification of the app, significance and challenge in developing the app.
- (2 marks) Solution: clear description of the workflow of the app with wireframes or UI designs for every user category of the app.
- (2 marks) Technical approach: clear description of how you plan to implement the app with technical requirements.
- (1 mark) Plan: **application specific** implementation schedule, appropriate workload distribution and collaborative development approaches.
- (1 mark) Potential setbacks: identification of **application specific** potential setbacks and solutions.
- (1 mark) Overall proposal writing.

Project Assessment – Final [30 marks]

- **15 marks will be allocated by a panel of judges evaluating all deliverables including in-class presentation and demo at Week 12 in-class presentation. These 15 marks are distributed as follows;**
  - (2 marks) Novelty and significance of the problem,
  - (4 marks) Creativity of the solution including proper presentation/demonstration of the solution.
  - (2 marks) Challenges involved in developing your app and the amount of effort that you have put in developing the final app.
  - (2 marks) Readiness to distribute the app to users.
  - (2 marks) Presentation.
  - (3 marks) Demo.

- **15 marks will be allocated by the course coordinator and tutors evaluating the following three deliverables offline.**
  - (4 marks) Source code of the app.
  - (8 marks) Final report.
  - (3 marks) Project video.

Group Project

- Register your group in Canvas.

- Discuss your idea with me.

- Sometimes I ask questions, argue, …
  - Don’t agree with me always, come with evidence !

- How do I look at your idea…
  - **As a teacher, As an Engineer/Developer, As an Investor**

Android Programming Basics - 1

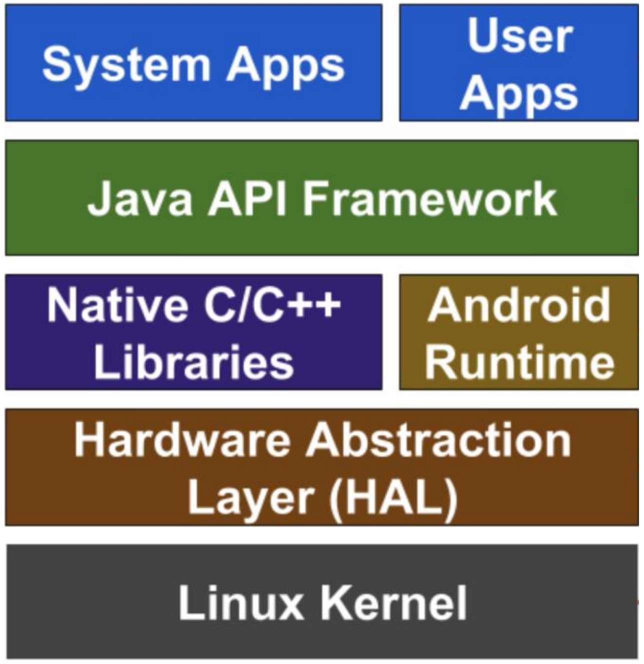
COMP5216/ COMP4216

S2, 2023

THE UNIVERSITY OF SYDNEY

What is Android?

Major components of Android Stack



- **Applications:** Users interact with the device via the apps. Can be either first party or third party.
- **Android Framework:** Provides basic functions such as communication between apps, managing voice calls or managing app life cycles.
- **Native Libraries:** C/C++ libraries that contain instructions to the device on handling different types of data. E.g. Webkit, SSL, SQLite, and OpenGL.
- **Android Runtime:** Dalvik Virtual Machine and Core Libraries.
- **Hardware Abstraction Layer (HAL):** Converts the Java API calls to system calls that is understood by the Linux kernel.
- **Linux Kernel:** Additional modifications done by Google to make it suitable for smartphones (E.g. power management). Handles all conventional operating system functions such as process management and memory management.

Building blocks of Android

App components

- **Activities**
- Services
- Broadcast Receivers
- Content Providers

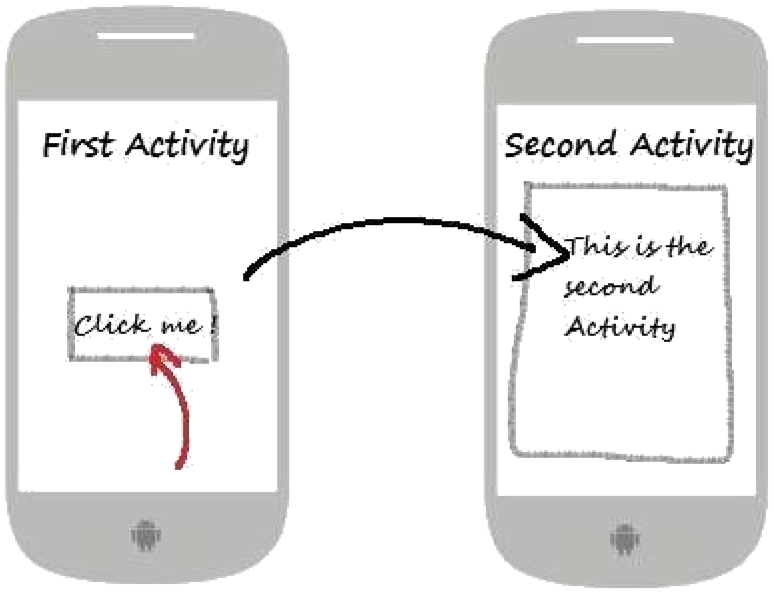
Activating components

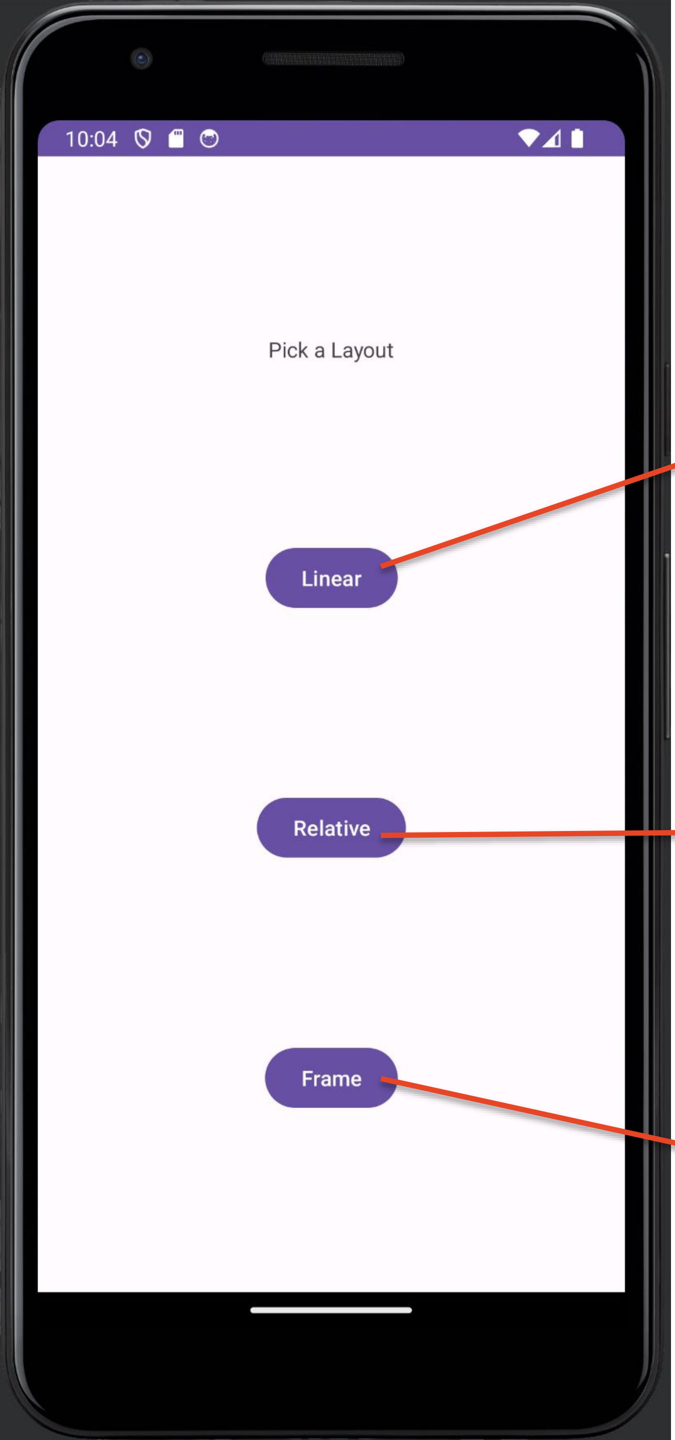
- **Intent**

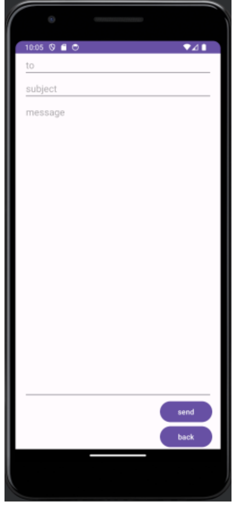
Activity

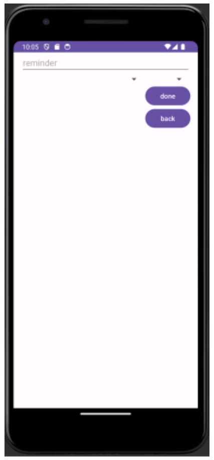
- One of the basic building block of Android
- Most common component of Android development
- Represents a single screen with a user interface
- A single app can have multiple activities.
  E.g. A game app might have different activities for login screen, scores page, and the game play screens
- Associated with a XML file that defines the arrangement of GUI components.
- https://developer.android.com/guide/components/activities/intro-activities

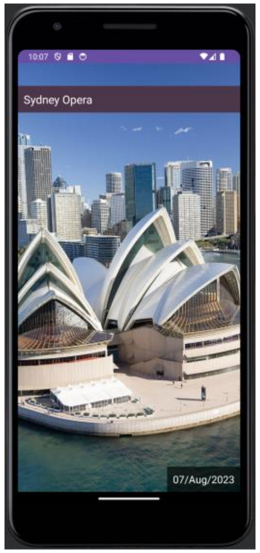
Activity


First Activity
Click me!
Second Activity
This is the second Activity


10:04
Pick a Layout
Linear
Relative
Frame


10:05
to
subject
message
send
back

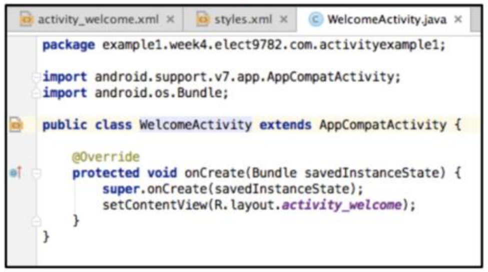

10:05
reminder
done
back


10:07
Sydney Opera
07/Aug/2023

Activity Example

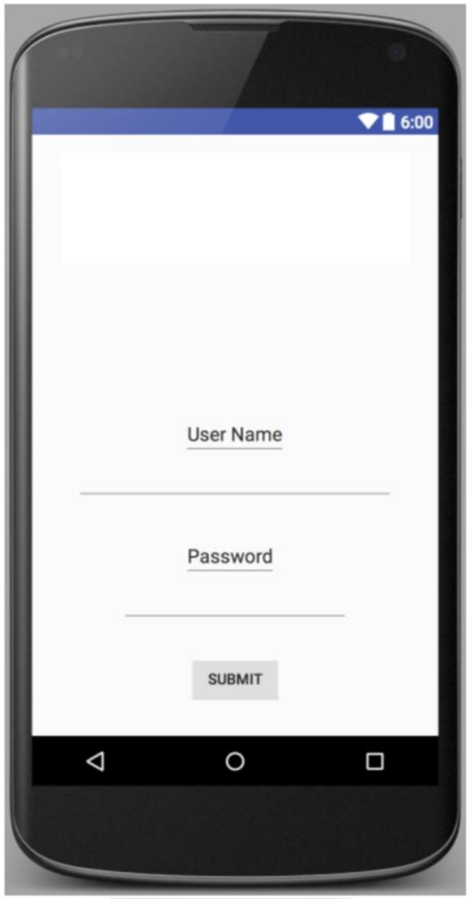
WelcomeActivity

WelcomeActivity.java



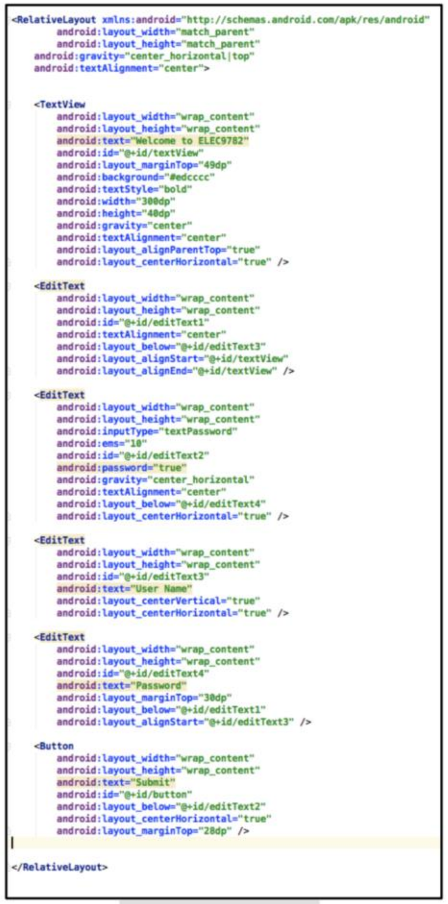
```java
package example1.week4.elect9782.com.activityexample1;

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

public class WelcomeActivity extends AppCompatActivity {

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

Activity Operation is written in Java



Design View

activity_main.xml



```xml
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
    android:gravity="center_horizontal|top"
    android:textAlignment="center">


    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Welcome to ELEC9782"
        android:id="@+id/textView"
        android:layout_marginTop="49dp"
        android:background="#edcccc"
        android:textStyle="bold"
        android:width="300dp"
        android:height="40dp"
        android:gravity="center"
        android:textAlignment="center"
        android:layout_alignParentTop="true"
        android:layout_centerHorizontal="true" />

    <EditText
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:id="@+id/editText1"
        android:textAlignment="center"
        android:layout_below="@+id/editText3"
        android:layout_alignStart="@+id/textView"
        android:layout_alignEnd="@+id/textView" />

    <EditText
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:inputType="textPassword"
        android:ems="10"
        android:id="@+id/editText2"
        android:password="true"
        android:gravity="center_horizontal"
        android:textAlignment="center"
        android:layout_below="@+id/editText4"
        android:layout_centerHorizontal="true" />

    <EditText
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:id="@+id/editText3"
        android:text="User Name"
        android:layout_centerVertical="true"
        android:layout_centerHorizontal="true" />

    <EditText
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:id="@+id/editText4"
        android:text="Password"
        android:layout_marginTop="30dp"
        android:layout_below="@+id/editText1"
        android:layout_alignStart="@+id/editText3" />

    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Submit"
        android:id="@+id/button"
        android:layout_below="@+id/editText2"
        android:layout_centerHorizontal="true"
        android:layout_marginTop="28dp" />

</RelativeLayout>
```

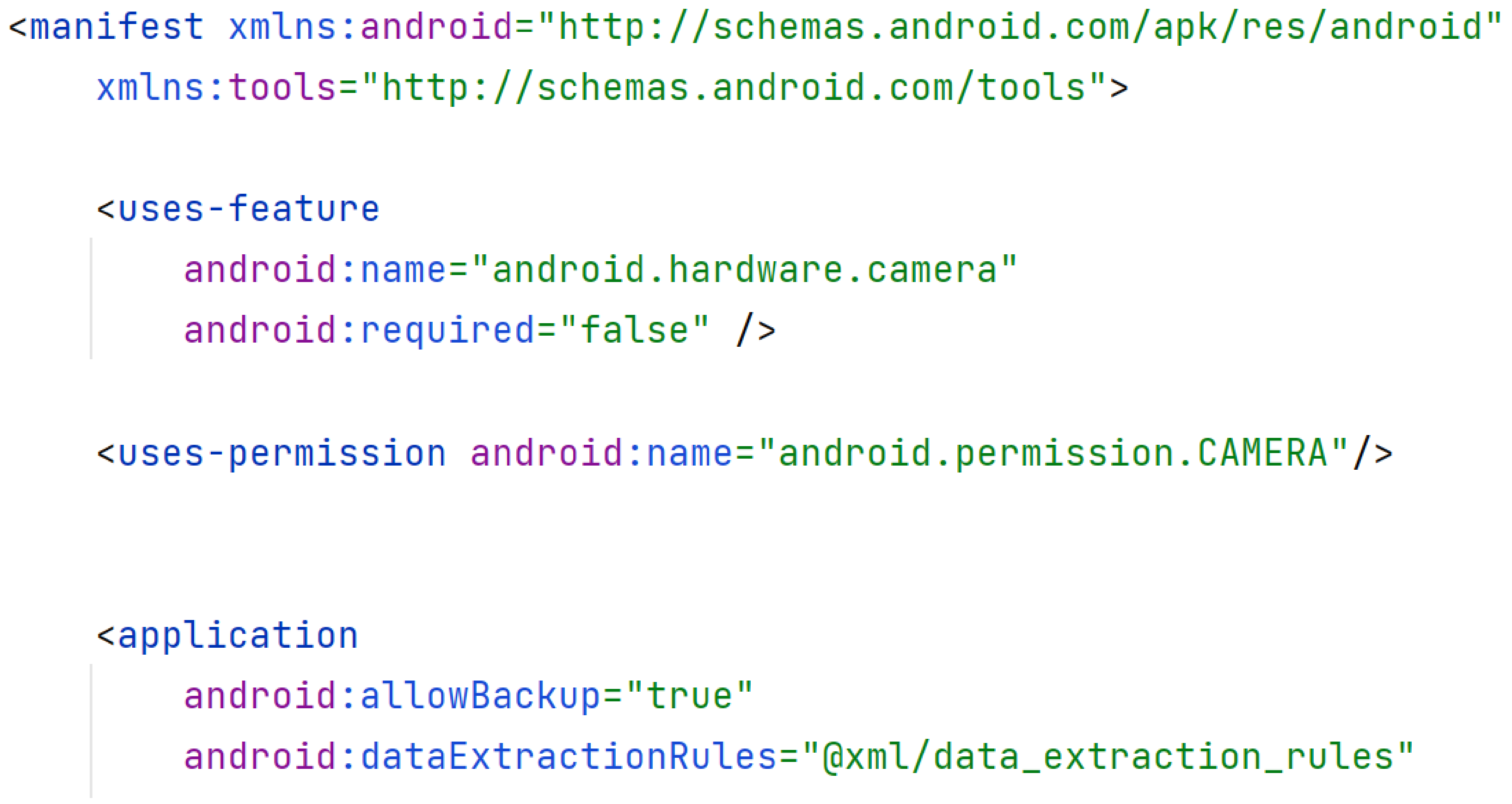
Text View

Activity Layout and other resources

Android Manifest File

- Every app project must have an AndroidManifest.xml file
  - with precisely that name, at the root of the project source set.
- The manifest file describes essential information about your app to the Android build tools, the Android operating system, and Google Play.
- The manifest file is declares;
  - The components of the app
  - The permissions that the app needs
  - The hardware and software features the app requires
  - …

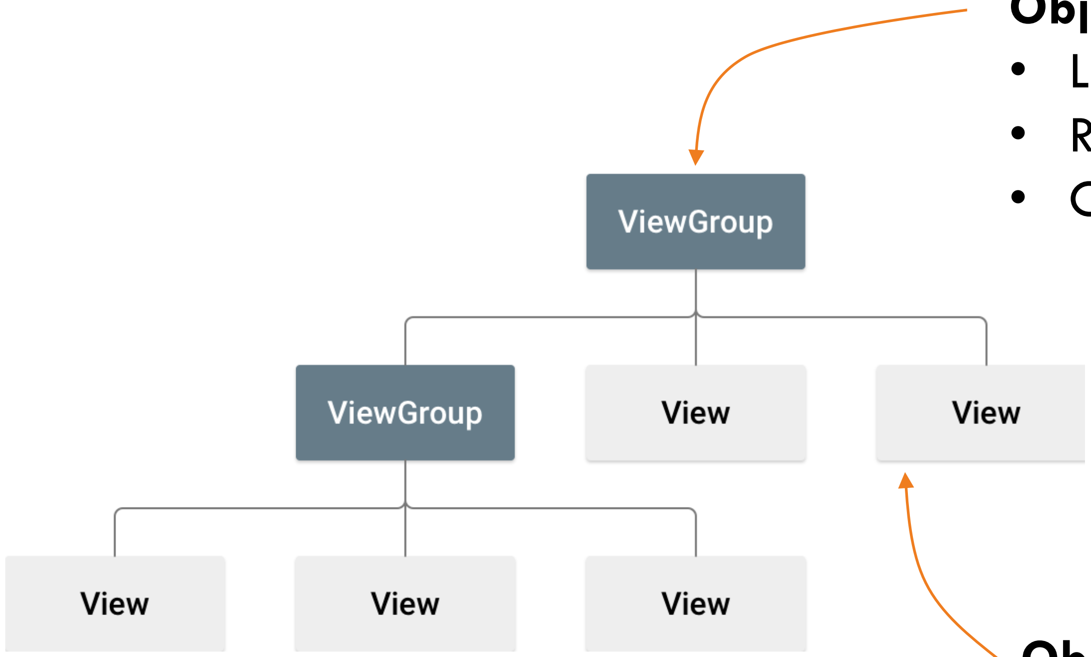
Android Manifest File- Permissions



```xml
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools">

    <uses-feature
        android:name="android.hardware.camera"
        android:required="false" />

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


    <application
        android:allowBackup="true"
        android:dataExtractionRules="@xml/data_extraction_rules"
```

Various Layouts

- Layout defines the visual structure of the GUI.
- View hierarchy



Objects - Layouts

- Linear Layout
- Relative Layout
- Constraint Layout

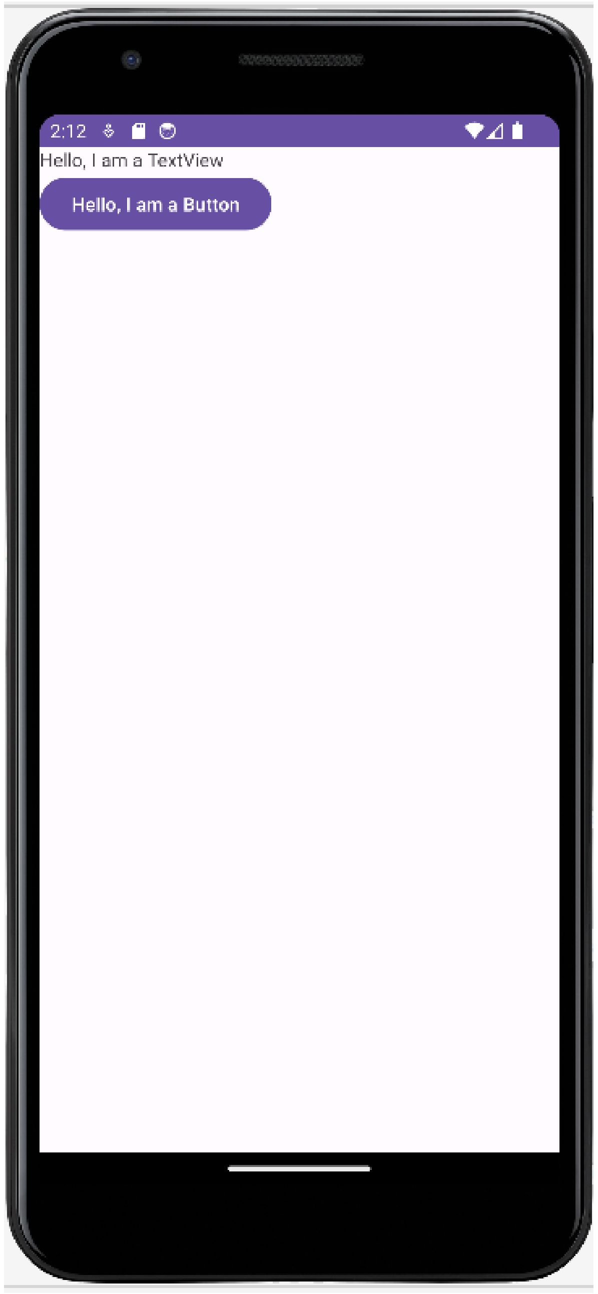
Objects - Widgets

- Buttons
- Text view

Layouts

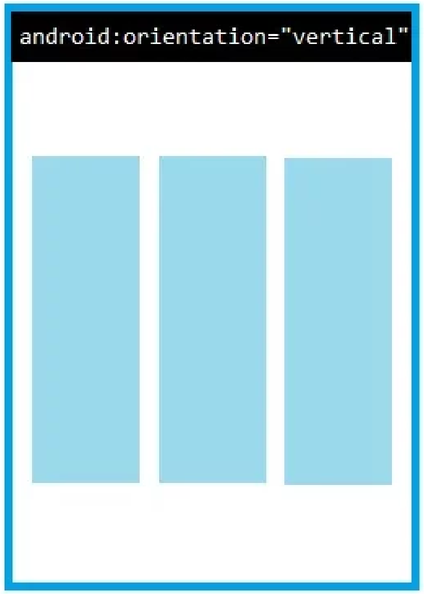
- Can declare Layouts
  - Writing the XML
  - Using Android Studio's "Layout Editor"

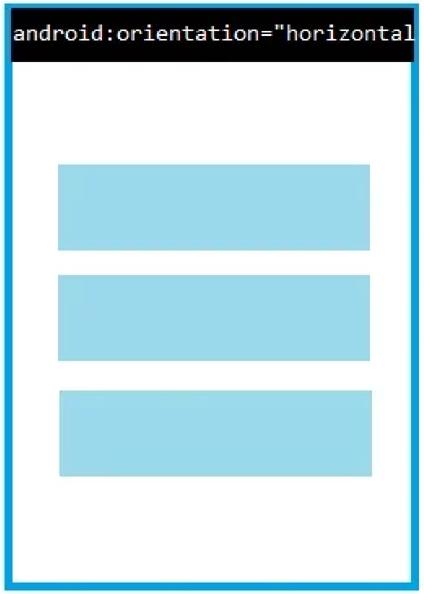
```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
              android:layout_width="match_parent"
              android:layout_height="match_parent"
              android:orientation="vertical" >
    <TextView android:id="@+id/text"
              android:layout_width="wrap_content"
              android:layout_height="wrap_content"
              android:text="Hello, I am a TextView" />
    <Button android:id="@+id/button"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Hello, I am a Button" />
</LinearLayout>
```



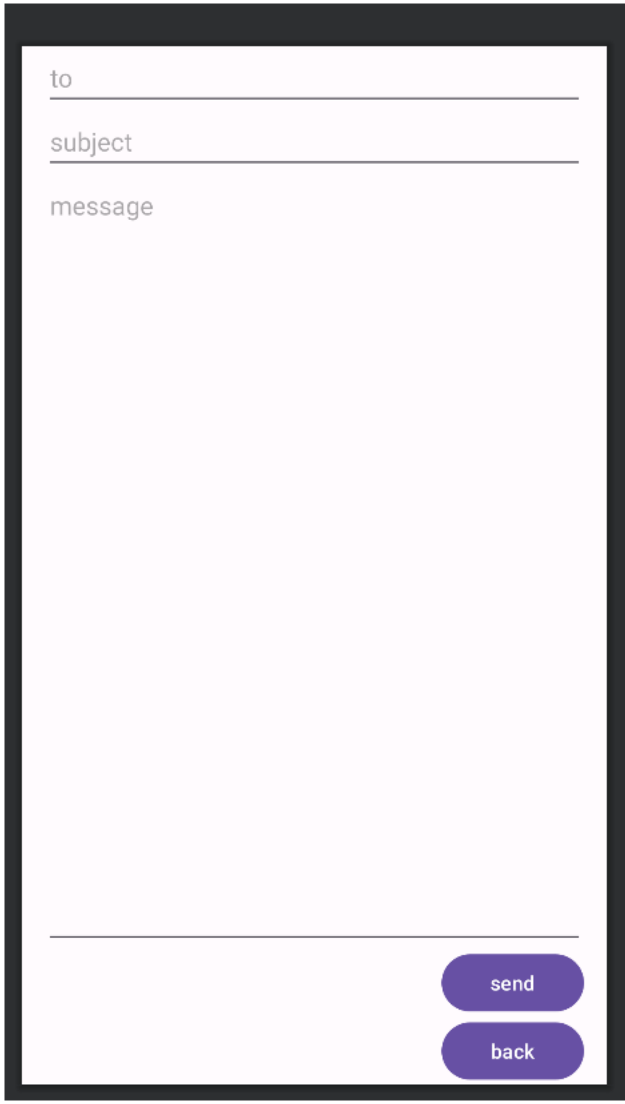
Types of UI Layouts in Android – Linear Layout

```
<LinearLayout
xmlns:android="http://schemas.android.com/apk/res/android"
android:layout_width="match_parent"
android:layout_height="match_parent"
android:orientation="horizontal">
       <!-- Include other widget or layout tags here. These are
       considered
       "child views" or "children" of the linear layout -->
</LinearLayout>
```



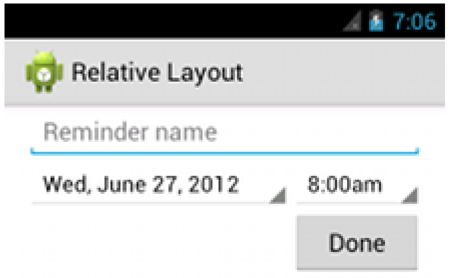


```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingLeft="16dp"
    android:paddingRight="16dp"
    android:orientation="vertical" >
    <EditText
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:hint="to" />
    <EditText
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:hint="subject" />
    <EditText
        android:layout_width="match_parent"
        android:layout_height="0dp"
        android:layout_weight="1"
        android:gravity="top"
        android:hint="message" />
    <Button
        android:layout_width="100dp"
        android:layout_height="wrap_content"
        android:layout_gravity="right"
        android:text="send" />
    <Button
        android:id="@+id/btnback"
        android:layout_width="100dp"
        android:layout_height="wrap_content"
        android:layout_gravity="right"
        android:text="back" />
</LinearLayout>
```



Types of UI Layouts in Android – Relative Layout

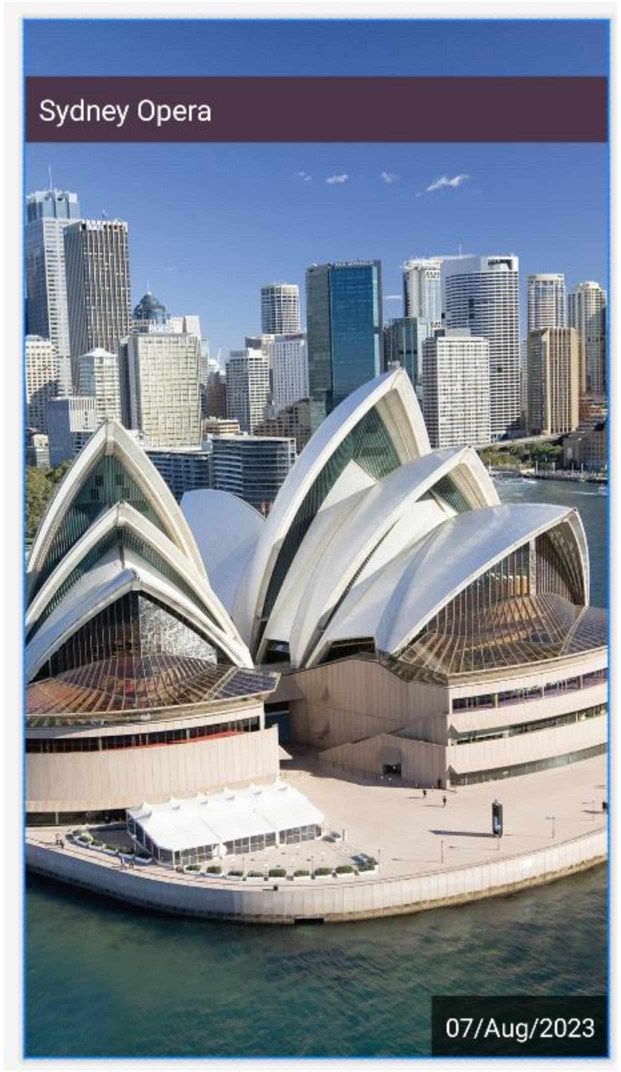
```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingLeft="16dp"
    android:paddingRight="16dp" >
    <EditText
        android:id="@+id/name"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:hint="@string/reminder" />
    <Spinner
        android:id="@+id/dates"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_below="@id/name"
        android:layout_alignParentLeft="true"
        android:layout_toLeftOf="@+id/times" />
    <Spinner
        android:id="@id/times"
        android:layout_width="96dp"
        android:layout_height="wrap_content"
        android:layout_below="@id/name"
        android:layout_alignParentRight="true" />
    <Button
        android:layout_width="96dp"
        android:layout_height="wrap_content"
        android:layout_below="@id/times"
        android:layout_alignParentRight="true"
        android:text="@string/done" />
</RelativeLayout>
```



Types of UI Layouts in Android – Frame Layout

– FrameLayout is designed to block out an area on the screen to display a single item

```
<FrameLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical">
    <ImageView
        android:id="@+id/imgvw1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:scaleType="centerCrop"
        android:src="@drawable/sydney" />
    <TextView
        android:id="@+id/txtvw1"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginTop="40dp"
        android:background="#4C374A"
        android:padding="10dp"
        android:text="Sydney Opera"
        android:textColor="#FFFFFF"
        android:textSize="20sp" />
    <TextView
        android:id="@+id/txtvw2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="right|bottom"
        android:background="#AA000000"
        android:padding="10dp"
        android:text="07/Aug/2023"
        android:textColor="#FFFFFF"
        android:textSize="18sp" />
</FrameLayout>
```



What is the difference between gravity and layout_gravity in Android? (HW)

Types of UI Layouts in Android – Constraint Layout

- ConstraintLayout permits the creation of complicated layouts with a flat view hierarchy
- Each view must have a minimum of one constraint for each axis, but often more are necessary.

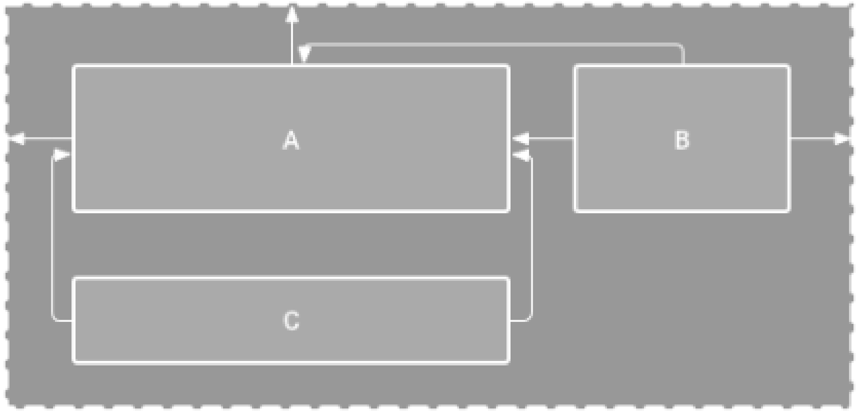


Figure 1. The editor shows view C below A, but it has no vertical constraint.

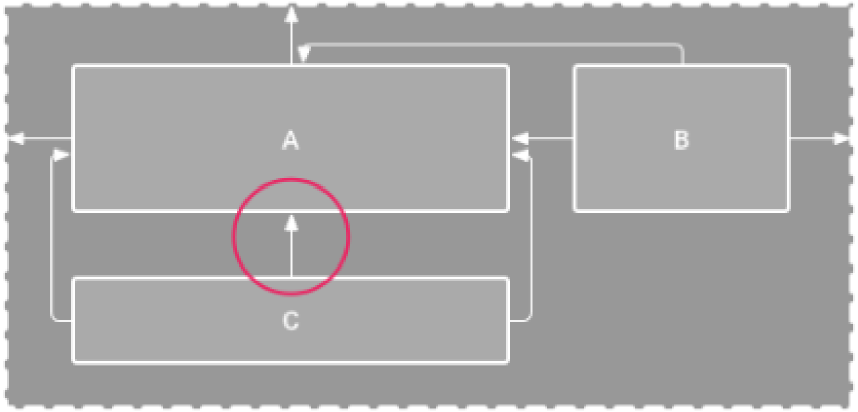
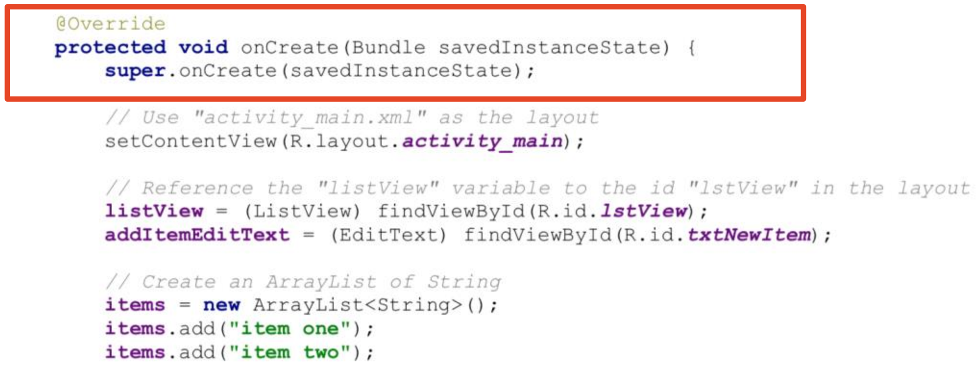


Figure 2. View C is now vertically constrained below view A.

Build a responsive UI with ConstraintLayout | Android Developers (HW)

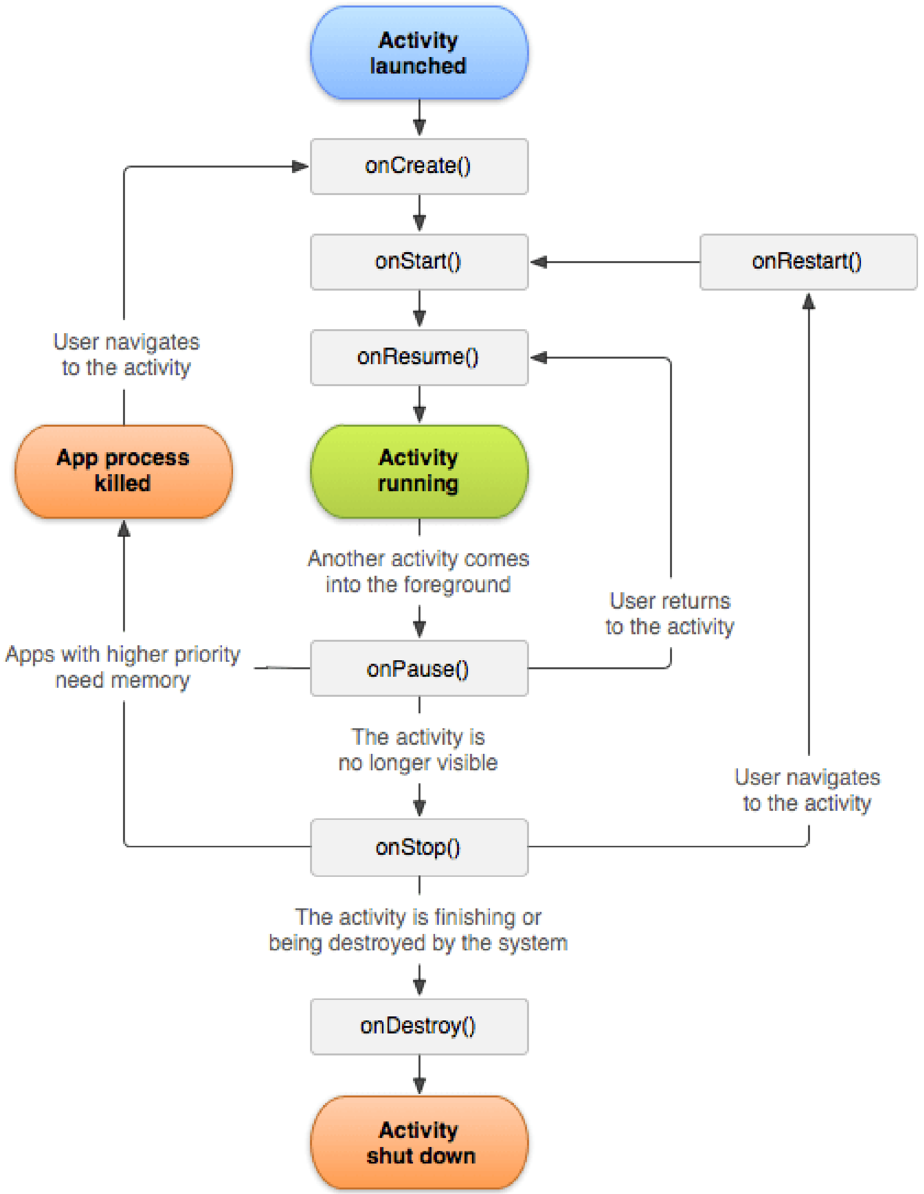
Activity Lifecycle

- You can override lifecycle methods to develop your customized activity.
  - E.g. What to do after starting the app → Override onCreate() method
  - From tutorial 1;



```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);

    // Use "activity_main.xml" as the layout
    setContentView(R.layout.activity_main);

    // Reference the "listView" variable to the id "lstView" in the layout
    listView = (ListView) findViewById(R.id.lstView);
    addItemEditText = (EditText) findViewById(R.id.txtNewItem);

    // Create an ArrayList of String
    items = new ArrayList<String>();
    items.add("item one");
    items.add("item two");
```

Activity Lifecycle


Activity launched
onCreate()
onStart()
onRestart()
onResume()
User navigates to the activity
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

Activity Lifecycle

- **onCreate()**
  - The Android activity lifecycle starts with the onCreate() method. This method is called when the user clicks on your app's icon, which causes this method to create the activity.
- **onStart()**
  - After views have been initialized and the layout has been set in the onCreate() method, the onStart() method is called. This method makes the activity visible to the user.
- **onResume()**
  - After the activity is visible to the user, the onResume() method is called when the user starts interacting with it.
- **onPause()**
  - When the user leaves the current activity, the system pauses all operations occurring on the activity and calls the onPause() method.
- **onStop()**
  - When the user presses the back button or navigates to another activity, the onStop() method is called since the activity is no longer visible to the user.
- **onDestroy()**
  - This method is called before the system destroys the activity.

Activity Lifecycle – onPause method

– Always save the user’s data on the onPause() method.

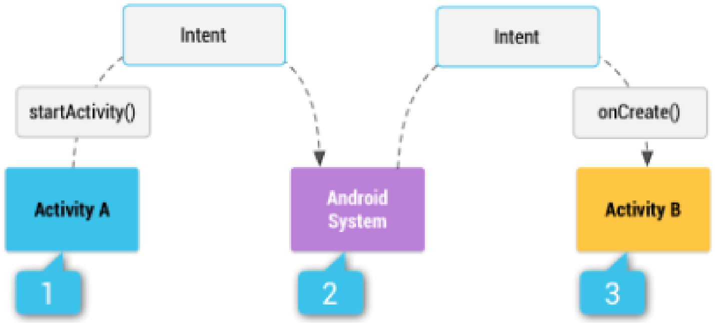
```
@Override
    protected void onPause() {
        super.onPause();

        SharedPreferences sharedPreferences = getSharedPreferences("MySharedPref
        SharedPreferences.Editor myEdit = sharedPreferences.edit();

//use the putString and putInt methods to store the users text.
        myEdit.putString("model", model.getText().toString());
        myEdit.putInt("price", Integer.parseInt(price.getText().toString()));

//save the text by invoking the apply() method
        myEdit.apply();
    }
```

– Releasing resources when the app is inactive

```
protected void onPause() {
    super.onPause();
    sensorManager.unregisterListener(this);
}
```

Intents

- Intent is a messaging object to request an action from another app component.
- Primary use-cases:
  - **To start an activity**
  - **To start a service**
  - **To deliver a broadcast**



- Intent types:
  - **Explicit Intents**: Communicate within the same application. Need to specify the exact name of the component , e.g. class name. ??
  - **Implicit Intents**: Communicate between applications. Requested by declaring the general action to perform, e.g. location. ??

- https://developer.android.com/guide/components/intents-filters

Explicit vs Implicit Intents

- **Explicit Intents**: specify which application will satisfy the intent, by supplying either the target app's package name or a fully-qualified component class name.

```
Intent intent = new Intent(FirstActivtiy.this,
SecondActivity.class);
startActivity(intent);
```

- **Implicit Intents**: do not name a specific component, but instead declare a general action to perform, which allows a component from another app to handle it.

```
Intent intent = new Intent();
intent.setAction(android.content.Intent.ACTION_VIEW);
intent.setData(Contract.Contacts.CONTENT_URL);
startActivity(intent);
```

Building bocks of an Intent

1. **Component name**
   - Name of the component to start
     - Must specify the name for *Explicit* Intent, e.g. class name of the new Activity.
     - Empty for *Implicit* Intent

2. **Action**
   - String that specifies the desired operation, e.g. view or pick
     - `ACTION_VIEW` - to show information to a user

```
Uri webpage = Uri.parse("https://www.android.com");
Intent webIntent = new Intent(Intent.ACTION_VIEW, webpage);
```

     - `ACTION_SEND` – to share data through another app e.g email, social media
     - `ACTION_DIAL` - Dial a number
     - `ACTION_EDIT` - Display data to edit
     - `ACTION_SYNC` - Synchronise device data with a server
     - `ACTION_MAIN` - Start as initial activity of the app.
     - …

Building bocks of an Intent

3. **Data**
   - Data and type of data (MIME type) associated with the Intent
   - Type of data should be related to the action
     - E.g. If the action is ACTION_DIAL, data should be the phone number.
   - Formatted as URI object (Uniform Resource Identifier)
     - `Uri.prase(“http://www.google.com”)`

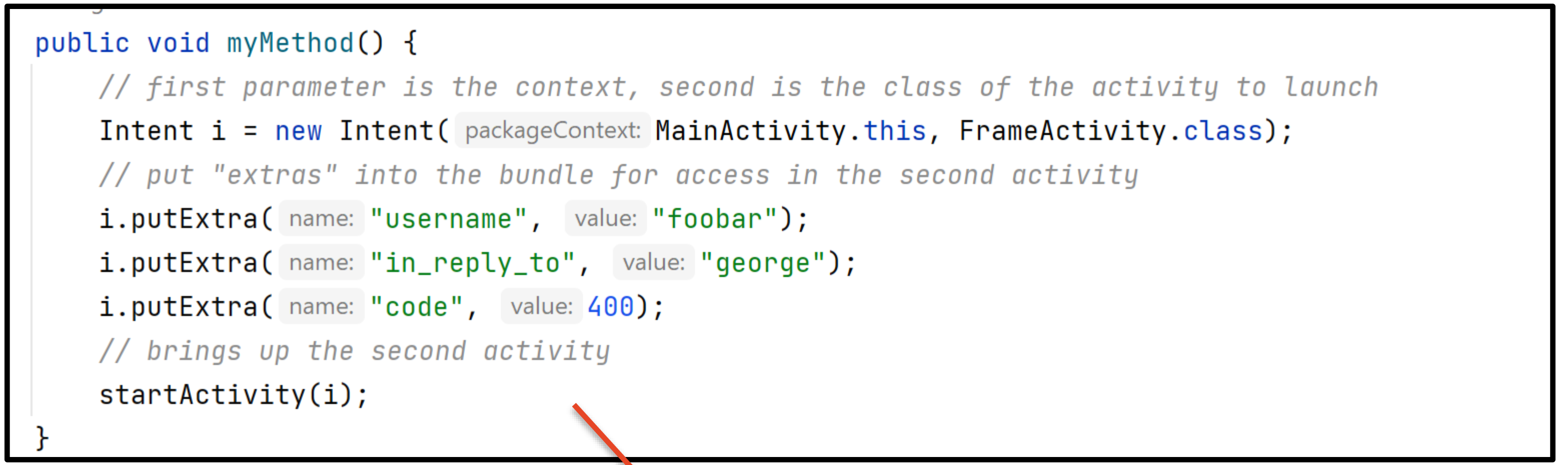
   - To set only the data URI, call setData().
   - To set only the MIME type, call setType().
   - If necessary, you can set both explicitly with setDataAndType().

Building bocks of an Intent

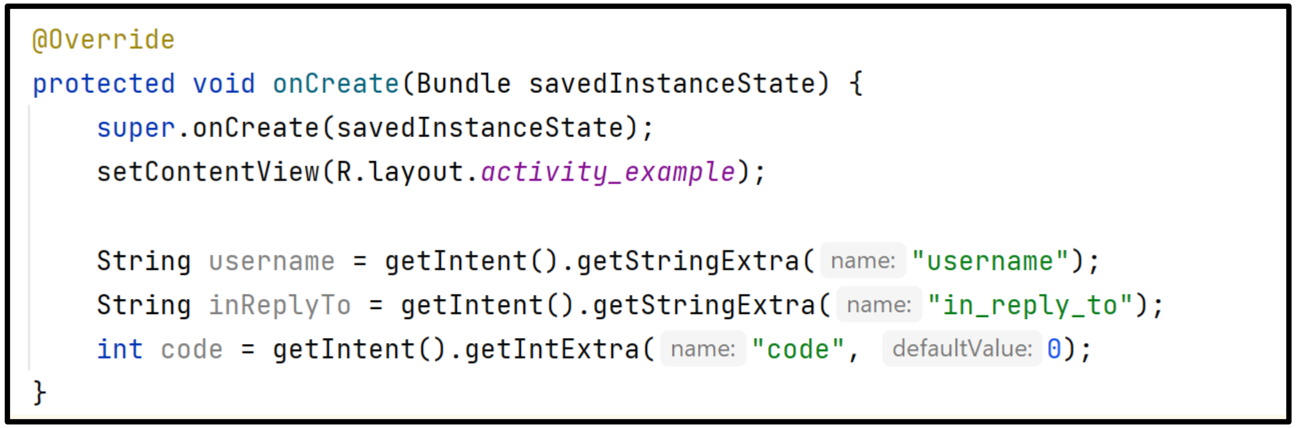
4. **Category**
   - String containing additional information about the component
     - CATEGORY_BROWSABLE – To start a web browser to display data
     - CATEGORY_LAUNCHER - The activity is the initial activity of a task and is listed in the system's application launcher.
   - Specify the category with `addCategory()`

5. **Extras**
   - Key-value pairs that carry additional information to complete the action
   - Add extra info with `putExtra()`

6. **Flags**
   - Metadata for the intent
     - E.g. How to launch the activity, how to treat it after launching, etc.
   - Can set flags using `setFalgs()`



```java
public void myMethod() {
    // first parameter is the context, second is the class of the activity to launch
    Intent i = new Intent( packageContext: MainActivity.this, FrameActivity.class);
    // put "extras" into the bundle for access in the second activity
    i.putExtra( name: "username",  value: "foobar");
    i.putExtra( name: "in_reply_to",  value: "george");
    i.putExtra( name: "code",  value: 400);
    // brings up the second activity
    startActivity(i);
}
```



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

    String username = getIntent().getStringExtra( name: "username");
    String inReplyTo = getIntent().getStringExtra( name: "in_reply_to");
    int code = getIntent().getIntExtra( name: "code",  defaultValue: 0);
}
```

Example

- Start another activity using an Intent
- Example: Tutorial 2
  - What type of an Intent is used ?

```
@Override
public void onItemClick(AdapterView<?> parent, View view, int position, long id) {
    String updateItem = (String) itemsAdapter.getItem(position);
    Log.i("MainActivity", "Clicked item " + position + ": " + updateItem);

    Intent intent = new Intent(MainActivity.this, EditToDoItemActivity.class);
    if (intent != null) {
        // put "extras" into the bundle for access in the edit activity
        intent.putExtra("item", updateItem);
        intent.putExtra("position", position);
        // brings up the second activity
        startActivityForResult(intent, EDIT_ITEM_REQUEST_CODE);
        itemsAdapter.notifyDataSetChanged();
    }
}
```

Example

- Start another activity using an Intent
- Example: Tutorial 2
  - What type of an Intent is used ?

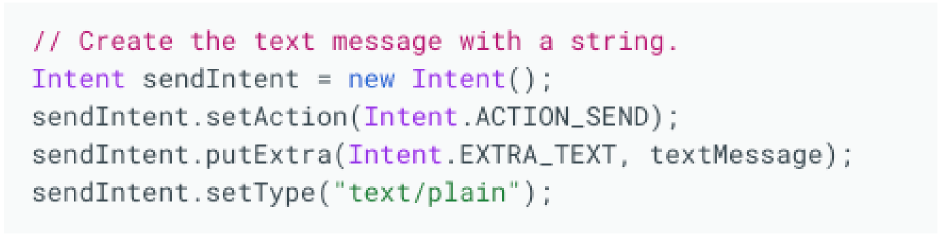
```java
@Override
public void onItemClick(AdapterView<?> parent, View view, int position, long id) {
    String updateItem = (String) itemsAdapter.getItem(position);
    Log.i("MainActivity", "Clicked item " + position + ": " + updateItem);

    Intent intent = new Intent(MainActivity.this, EditToDoItemActivity.class);
    if (intent != null) {
        // put "extras" into the bundle for access in the edit activity
        intent.putExtra("item", updateItem);
        intent.putExtra("position", position);
        // brings up the second activity
        startActivityForResult(intent, EDIT_ITEM_REQUEST_CODE);
        itemsAdapter.notifyDataSetChanged();
    }
}
```

Key-value pairs carrying additional information

Explicitly mention second activity name – Explicit Intent

Example 2

- Communicate between apps.
- By declaring the general action to perform. In this case,
  - Action: `ACTION_SEND`
  - Extra: Content to share with other people



```java
// Create the text message with a string.
Intent sendIntent = new Intent();
sendIntent.setAction(Intent.ACTION_SEND);
sendIntent.putExtra(Intent.EXTRA_TEXT, textMessage);
sendIntent.setType("text/plain");
```

- What type of intent is this?

Implicit Intent

- Communicate between apps.
- By declaring the general action to perform. In this case,
  - Action: ACTION_SEND
  - Extra: Content to share with other people

```
// Create the text message with a string.
Intent sendIntent = new Intent();
sendIntent.setAction(Intent.ACTION_SEND);
sendIntent.putExtra(Intent.EXTRA_TEXT, textMessage);
sendIntent.setType("text/plain");
```

- What type of intent is this?
- What can go wrong with the code above code block?
  - If no other apps can handle the intent, you should catch the ActivityNotFoundException to avoid crashing your app

```
// Try to invoke the intent.
try {
    startActivity(sendIntent);
} catch (ActivityNotFoundException e) {
    // Define what your app should do if no activity can handle the intent.
}
```

Intent Filters

- Declare which Intents that your app can receive with **intent-filter** element in your **AndroidManifest.xml**
- This is how Android pass Implicit Intents to relevant apps
- Define `<action/>, <data/>` and `<category/>`
- E.g.

```
<activity android:name="ShareActivity">
    <intent-filter>
        <action android:name="android.intent.action.SEND"/>
        <category android:name="android.intent.category.DEFAULT"/>
        <data android:mimeType="text/plain"/>
    </intent-filter>
</activity>
```

Intent Filters Example

- Assume you're developing a music player app
  - Your app might have an activity to play music
  - You want your app to be able to respond to certain intents, like when a user selects a music file in a file explorer app and chooses to play it using your music player app.

```
<activity android:name=".MusicPlayerActivity">
    <intent-filter>
        <action android:name="android.intent.action.VIEW" />
        <category android:name="android.intent.category.DEFAULT" /
        <data android:mimeType="audio/*" />
    </intent-filter>
</activity>
```

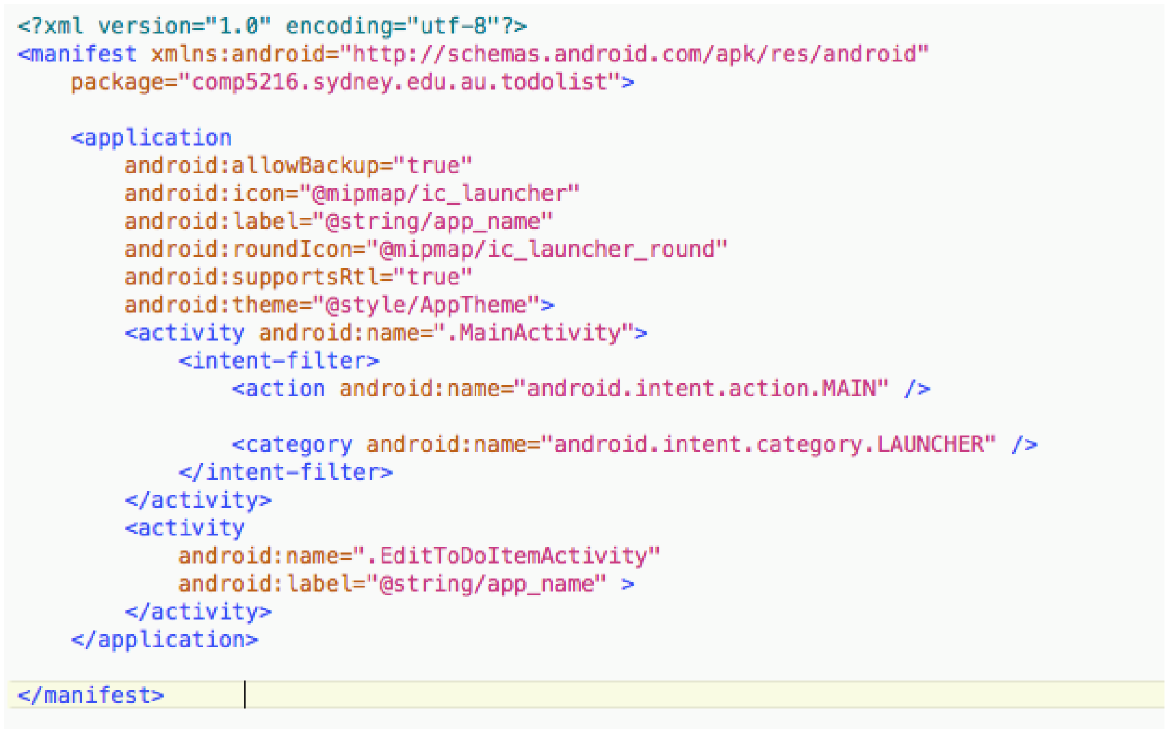
This Intent Filter specifies that the activity can respond to the "ACTION_VIEW" intent action when the data type is "audio/*

Intent Filters

- Who had a look at the AndroidManifest.xml files of Tutorial 1?
- Were there any Intent filter?

Intent Filters

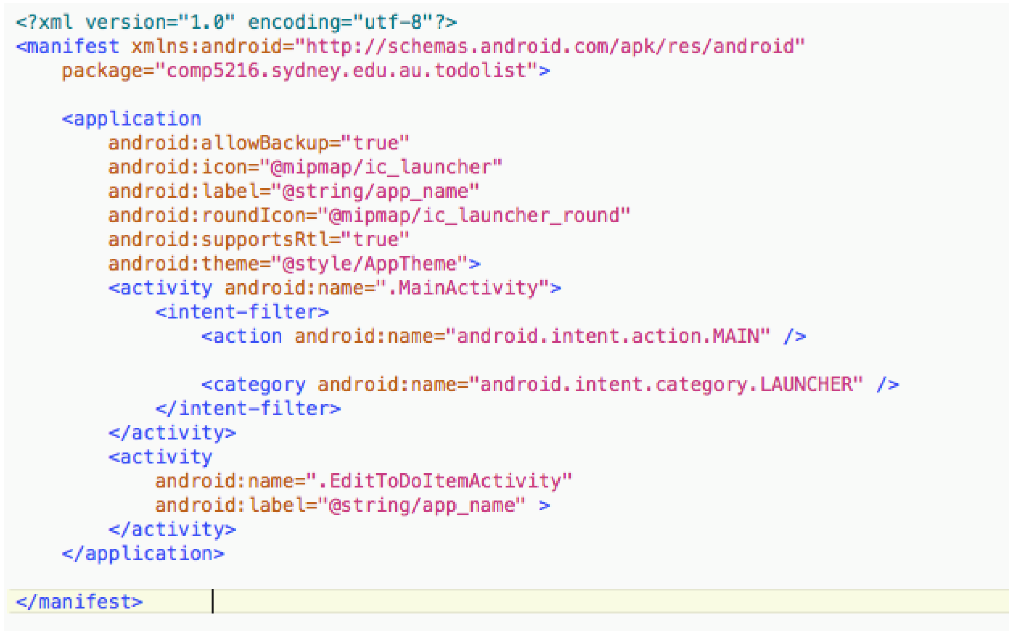
- We were not planning to receive any Intents. We still have **default filters !**
  - E.g. Tutorial 1 – AndroidManifest.xml



```xml
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="comp5216.sydney.edu.au.todolist">

    <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/AppTheme">
        <activity android:name=".MainActivity">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
        <activity
            android:name=".EditToDoItemActivity"
            android:label="@string/app_name" >
        </activity>
    </application>

</manifest>
```

Intent Filters

- `ACTION_MAIN` indicates this activity is the main entry point when the user launch the app and does not expect any intent data.
- `CATEGORY_LAUNCHER` indicates that activity's icon should be placed in the system's app launcher.



```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="comp5216.sydney.edu.au.todolist">

    <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/AppTheme">
        <activity android:name=".MainActivity">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
        <activity
            android:name=".EditToDoItemActivity"
            android:label="@string/app_name" >
        </activity>
    </application>

</manifest>
```

Next Week

- Capabilities of modern smartphones
  - Sensors
  - Audio
  - Connectivity
  - Camera
- Android Basics 2
  - Broadcast Receiver
  - Content Provider
  - Services

https://www.geckoandfly.com/13143/50-things-smartphone-replaced-will-replace-future/