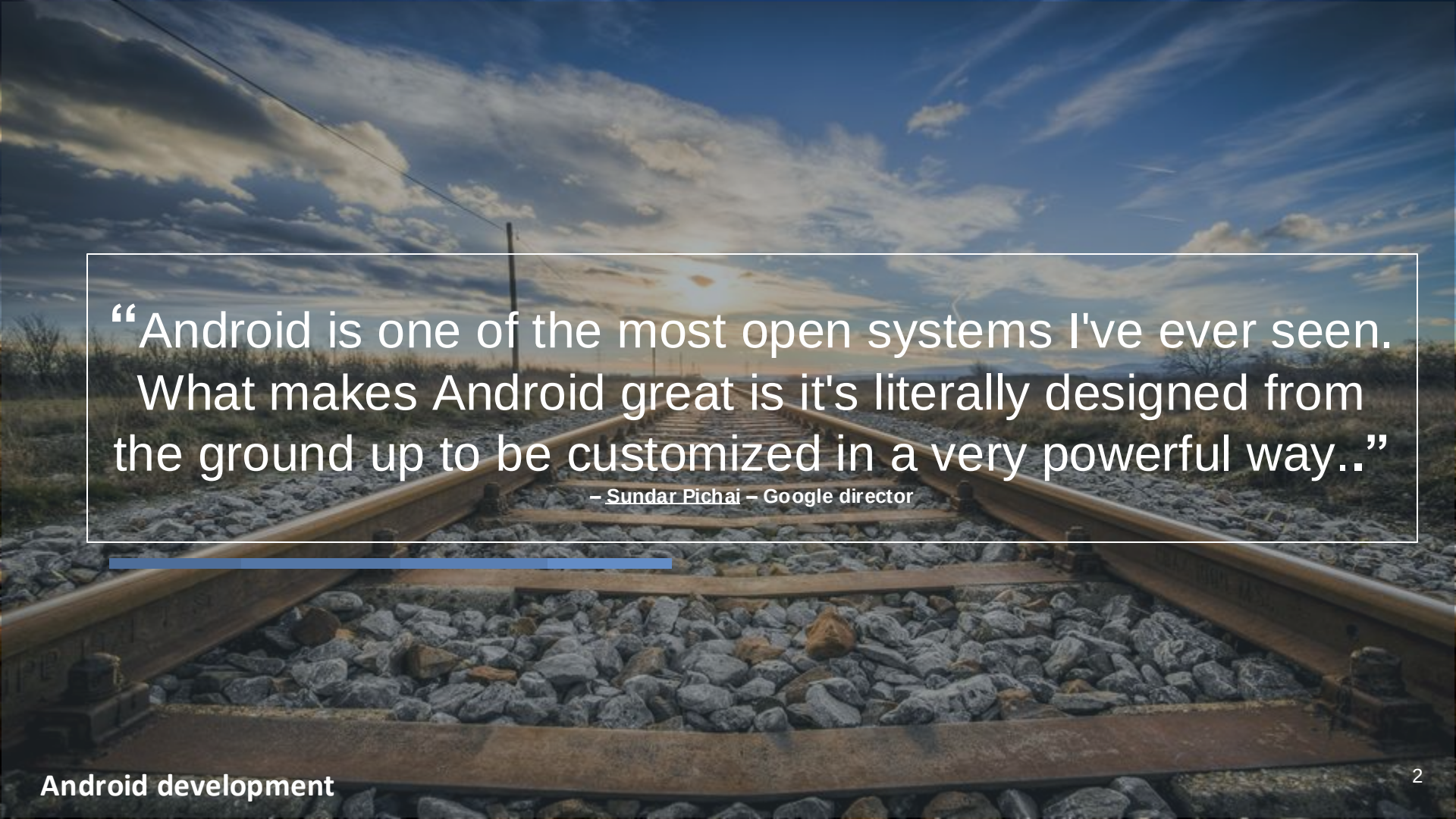
Mobile Development

(Lab 2 – GUI)

Instructor: Tran Vinh Khiem

March 1st, 2022

S³T
Smart Software System Team



“Android is one of the most open systems I've ever seen. What makes Android great is it's literally designed from the ground up to be customized in a very powerful way..”

– Sundar Pichai – Google director

Learning Objectives

- Provide you with realistic, hands-on experience developing Android applications.
- Create a portfolio of apps that you can show your friends, discuss in interviews, and borrow for other applications.

GUI widgets

9:26:00 pm Analog/DigitalClock	Button 1 Button 2 Button 3 Button	Plain Serif Bold Italic Checkbox	S M T W T F S 12:15 PM Date/TimePicker
EditText 1 (206)555-1212 •••••••••••• EditText	Gallery	ImageView/Button	ProgressBar
Plain Serif Bold Bold & Italic RadioButton	Spinner Spinner	Plain Serif Bold Italic TextView	Google MapView, WebView

Button

A clickable widget with a text label

- key attributes:

`android:clickable="bool"`	set to false to disable the button
`android:id="@+id/theID"`	unique ID for use in Java code
`android:onClick="function"`	function to call in activity when clicked (must be public, void, and take a View arg)
`android:text="text"`	text to put in the button

- represented by `Button` class in Java code

```
Button b = (Button) findViewById(R.id.theID);
...
```

Image Button

A clickable widget with an image label

- key attributes:

`android:clickable="`***`bool`***`"`	set to false to disable the button
`android:id="@+id/`***`theID`***`"`	unique ID for use in Java code
`android:onClick="`***`function`***`"`	function to call in activity when clicked (must be public, void, and take a View arg)
`android:src="@drawable/`***`img`***`"`	image to put in the button (must correspond to an image resource)

- to set up an image resource:
  - put image file in project folder **app/src/main/res/drawable**
  - use `@drawable/foo` to refer to `foo.png`
    - use simple file names with only letters and numbers

Image View

Displays an image without being clickable

- key attributes:

`android:id="@+id/`***`theID`***`"`	unique ID for use in Java code
`android:src="@drawable/`***`img`***`"`	image to put in the screen (must correspond to an image resource)

- to change the visible image, in Java code:
  - get the ImageView using findViewById
  - call its **setImageResource** method and pass R.drawable.***filename***

EditText

An editable text input box

- key attributes:

`android:hint="`***`text`***`"`	gray text to show before user starts to type
`android:id="@+id/`***`theID`***`"`	unique ID for use in Java code
`android:inputType="`***`type`***`"`	what kind of input is being typed; `number,phone,date,time,...`
`android:lines="`***`int`***`"`	number of visible lines (rows) of input
`android:maxLines="`***`int`***`"`	max lines to allow user to type in the box
`android:text="`***`text`***`"`	initial text to put in box (default empty)
`android:textSize="`***`size`***`"`	size of font to use (e.g. "20dp")

  - others: `capitalize, digits, fontFamily, letterSpacing, lineSpacingExtra, minLines, numeric, password, phoneNumber, singleLine, textAllCaps, textColor, typeface`

CheckBox

An individual toggleable on/off switch

- key attributes:

`android:checked="`***`bool`***`"`	set to true to make it initially checked
`android:clickable="`***`bool`***`"`	set to false to disable the checkbox
`android:id="@+id/`***`theID`***`"`	unique ID for use in Java code
`android:onClick="`***`function`***`"`	function to call in activity when clicked (must be public, void, and take a View arg)
`android:text="`***`text`***`"`	text to put next to the checkbox

- In Java code:

```
CheckBox cb = (CheckBox) findViewById(R.id.theID);
cb.toggle();
cb.setChecked(true);
cb.performClick();
```

RadioButton

A toggleable on/off switch; part of a group

- key attributes:

`android:checked="`***`bool`***`"`	set to true to make it initially checked
`android:clickable="`***`bool`***`"`	set to false to disable the button
`android:id="@+id/`***`theID`***`"`	unique ID for use in Java code
`android:onClick="`***`function`***`"`	function to call in activity when clicked (must be public, void, and take a View arg)
`android:text="`***`text`***`"`	text to put next to the button

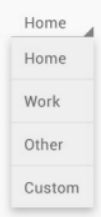
- need to be nested inside a `RadioGroup` tag in XML so that only one can be selected at a time

Radio Group

```
<LinearLayout ...
        android:orientation="vertical"
        android:gravity="center|top">
    <RadioGroup ...
            android:orientation="horizontal">
        <RadioButton ... android:id="@+id/lions"
                         android:text="Lions"
                         android:onClick="radioClick" />
        <RadioButton ... android:id="@+id/tigers"
                         android:text="Tigers"
                         android:checked="true"
                         android:onClick="radioClick" />
        <RadioButton ... android:id="@+id/bears"
                         android:text="Bears, oh my!"
                         android:onClick="radioClick" />
    </RadioGroup>
</LinearLayout>
```

Spinner

A drop-down menu of selectable choices



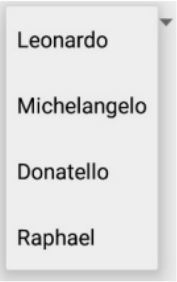
- key attributes:

`android:clickable="`***`bool`***`"`	set to false to disable the spinner
`android:id="@+id/`***`theID`***`"`	unique ID for use in Java code
`android:entries="@array/`***`array`***`"`	set of options to appear in spinner (must match an array in `strings.xml`)
`android:prompt="@string/`***`text`***`"`	title text when dialog of choices pops up

- also need to handle events in Java code (see later)
  - must get the Spinner object using findViewById
  - then call its setOnItemSelectedListener method (see example)

Spinner XML

```
<LinearLayout ...>
    <Spinner ... android:id="@+id/tmnt"
        android:entries="@array/turtles"
        android:prompt="@string/choose_turtle" />
    <TextView ... android:id="@+id/result" />
</LinearLayout>
```

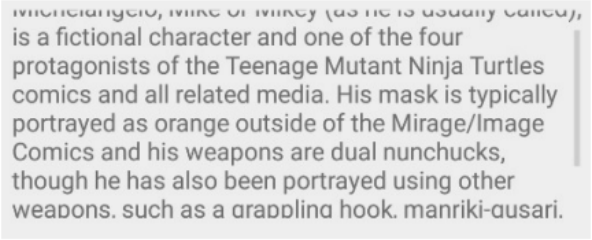


- in res/values/strings.xml:

```
<resources>
    <string name="choose_turtle">Choose a turtle:</string>
    <string-array name="turtles">
        <item>Leonardo</item>
        <item>Michelangelo</item>
        <item>Donatello</item>
        <item>Raphael</item>
    </string-array>
</resources>
```

ScrollView

A container with scrollbars around another widget or container

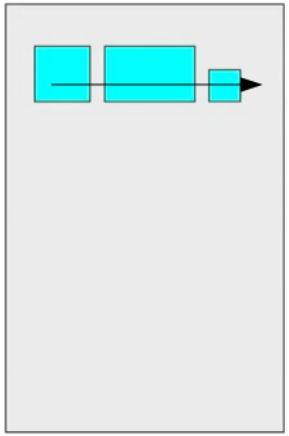



```
<LinearLayout ...>
    ...
    <ScrollView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content">
        <TextView ... android:id="@+id/turtle_info" />
    </ScrollView>
</LinearLayout>
```

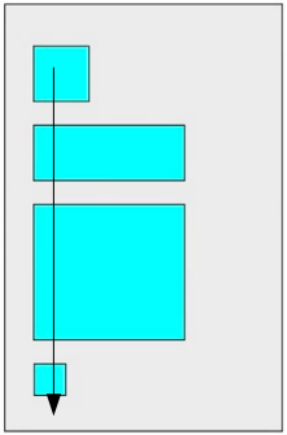
Linear Layout

- lays out widgets/views in a single line
- **orientation** of horizontal (default) or vertical
- items do *not* wrap if they reach edge of screen!

horizontal



vertical



Linear Layout Example

```
<LinearLayout ...
        android:orientation="horizontal"
        tools:context=".MainActivity">
    <Button ... android:text="Button 1" />
    <Button ... android:text="Button 2 Hooray" />
    <Button ... android:text="Button 3" />
    <Button ... android:text="Button 4
                        Very Long Text" />
</LinearLayout>
```

- In our examples, we'll use `...` when omitting boilerplate code that is auto-generated by Android Studio and not relevant to the specific example at hand.

Linear Layout Example

```
<LinearLayout ...
        android:orientation="vertical"
        tools:context=".MainActivity">
    <Button ... android:text="Button 1" />
    <Button ... android:text="Button 2
                                Hooray" />
    <Button ... android:text="Button 3" />
    <Button ... android:text="Button 4
                        Very Long Text" />
</LinearLayout>
```

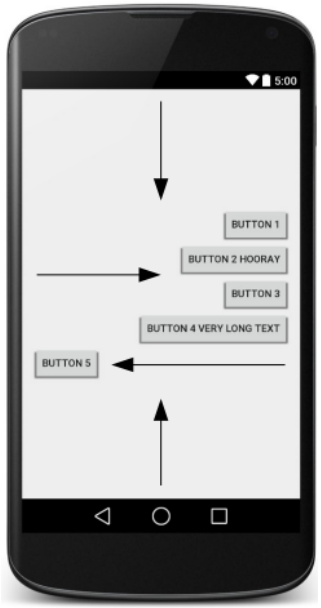
Linear Layout Example

```
<LinearLayout ...
        android:orientation="horizontal"
        tools:context=".MainActivity">
    <Button ... android:text="Button 1" />
    <Button ... android:text="Button 2 Hooray" />
    <Button ... android:text="Button 3" />
    <Button ... android:text="Button 4
                        Very Long Text" />
</LinearLayout>
```

- In our examples, we'll use `...` when omitting boilerplate code that is auto-generated by Android Studio and not relevant to the specific example at hand.

Gravity

- **gravity**: alignment direction that widgets are pulled
  - top, bottom, left, right, center
  - combine multiple with |
  - set `gravity` on the layout to adjust all widgets; set `layout_gravity` on an individual widget

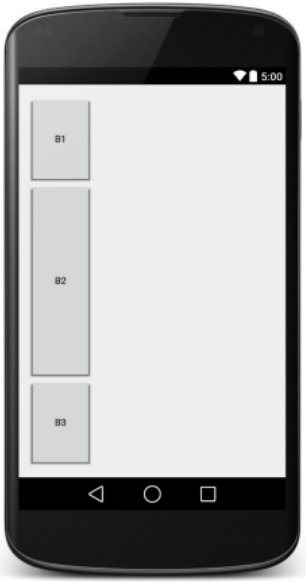



```
<LinearLayout ...
        android:orientation="vertical"
        android:gravity="center|right">
    <Button ... android:text="Button 1" />
    <Button ... android:text="Button 2 Hooray" />
    <Button ... android:text="Button 3" />
    <Button ... android:text="Button 4 Very Long Text" />
    <Button ... android:text="Button 5"
                android:layout_gravity="left" />
</LinearLayout>
```

Weight

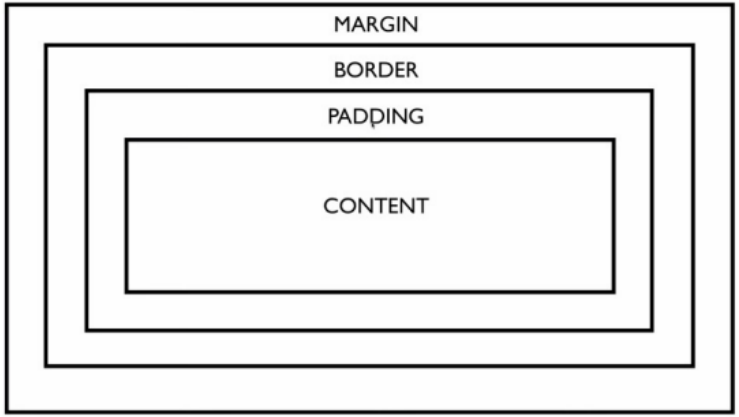
- **weight**: gives elements relative sizes by integers
  - widget with weight ***K*** gets ***K***/total fraction of total size
  - cooking analogy: "2 parts flour, 1 part water, ..."

```
<LinearLayout ...
        android:orientation="vertical">
    <Button ... android:text="B1"
               android:layout_weight="1" />
    <Button ... android:text="B2"
               android:layout_weight="3" />
    <Button ... android:text="B3"
               android:layout_weight="1" />
</LinearLayout>
```



Widget box model

- **content**: every widget or view has a certain size (width x height) for its content, the widget itself
- **padding**: you can artificially increase the widget's size by applying padding in the widget just outside its content
- **border**: outside the padding, a line around edge of widget
- **margin**: separation from neighboring widgets on screen



Width and Height of widget

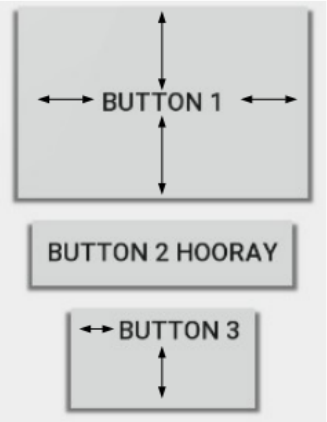
- **width** and **height** of a widget can be:
  - `wrap_content` : exactly large enough to fit the widget's content
  - `match_parent` : as wide or tall as 100% of the screen or layout
  - a specific fixed width such as 64dp *(not usually recommended)*
    - *dp = device pixels; dip = device-independent pixels; sp = scaling pixels*

```
<Button ...
        android:layout_width="match_parent"
        android:layout_height="wrap_content" />
```

Padding

- **padding**: extra space *inside* widget
  - set `padding` to adjust all sides; paddingTop, Bottom, Left, Right for one side
  - usually set to specific values like 10dp

    (some widgets have a default value ~16dp)

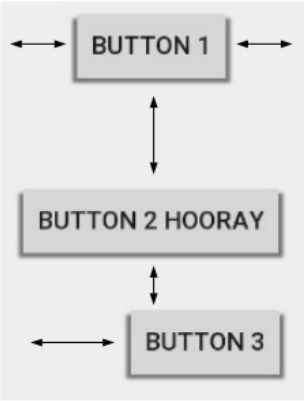



```
<LinearLayout ...
        android:orientation="vertical">
    <Button ... android:text="Button 1"
                android:padding="50dp" />
    <Button ... android:text="Button 2 Hooray" />
    <Button ... android:text="Button 3"
                android:paddingLeft="30dp"
                android:paddingBottom="40dp" />
</LinearLayout>
```

Margin

- **margin**: extra space *outside* widget to separate it from others
  - set `layout_margin` to adjust all sides; layout_marginTop, Bottom, Left, Right
  - usually set to specific values like 10dp

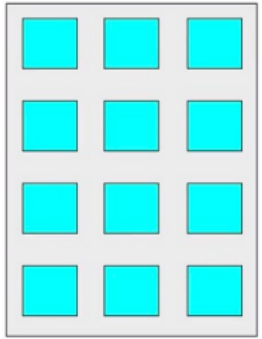
    *(set defaults in res/values/**dimens.xml**)*




```
<LinearLayout ...
        android:orientation="vertical">
    <Button ... android:text="Button 1"
                android:layout_margin="50dp" />
    <Button ... android:text="Button 2 Hooray" />
    <Button ... android:text="Button 3"
                android:layout_marginLeft="30dp"
                android:layout_marginTop="40dp" />
</LinearLayout>
```

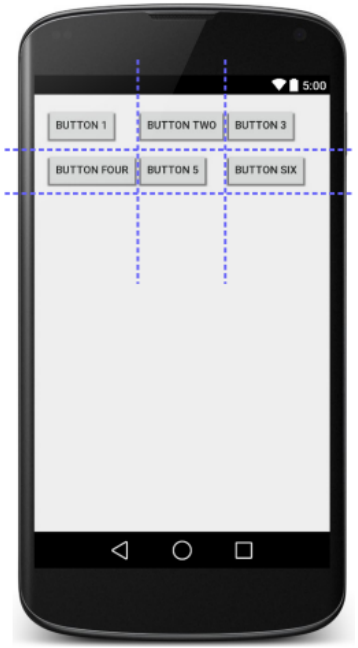
Grid Layout

- lays out widgets/views in lines of **rows** and **columns**
  - orientation attribute defines row-major or column-major order
  - introduced in Android 4; replaces older TableLayout
- by default, rows and columns are equal in size
  - each widget is placed into "next" available row/column index unless it is given an explicit `layout_row` and `layout_column` attribute
  - grid of 4 rows, 3 columns:



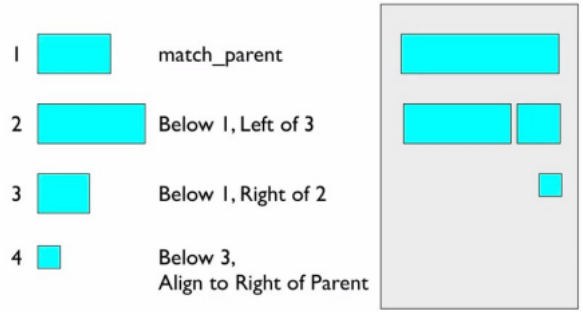
Grid Layout

```
<GridLayout ...
        android:rowCount="2"
        android:columnCount="3"
        tools:context=".MainActivity">
    <Button ... android:text="Button 1" />
    <Button ... android:text="Button Two" />
    <Button ... android:text="Button 3" />
    <Button ... android:text="Button Four" />
    <Button ... android:text="Button 5" />
    <Button ... android:text="Button Six" />
</GridLayout>
```



Relative Layout

- each widget's position and size are relative to other views
  - relative to "parent" (the activity itself)
  - relative to other widgets/views
  - x-positions of reference: left, right, center
  - y-positions of reference: top, bottom, center
- intended to reduce the need for nested layouts



Relative Layout

```
<RelativeLayout ... >
    <Button ... android:id="@+id/b1" android:text="B1"
        android:layout_alignParentTop="true"
        android:layout_centerHorizontal="true" />
    <Button ... android:id="@+id/b2" android:text="B2"
        android:layout_alignParentLeft="true"
        android:layout_below="@+id/b1" />
    <Button ... android:id="@+id/b3" android:text="B3"
        android:layout_centerHorizontal="true"
        android:layout_below="@+id/b2" />
    <Button ... android:id="@+id/b4" android:text="B4"
        android:layout_alignParentRight="true"
        android:layout_below="@+id/b2" />
    <TextView ... android:id="@+id/tv1"
        android:text="I'm a TextView!"
        android:layout_centerInParent="true" />
    <Button ... android:id="@+id/b5" android:text="B5"
        android:padding="50dp"
        android:layout_centerHorizontal="true"
        android:layout_alignParentBottom="true"
        android:layout_marginBottom="50dp" />
</RelativeLayout>
```

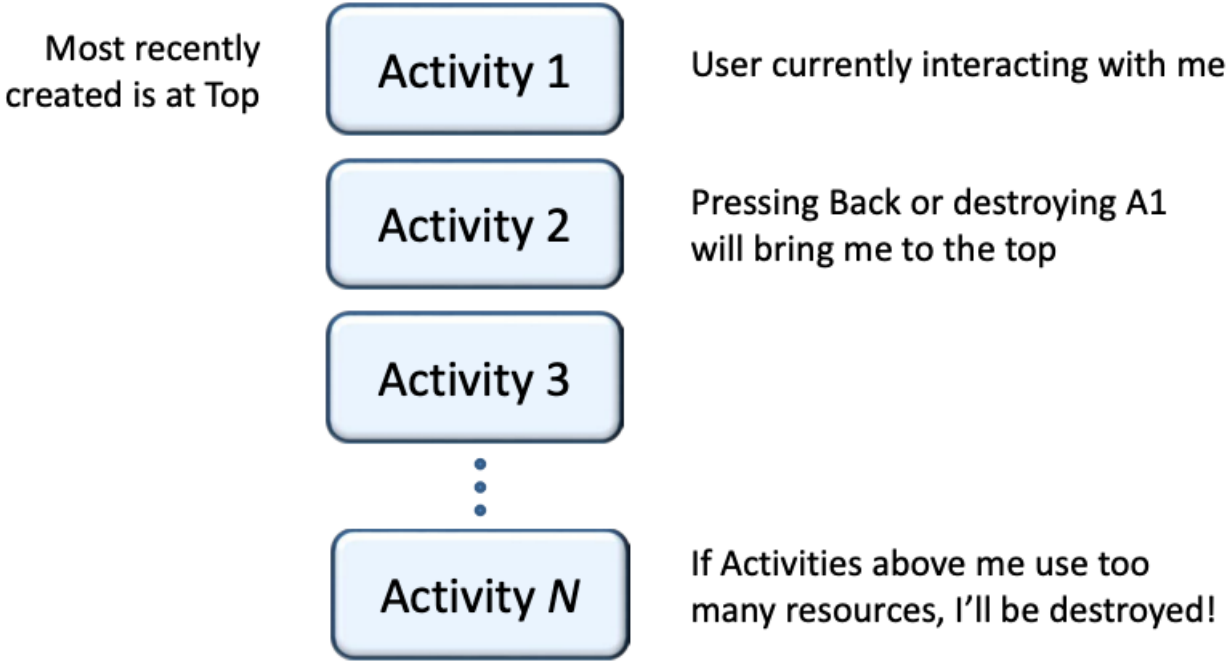
Frame Layout

- meant to hold only a single widget inside, which occupies the entirety of the activity
  - most commonly used with layout fragments (seen later)
  - less useful for more complex layouts

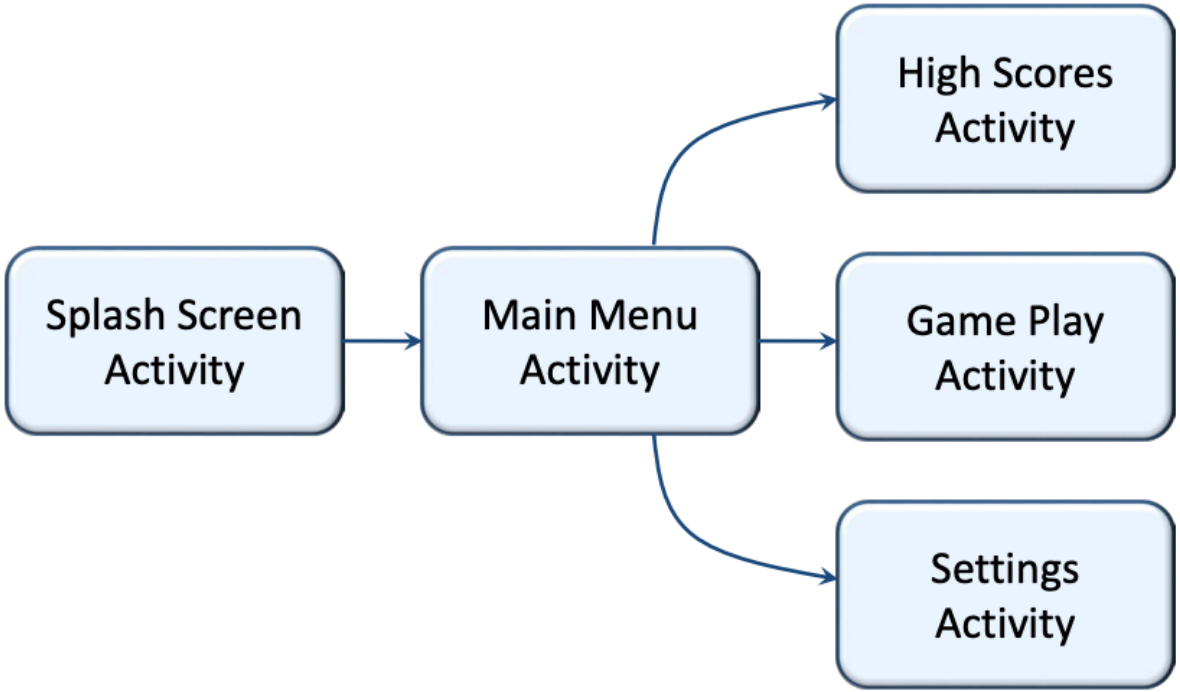
    (can put in multiple items and move them to "front" in Z-order)

```
<FrameLayout ... >
    <ImageView
     android:src="@drawable/jellybean"
     ... />
</FrameLayout>
```

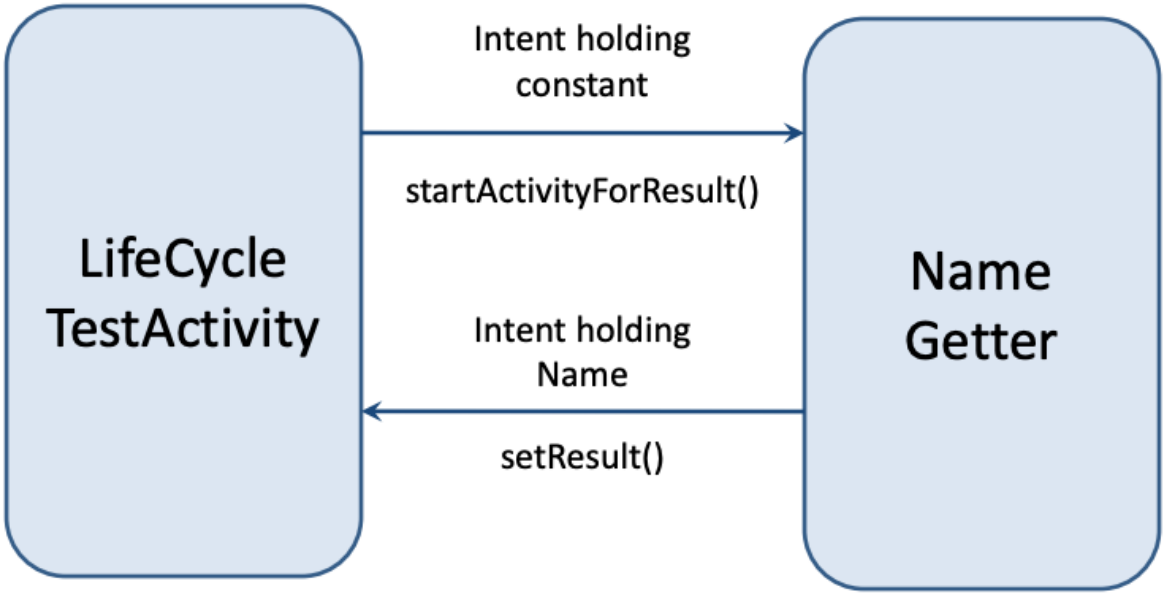
Activity Stack



Activity Stack



Intent


LifeCycle
TestActivity
Intent holding
constant
startActivityForResult()
Intent holding
Name
setResult()
Name
Getter

Intent Example

Intent to Launch Activity or change purpose of existing Activity

```
Context.startActivity()
Activity.startActivityForResult()
Activity.setResult()
```

Intent to Initiate Service or give new instructions to existing Service

```
Context.startService()
Context.bindService()
```

Intents intended for Broadcast Receivers

```
Context.sendBroadcast()
Context.sendOrderedBroadcast()
Context.sendStickyBroadcast()
```

The Android System finds the right application component to respond to intents, instantiating them if necessary.

Intent Example

```
/** Called when the user clicks the Send button */
public void sendMessage(View view) {
    Intent intent = new Intent(this, DisplayMessageActivity.class);
    EditText editText = (EditText) findViewById(R.id.edit_message);
    String message = editText.getText().toString();
    intent.putExtra(EXTRA_MESSAGE, message);
    startActivity(intent);
}
```

```
public final static String EXTRA_MESSAGE
    = "scottm.utexas.myfirstapp.MESSAGE";
```

Intent Example

```
public void takePhoto(View v) {
    // create directory if necessary
    File photoDir
            = new File(Environment.getExternalStorageDirectory()
            + "/intentExamplePhotos/");

    if(photoDir.mkdirs())
        Log.d(TAG, "mkdirs returned true: " + photoDir);
    else
        Log.d(TAG, "mkdirs returned false: " + photoDir);

    // create Intent to take picture via camera and specify locatio
    // to store image so we can retrieve easily
    Intent intent = new Intent(MediaStore.ACTION_IMAGE_CAPTURE);
    File file = new File(fileName);
    outputFileUri = Uri.fromFile(file);
    intent.putExtra(MediaStore.EXTRA_OUTPUT, outputFileUri);
    startActivityForResult(intent, TAKE_PICTURE);

}
```

Static List

- **dynamic list**: Content is read or generated as the program runs.
  - Comes from a data file, or from the internet, etc.
  - Must be set in the Java code.

  - Suppose we have the following file and want to make a list from it:

```
// res/raw/oses.txt
Android
iPhone
...
Max OS X
```

List event

- List views respond to the following events:
  - **setOnItemClickListener**(AdapterView.OnItemClickListener)

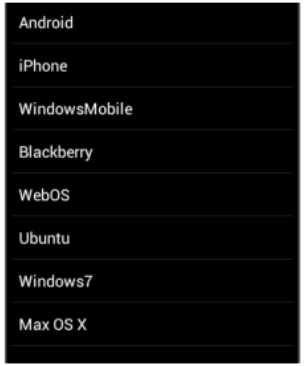
    Listener for when an item in the list has been clicked.

  - **setOnItemLongClickListener**(AdapterView.OnItemLongClickListener)

    Listener for when an item in the list has been clicked and held.

  - **setOnItemSelectedListener**(AdapterView.OnItemSelectedListener)

    Listener for when an item in the list has been selected.



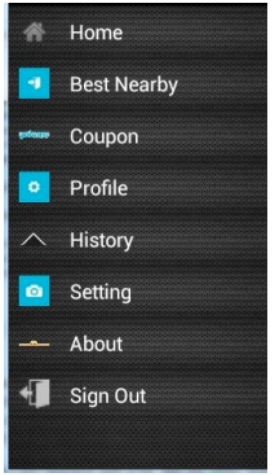
- Others:
  - onDrag, onFocusChanged, onHover, onKey, onScroll, onTouch, ...

List event listener example

```
ListView list = (ListView) findViewById(R.id.id);
list.setOnItemClickListener(
    new AdapterView.OnItemClickListener() {
        @Override
        public void onItemClick(AdapterView<?> list,
                                View row,
                                int index,
                                long rowID) {
            // code to run when user clicks that item
            ...
        }
    }
);
```

Custom list layout

- If you want your list to look different than the default appearance (of just a text string for each line), you must:
  - Write a short **layout XML file** describing the layout for each row.
  - Write a **subclass of `ArrayAdapter`** that overrides the **`getView`** method to describe what view must be returned for each row.



Custom list layout

```
<!-- res/layout/mylistlayout.xml -->
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout ... android:orientation="horizontal">
     <ImageView ... android:id="@+id/list_row_image"
         android:layout_width="100dp"
         android:layout_height="100dp"
         android:src="@drawable/smiley" />

     <TextView ... android:id="@+id/list_row_text"
         android:textStyle="bold"
         android:textSize="22dp"
         android:text=""
         android:background="#336699" />
</LinearLayout>
```

Permission

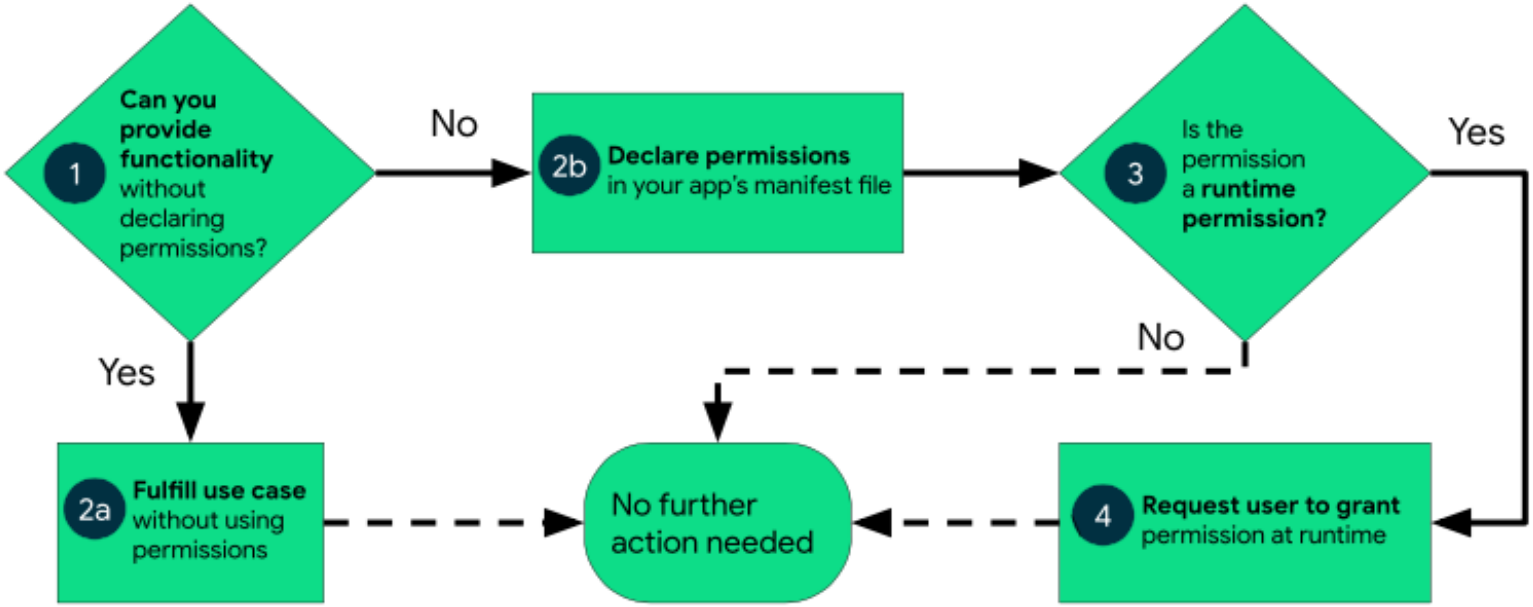

```xml
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    package="com.example.bt2">
    <uses-permission android:name="android.permission.CALL_PHONE"></uses-permission>


    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="BT2"
```

Android permission list:
https://developer.android.com/reference/android/Manifest.permission

https://gist.github.com/Arinerron/1bcaadc7b1cbeae77de0263f4e15156f

Permission


1
Can you provide functionality without declaring permissions?
No
2b
Declare permissions in your app's manifest file
3
Is the permission a runtime permission?
Yes
No
Yes
2a
Fulfill use case without using permissions
No further action needed
4
Request user to grant permission at runtime

Dangerous permission

- <uses-feature android:name="android.hardware.camera"

  android:required=“false”/”true” />

Exercise – Android Cat & Dog Identifier

- Access to this link:

  https://www.youtube.com/watch?v=ieHP5ICpDSY&list=PLxefhmF0pcPkV4dbZoefWWnv47DKJUdck

- Do as tutorial and make Cat & Dog Identifier.

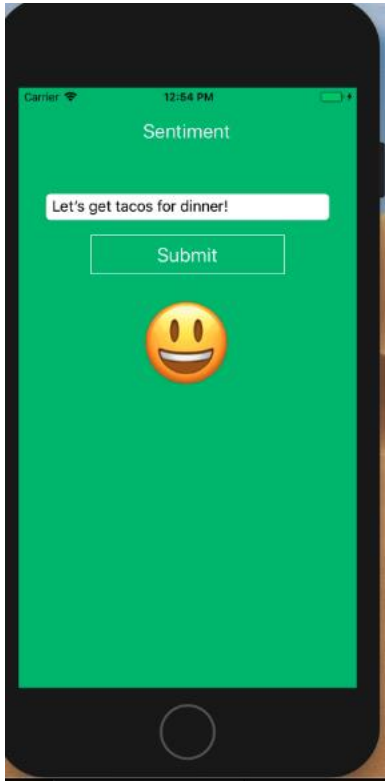
Exercise – Android Image to Text

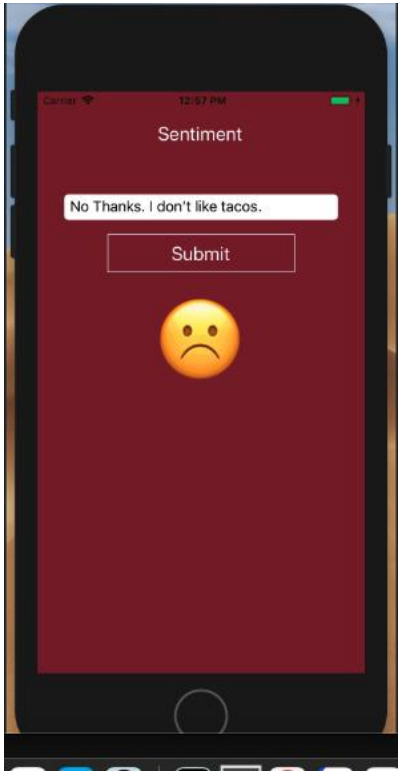
- Access to this link: https://www.youtube.com/watch?v=4KLKe9d-sLM&list=PLxefhmF0pcPlAFYbsWlfDaNS8rwyWKkPh&index=1
- Do as tutorial and make Image to Text app

Exercise – Android Mobile OCR App (Image to Text)

- Access to this link: https://www.youtube.com/watch?v=4KLKe9d-sLM&list=PLxefhmF0pcPlAFYbsWlfDaNS8rwyWKkPh&index=1
- Do as tutorial and make Image to Text app

Homework – Android Sentiment Analysis for Vietnamese





Homework – Android Sentiment Analysis for Vietnamese

- Download this dataset, model and use previous android app to predict the results: https://huggingface.co/wonrax/phobert-base-vietnamese-sentiment using PhoBERT.
- About training you can use Google Colab: https://colab.research.google.com/
- Compare results with your exercise in lab 1
- Read more:
  - Build API: https://phamdinhkhanh.github.io/2020/03/23/FlaskRestAPI.htm
  - Read more about PhoBERT: https://phamdinhkhanh.github.io/2020/06/04/PhoBERT_Fairseq.html

Homework – Flappy X, with X is you

- Access to tutorial of this India guys: https://www.youtube.com/watch?v=jKVLdlP3maU&list=PLhcYacorV7U7OM-IR14AupJDglqjnSPLX
- Build your own's flappy X app with X is your name and the bird's icon is your profile picture getting from CV exercise in lab 1.
- Next step is making your app smarter! Add AI to your Flappy X app using NEAT Python: https://www.youtube.com/watch?v=OGHA-elMrxI
- Finally, create a site and include your information from CV Exercise in Lab 1.

Q & A

Thank you for listening

"Coming together is a beginning;
Keeping together is progress;
Working together is success."
- HENRY FORD