Topics in Computer Science: Mobile Application Development

Prof. El-Sheikh

CS-591, Mobile Application Development

Lecture 3

Today’s Agenda

- Activity Lifecycle Revisited
- Bundling Information in Android
  - Two Ways
- Resources
  - String Resources – why not hardcode?
    - Multiple Language Support, Localization

  - Image Resources/Adding Images/Changing App Icon (if time)
  - Layouts (if time)
- Multiple Activities and navigating between them.
- Saving Activity State
- Simple Intents and Activity Management
- AndroidManifest.xml

Lecture Plan/Scaffolding

1. Review Activity Lifecycle + Activity Lifecycle Destruction and Recreation.
    - Show What happens to information in EditText and TextView, when device is rotated.
    - Discuss why sometimes state remembers, and why it's not reliable.
    - This will lead to discussion of using bundles.
2. Show that whenever a hard coded string is entered, the lightbulb warning symbol appears.
    - Discuss why.
    - Discuss how to resolve.
    - Discuss adding additional languages.
    - Show how to edit Strings XML
    - This will Segway into discussion of how images and layouts and everything else design related can be stored inside of resources.

- Discussion Additional Layouts, Images, Etc. Show how images are different, must create directory first. Show how layouts must have same components.
- Show early binding again with Checkboxes.
- Discuss what's inside an APK

Administrivia

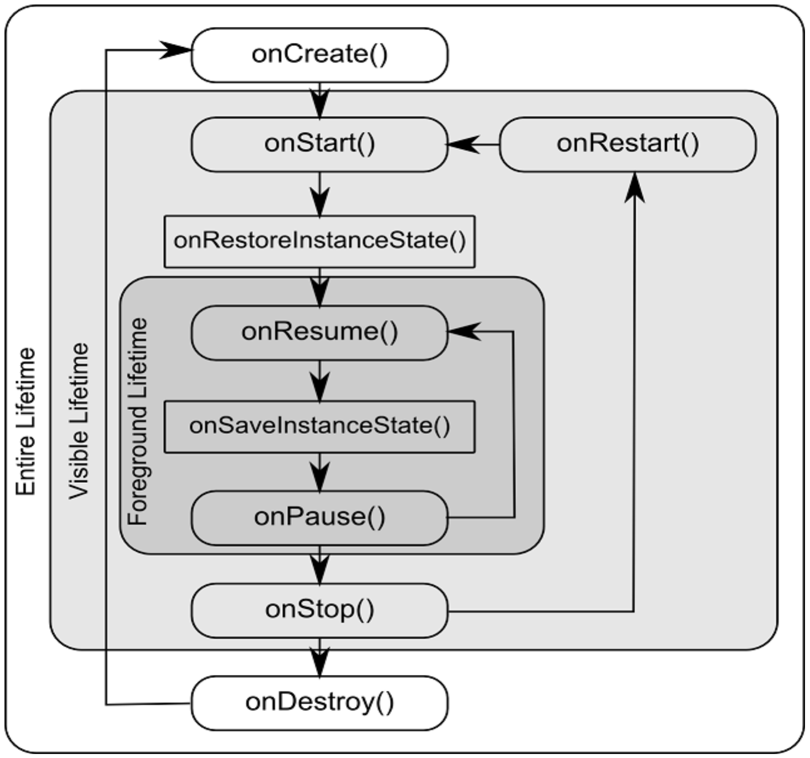
- I will assign permanent project teams next week.
  - Be flexible and nice to one another.
  - Remember don’t be a skater or an excuse maker.
- If you already know who you would like to work with let me know today.
- Homework 2 will be collected.
- Never email homework
- Homework 3 will be submitted using Blackboard Submit.
- Remember no Makeups of any kind.
  - To be fair: Will drop the lowest assignment, but try to do them all.
  - To be fair: Will excuse one absence for everyone, but try not to miss.

Administriva

- A note on Views. You will need to do some small amounts of research to get the properties and events just right. It just takes practice.
- Feel free to work in groups on all assignments
  - You must specify clearly who you worked with.
- Will be submitting code from now on in Bb.
  - Please submit one zip file per program, unless otherwise specified.

Bundles

Bundles and Activity Lifecycle



- What is Activity State?
  - eg, text in a textview, or the color of a button, or the position of seekbar, etc,
- When might an Activity be destroyed and recreated?
  - eg, simple screen rotation. Changing Locale, app going in background.
- Users expect their activity state to persist through these destructions/recreations.
- How is this done?
  - Using a Bundle of Information.
  - A Bundle is just a hashmap of info.
- ***Let's Try It.***

Data Bundles in Android

- Bundle Objects are the preferred way of passing data in Android.
- Sometimes the Bundle can hold Application State Information
  - Eg, when an activity is destroyed and recreated.
- Sometimes the Bundle can be used to pass information.
  - Eg, when one activity initiates the creation of another activity.

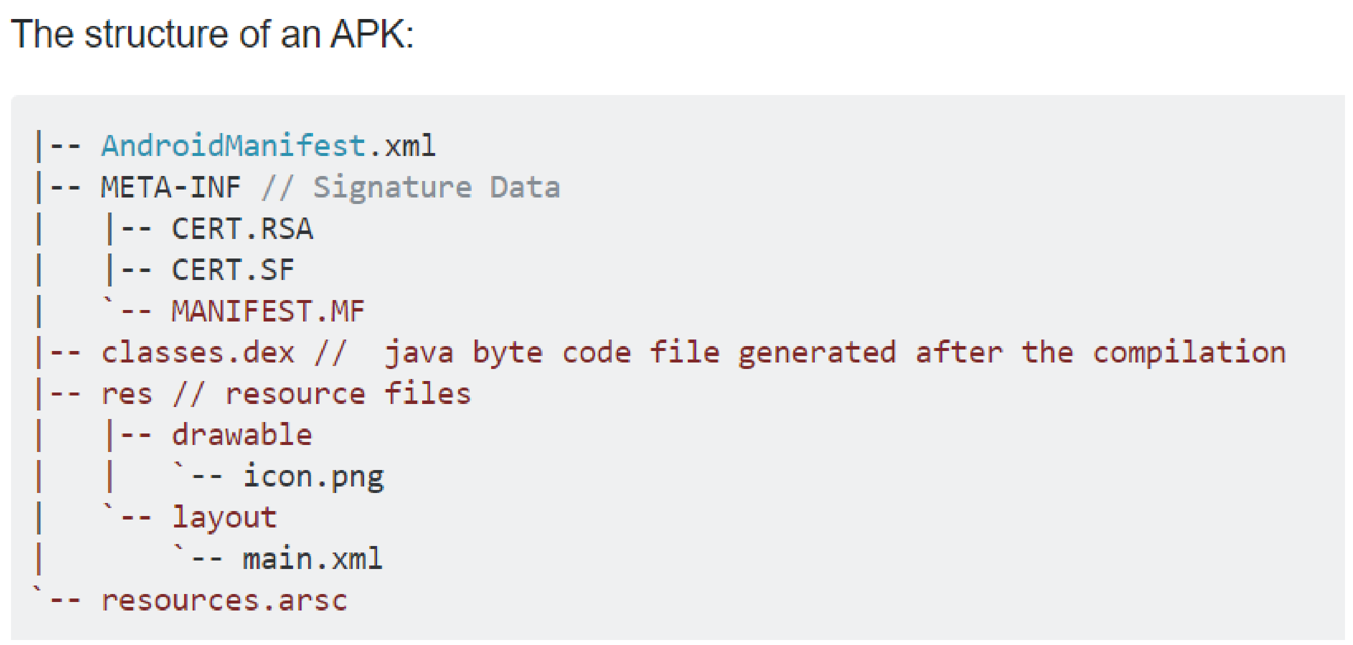
  - *Let's Try it, both ways.*
  - *Lect3_Bundles_Two_Ways*

APK & Resources

What is an APK?

- An APK is android package, basically in zip format with the apk extension. The apk contains several things, but primarily it contains a manifest, compiled code, and resources.
- Some of these resources are uncompiled.

What’s inside the ~~.zip~~ .apk



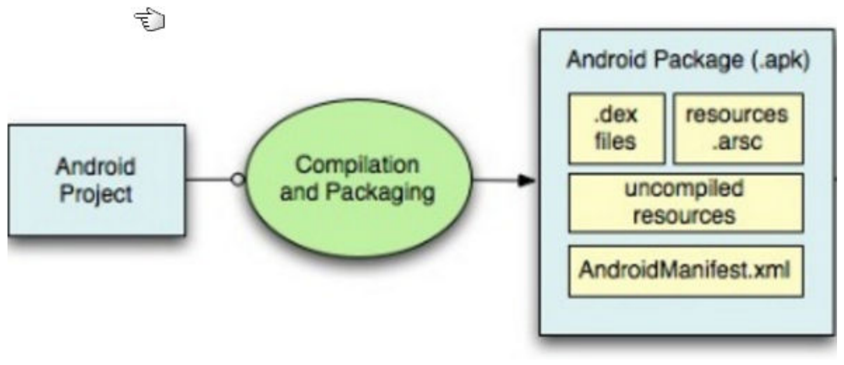
The structure of an APK:

```
|-- AndroidManifest.xml
|-- META-INF // Signature Data
|    |-- CERT.RSA
|    |-- CERT.SF
|    `-- MANIFEST.MF
|-- classes.dex //  java byte code file generated after the compilation
|-- res // resource files
|    |-- drawable
|    |    `-- icon.png
|    `-- layout
|         `-- main.xml
`-- resources.arsc
```

- The reference above is a bit dated, but is still germaine.
- Ref: https://stackoverflow.com/questions/8415098/what-is-inside-an-android-apk-file

Resources

- Build process results in an apk file which is eventually signed and uploaded to an Android Device.
- Resources are stored separately. They aren't "baked in".



- Android takes the added step of assigning consistent ID's for resources with the same name. *More on this later.*
- This is a great model. It means we can easily change things without adding code to handle device configurations.

Resources

- Android packages resources separately from the java/dex byte code
- ***Why does it do this?***
- ***Would it be easier to make code changes to handle the myriad device configurations?***
  - ***Screen Size***
  - ***Orientation***
  - ***Language/Locale***
  - ***PixelDensity***
  - ***Country Code***
  - ***…***

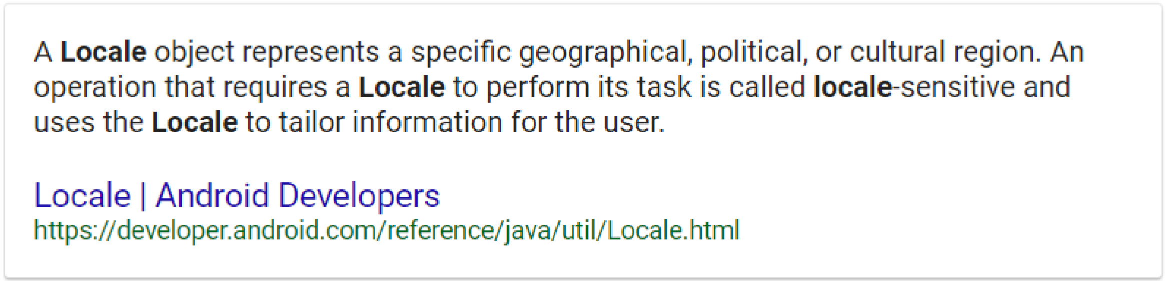
Things that will trigger resources differences and even dynamic changes.

- Android Framework will automatically choose resources to render, eg, strings, images, layouts, etc. depending on the device, it’s pixel density, it’s locale, it’s orientation, etc.

- Things that will trigger this switch:
  - Changing Locale
  - Changing Language of phone, how are these different?
  - Different device sizes
  - Different device orientations
  - Different device DPI’s (resolutions)

Resources: Locale

What is a Locale?


A **Locale** object represents a specific geographical, political, or cultural region. An operation that requires a **Locale** to perform its task is called **locale**-sensitive and uses the **Locale** to tailor information for the user.

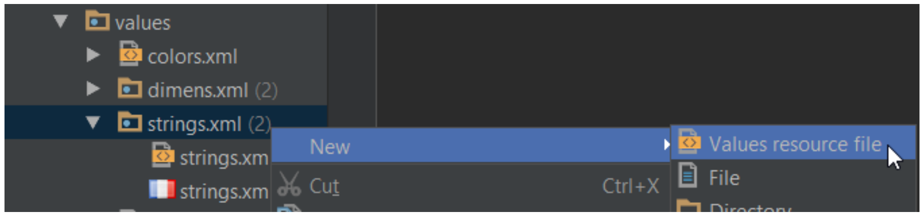
Locale | Android Developers
https://developer.android.com/reference/java/util/Locale.html

- In Android your application can automatically render itself differently by using different resources for a specific locale.
- For example, you can create resources for locales like, Greece, China, Japan, etc.
- *Question: What happens if your android App is run in Spain, but you haven't defined resources for that Locale? Will your App just crash?*
- *Answer: __No, it will use the default Locale.____________*

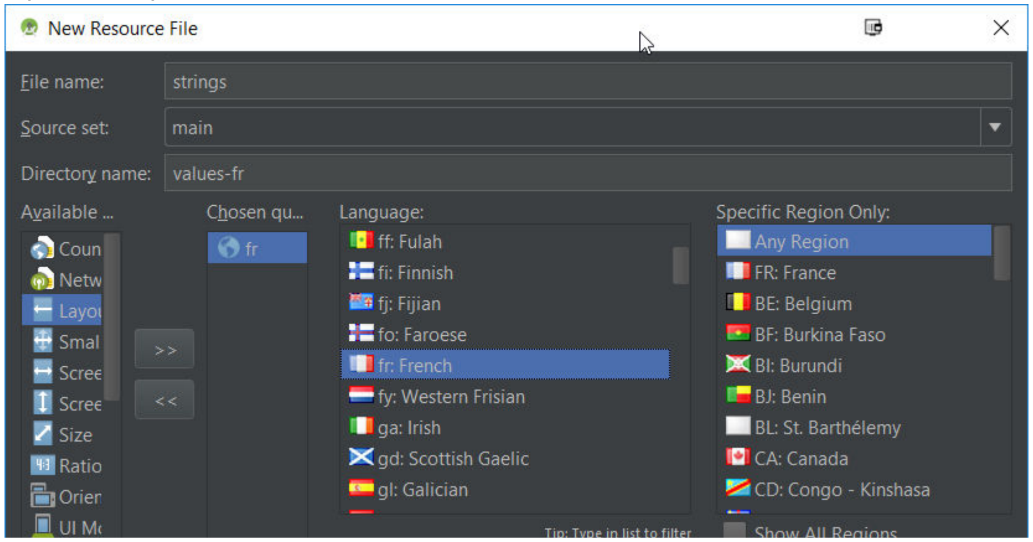
Ref: https://stackoverflow.com/questions/23348954/android-get-device-locale

Localizing Strings

1. Right Click on the strings.xml resource and create a New "Values resource file".



2. Name the file name "strings"
3. Choose the Country and optionally the specific region.



File name: strings

Source set: main

Directory name: values-fr

Chosen qu...: fr

Language: fr: French

Specific Region Only: Any Region

What’s inside an .apk file?

- The .apk file is generated as a result of compiling/recompiling your java code into dex code.
- The .apk file also includes:
  - A manifest, listing all of the stuff inside
  - A set of resources which can include Images & Strings.
- Why not just compile the Images and Strings into the byte-code?
  - App would certainly be faster if resources were *baked* in.

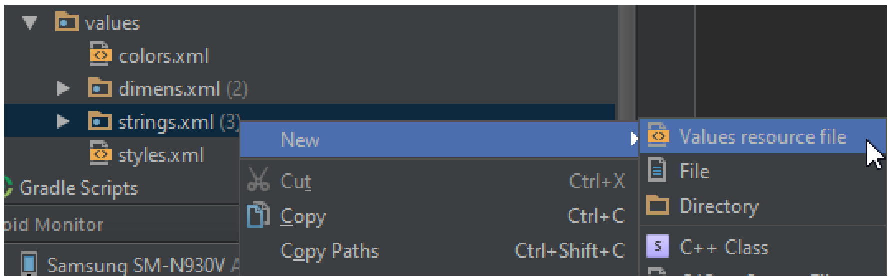
What’s inside an .apk file

- By separating resources from source code Android Apps can more effectively run across devices with different properties.
    - Different pixel densities
    - Portrait vs. Landscape
    - English vs. Spanish vs. Mandarin vs. Russian vs…
- Every resource gets its own unique ID. ***`R.java`***
- These IDs are typically referenced in the xml files, but can also be referenced from other java source code.
- Separation of resources from compiled code simplifies automatic configuration process when the app runs on different devices and in different locales.

String Resources

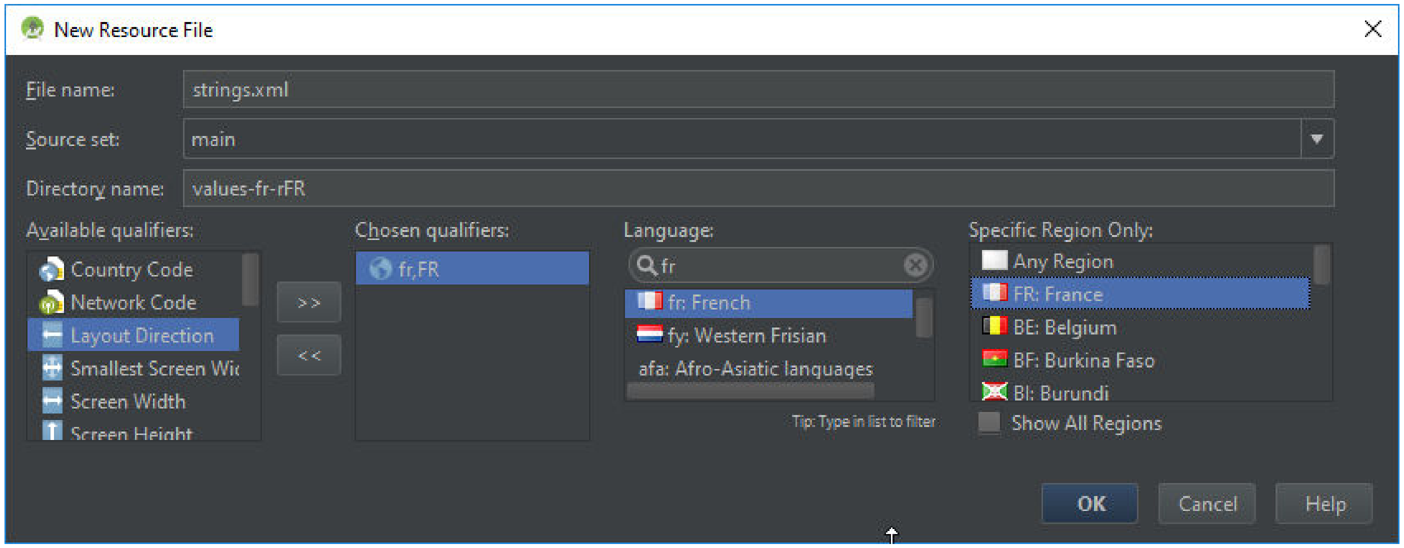
String Resources

- By making Strings a resource, it becomes easier to handle locale changes.
- Eg, if you change your phone's Locale to France, and want your App to be in French, create a new resource file. Just right click on Strings.xml and choose new Values resource File.



How to Handle Multiple Languages

- Now add a string resource.
  - Choose Locale as a qualifier, note it will disappear after you select it.
  - Choose a Language then choose the locale specific dialect.
- Next Change the Locale of your device in Settings.
  - Simple Step-by-Step here. http://www.wikihow.com/Change-the-Language-in-Android


New Resource File

File name: strings.xml

Source set: main

Directory name: values-fr-rFR

Available qualifiers: Country Code, Network Code, Layout Direction, Smallest Screen Wic, Screen Width, Screen Height

Chosen qualifiers: fr,FR

Language: fr

fr: French (selected), fy: Western Frisian, afa: Afro-Asiatic languages

Tip: Type in list to filter

Specific Region Only: Any Region, FR: France (selected), BE: Belgium, BF: Burkina Faso, BI: Burundi

Show All Regions

Let's Try It.

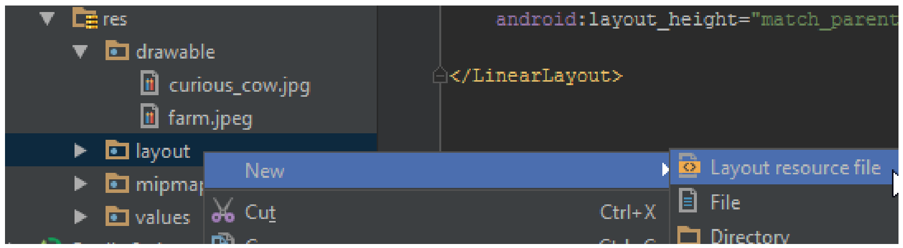
More on Locales

- What if your app does NOT support a language in the current Locale
  - Be nice, get it translated.
  - The default language in your app will be used.
- What if you have “conflicting” string resources. Eg, the same String Resource in a layout and in a language?
  - Language Wins!
  - But don’t do this, create a multiply qualified string resource, eg, Locale + Orientation + …

Layout Resources

Layout Resources

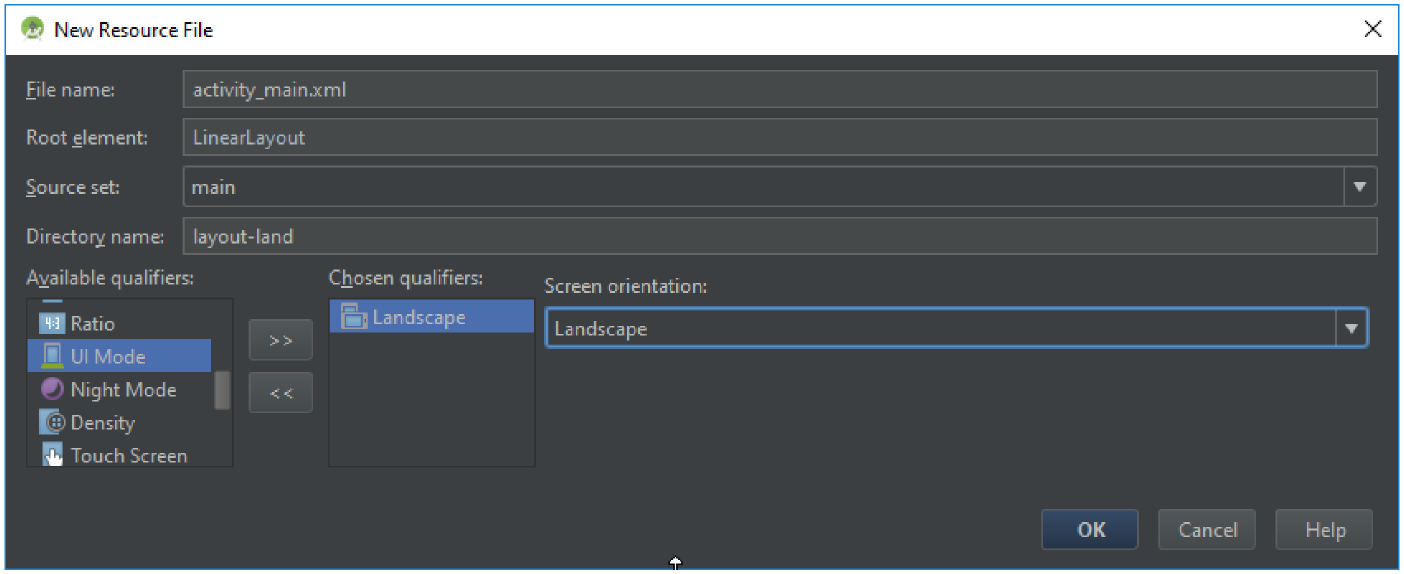
- A layout is just a resource. Eg, activity_main.xml is a resource with its own dynamically generated id in the R.java file.
- Layouts typically get :inflated" in the onCreate event of your Java App.
  - Eg, `setContentView(R.layout.activity_main);`
- You as a developer can generate multiple layouts for the same activity.
  - ***Why would you want to do this? Isn't one Activity XML per java file enough?***
  - ***How does Android choose the right activity_main file, given only one onCreate?***
- Right Click on layout folder under res.
- Choose ***`New>Layout resource file`***



```
android:layout_height="match_parent

</LinearLayout>
```

Let's Try It: Adding a new layout for portrait mode

- Be sure to keep the filename the same!
- When the device switches to landscape mode, Android OS will automatically grab the other activity_main.xml file.


New Resource File

File name: activity_main.xml

Root element: LinearLayout

Source set: main

Directory name: layout-land

Available qualifiers: Ratio, UI Mode, Night Mode, Density, Touch Screen

Chosen qualifiers: Landscape

Screen orientation: Landscape

Aside: Practicing Avoidance
Forcing portrait mode for an Activity

- There might be times when you want to disallow layout changes.
    - Eg, App only looks good in Portrait.

- In AndroidManifest.xml add the following property to the Activity.
    - **`Android:screenOrientation=“portrait”`**
    - **`Eg,`**

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

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

Additional Reading: Density Independence

- Official Android Studio on Density Independence
  - https://developer.android.com/guide/practices/screens_support.html
- MipMap Vs. Drawable Folders
  - http://stackoverflow.com/questions/28065267/mipmap-vs-drawable-folders

Image Resources

IDE doesn’t have good support for adding localized images

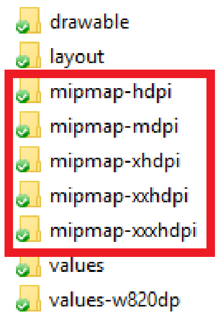
- Must do it manually, or use a 3rd party plugin.
- Ref: https://developer.android.com/guide/topics/resources/providing-resources.html

- Let’s Look at the directory Structure in Lect3_

Image Resources

- Images resources can be automatically maintained by Android.
- Android Studio will automatically change the pixel density of images for you, so your images look good on different devices.
- App icon is stored in the various mipmap-xxxxx folders. *Let's take a peak.*

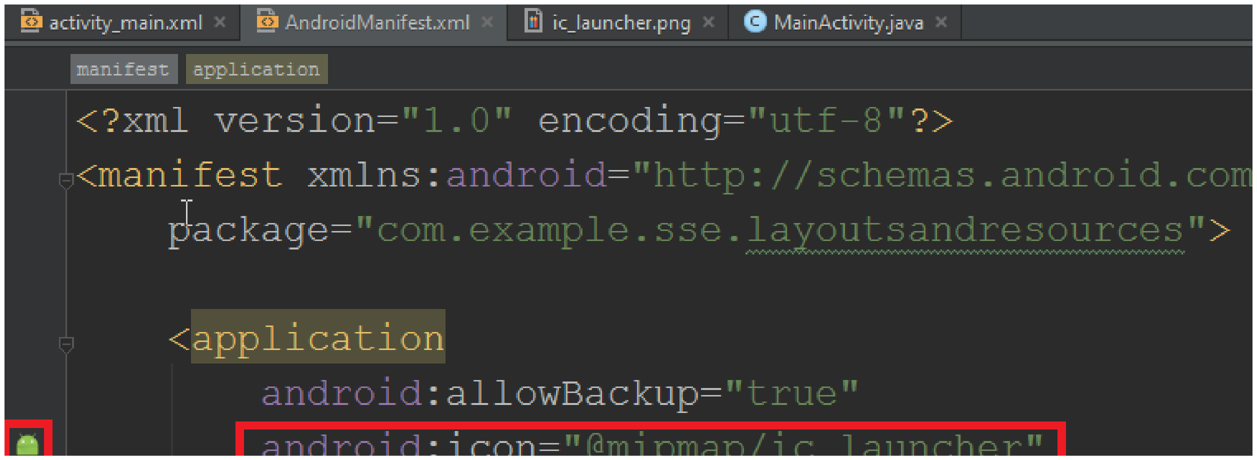
- Where do we keep images?
- In the /app/src/main/res/drawable



- IMPORTANT: image files must always be lower case!

Changing the App Icon

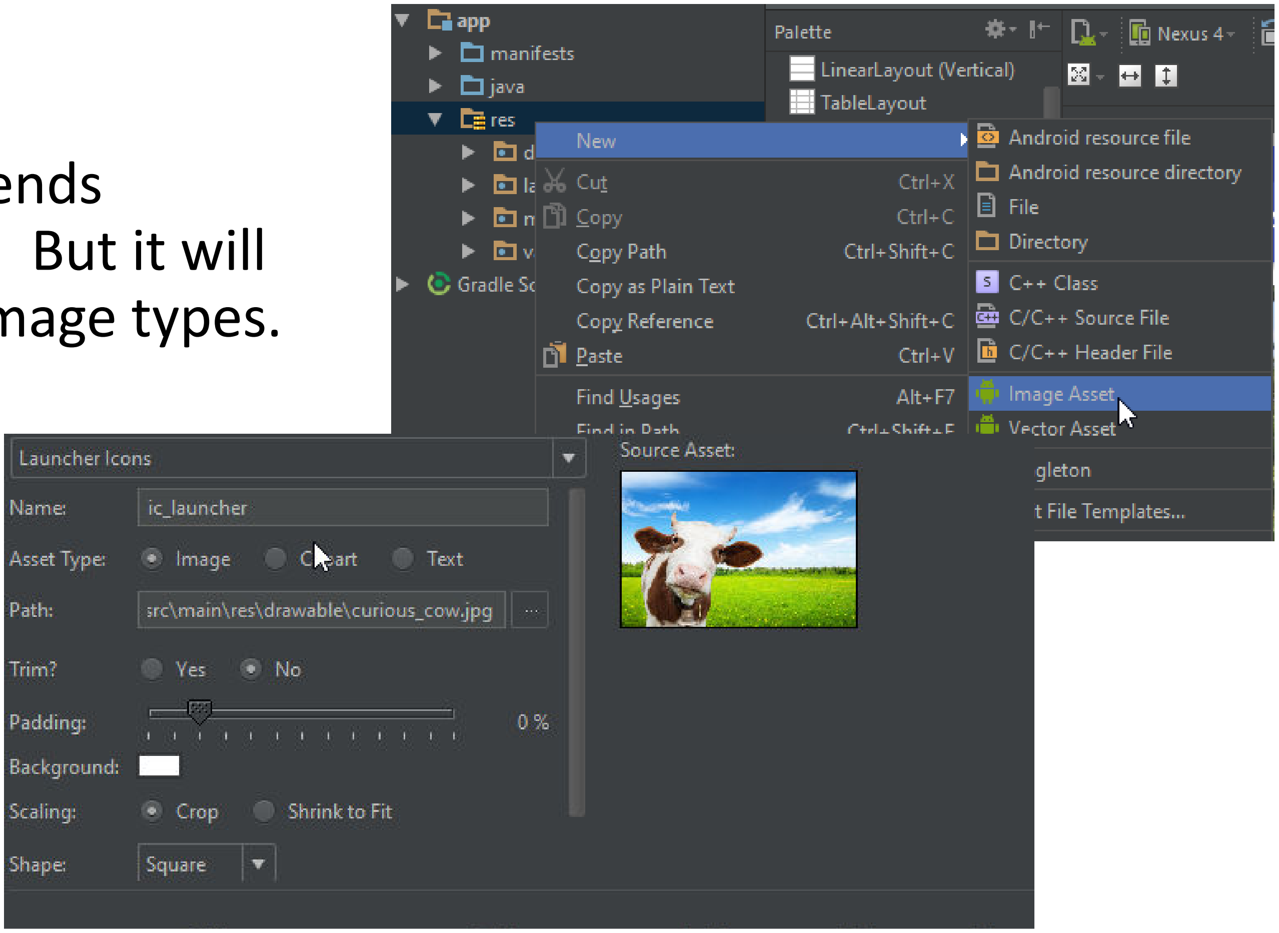
- Icons are stored in the mipmap resource directories. Multiple icons are stored, one for each resolution type.
- The icon is referred to in the AndroidManifest.xml. Change the file ic_launcher and you change the default icon.



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

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

Updating ic_launcher resource

- New Image Asset
- Android Recommends 512x512 png files. But it will work with other image types.



Launcher Icons

Name: ic_launcher

Asset Type: [✓ Image] Clipart Text

Path: src\main\res\drawable\curious_cow.jpg

Trim? Yes [✓ No]

Padding: 0 %

Background:

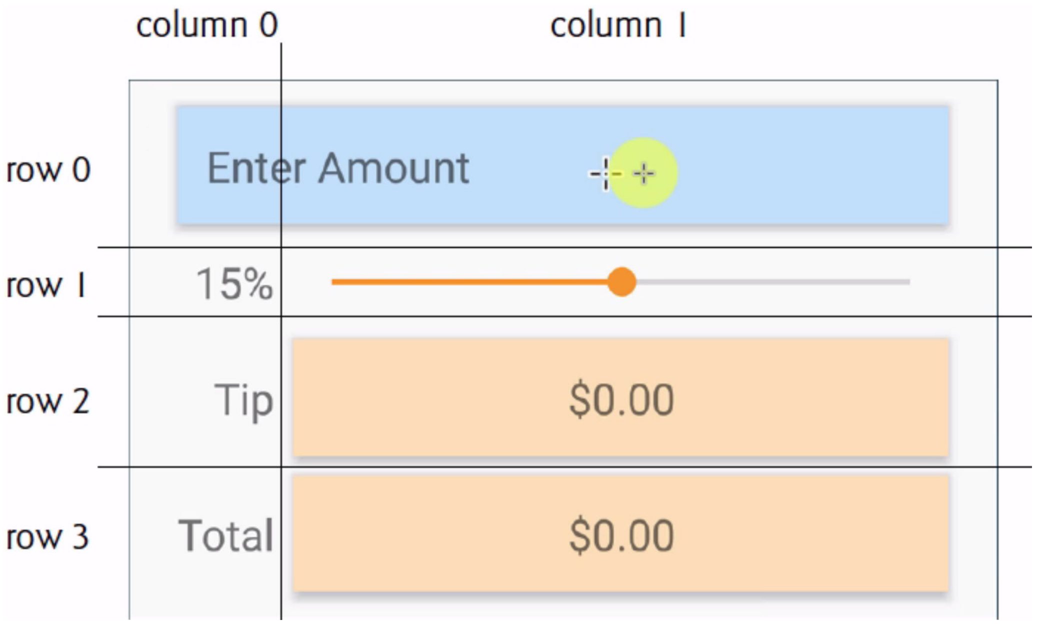
Scaling: [✓ Crop] Shrink to Fit

Shape: Square

You Try It.
Grid Layout

Story Boarding with Layouts in Mind

TIP Calculator, Labeled GridLayout View, helps plan our App.



a) Initial GUI

b) GUI after user enters the amount 56.32 and changes the tip percentage to 25%

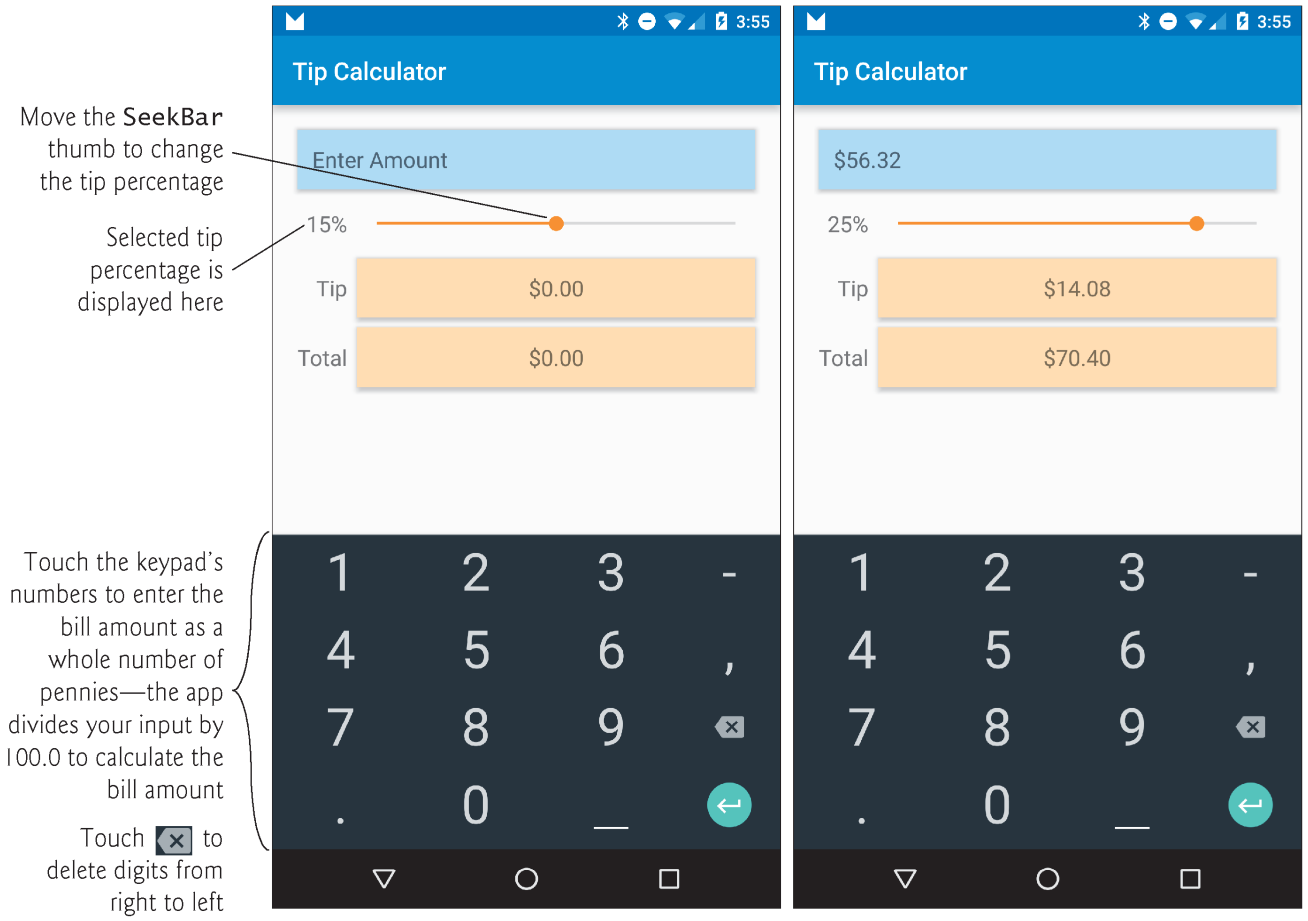

Move the SeekBar thumb to change the tip percentage
Selected tip percentage is displayed here
Touch the keypad's numbers to enter the bill amount as a whole number of pennies—the app divides your input by 100.0 to calculate the bill amount
Touch ⌫ to delete digits from right to left
Tip Calculator
Enter Amount
15%
Tip
$0.00
Total
$0.00
Tip Calculator
$56.32
25%
Tip
$14.08
Total
$70.40

Fig. 3.1 | Entering the bill total and calculating the tip.

Intents & Activity Management

Introduction to Intents

- Intents are a key feature of android.
- An intent is a message.
    - Message can target a single Activity
    - Or it can be broadcast, and multiple apps can respond.
- For example, if you are in an app that has *street addresses (eg, in your Contact, from Yelp, whatever)* and click “navigate” an intent is broadcast, and all apps that can perform navigation wake up. You pick the one you want.
- These other apps have intent-filters and android notifies them that you want navigation.
- Today we will focus on spawning multiple activities with intents.

Intents and Activity Management

- Example: App with two activities.
- Activity1 might be a login screen, that will only provide access to Activity2 if the user/pass is correct.
- How will Activity1 spawn Activity2? With an intent.
- In the AndroidManifest.xml we list all of the Activities that are part of the project. We also specify unique properties of these Activities, a signature.
- Activity1 will broadcast an intent to open Activity2 by using this *signature*.
- *Let's Try it.*

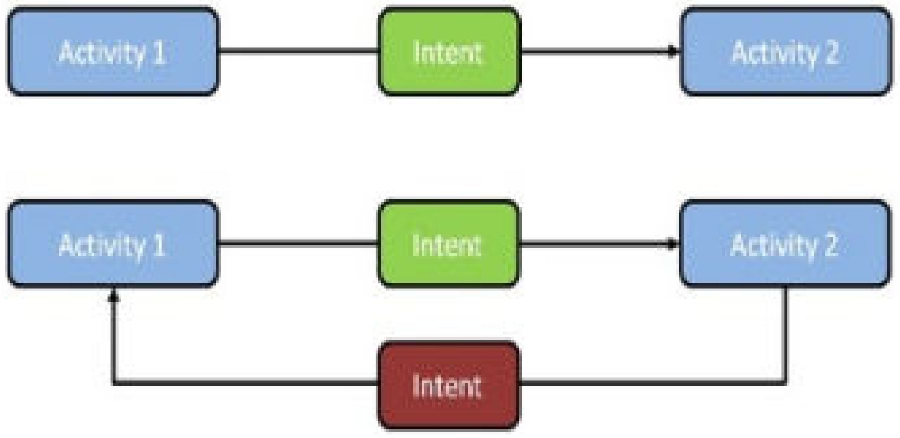


Image Ref: http://www.slideshare.net/AlexeyUstenko/android-application-structure

Let's Try It: Creating a Second Activity

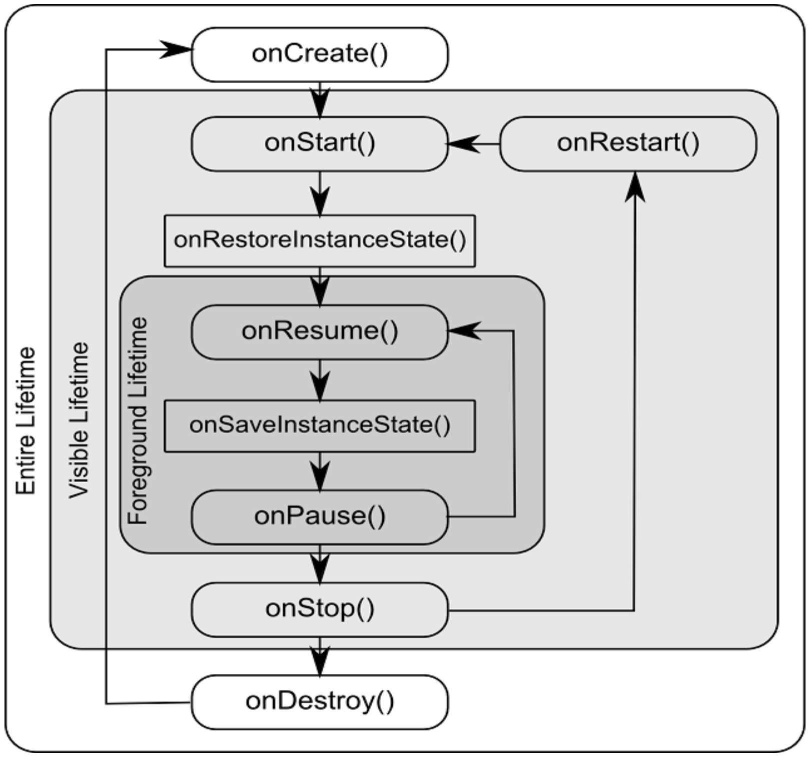
- Update Manifest, mostly automatic, but can do this manually.
- Create an explicit intent to open another Activity. *Let's Try it.*

```
<activity android:name=".MainActivity">
    android:screenOrientation="portrait"
    <intent-filter>
        <action android:name="android.intent.action.MAIN2" />

        <category android:name="android.intent.category.DEFAULT" />
    </intent-filter>
</activity>
```

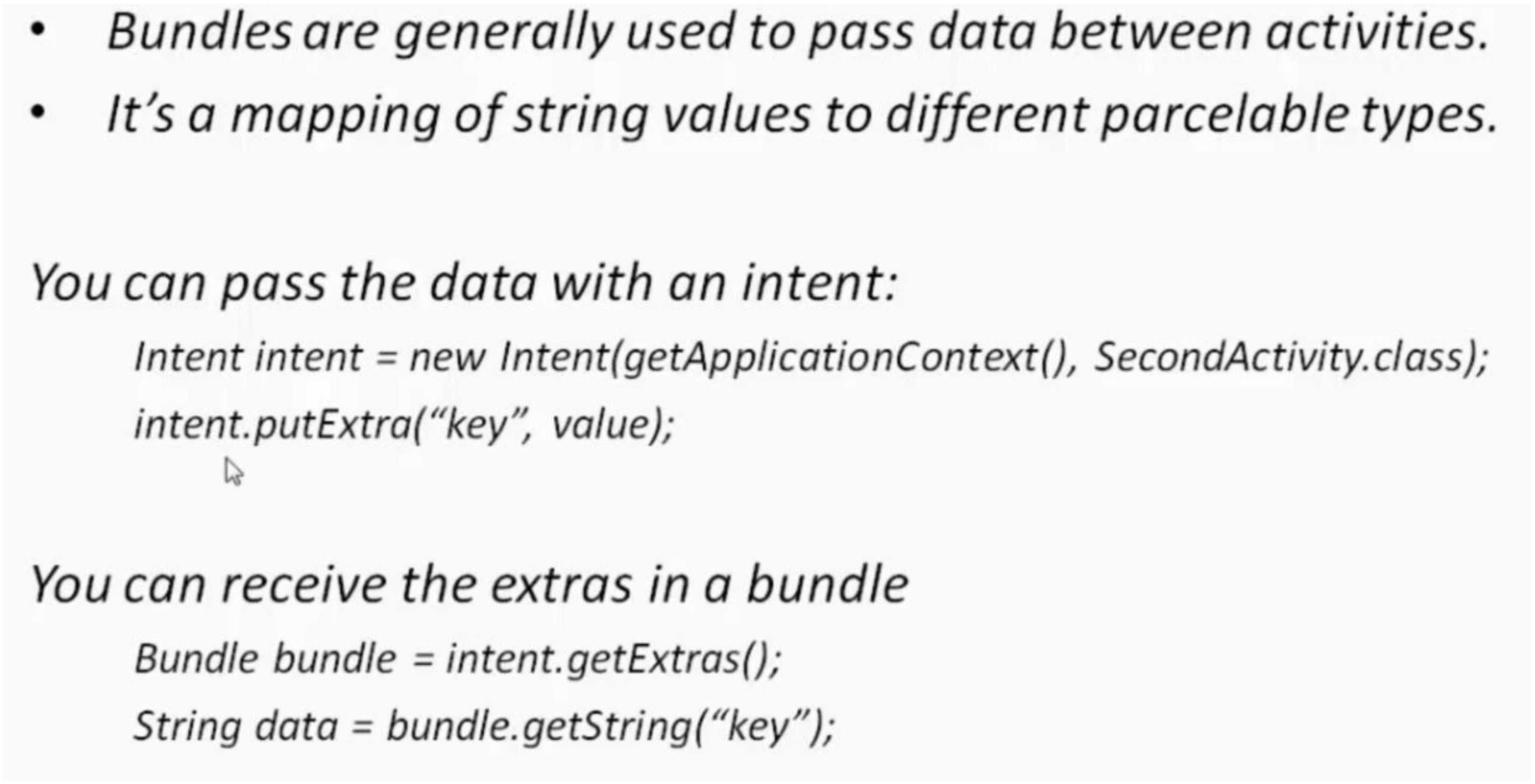
```
<activity android:name=".Main2Activity">
    <intent-filter>
        <action android:name="android.intent.action.MAIN" />

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

```
public void onClick(View v) {
     Intent i = new Intent(getApplicationContext(), MainActivity.class );
     startActivity(i);
}
```

Bundles and Activity Lifecycle



Passing Data in a bundle, from One Activity to the Next



- *Bundles are generally used to pass data between activities.*
- *It's a mapping of string values to different parcelable types.*

You can pass the data with an intent:

```
Intent intent = new Intent(getApplicationContext(), SecondActivity.class);
intent.putExtra("key", value);
```

You can receive the extras in a bundle

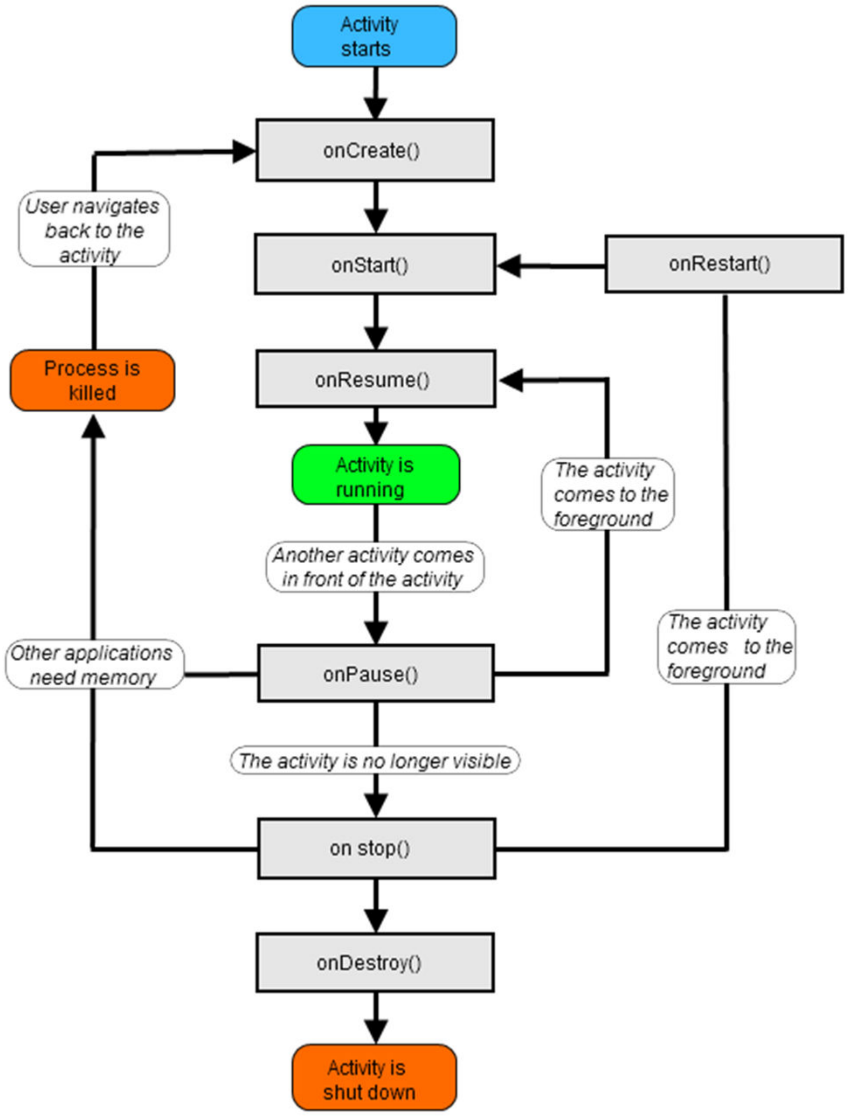
```
Bundle bundle = intent.getExtras();
String data = bundle.getString("key");
```

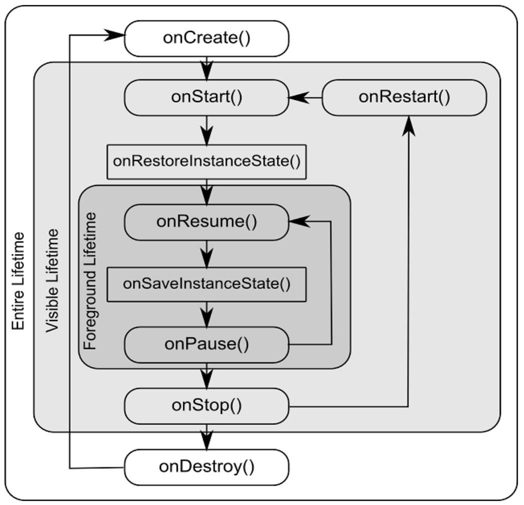
- **Ref: Husham Muneer**

Reminders

- If you know who you want to be in your group, let me know ASAP.
  - Otw, I will assign you to a group.
- Homework Submission this week is via Bb.
- Do not wait until last minute.

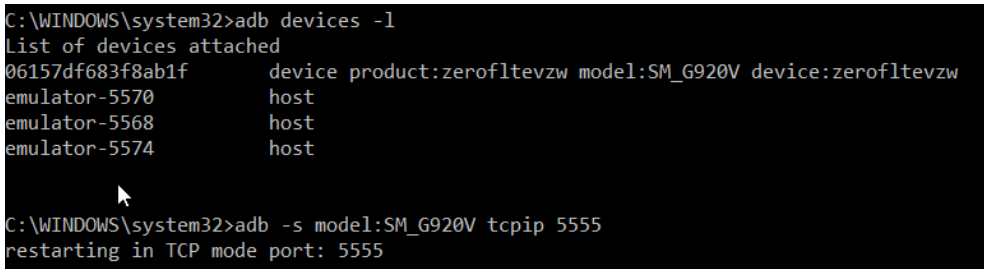
Misc


Activity starts
onCreate()
User navigates back to the activity
onStart()
onRestart()
Process is killed
onResume()
Activity is running
The activity comes to the foreground
Another activity comes in front of the activity
Other applications need memory
onPause()
The activity comes to the foreground
The activity is no longer visible
on stop()
onDestroy()
Activity is shut down

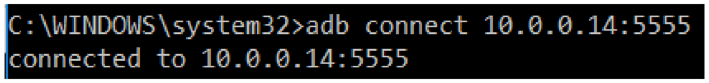

Entire Lifetime
Visible Lifetime
Foreground Lifetime
onCreate()
onStart()
onRestart()
onRestoreInstanceState()
onResume()
onSaveInstanceState()
onPause()
onStop()
onDestroy()

Pinchpoint – Debugging Wirelessly

- How to debug your physical device wirelessly
- No rooting, no additional sdks.
    1. get IP of your phone (WIFI Settings)
    2. Connect your device via USB.
    3. get list of devices attached to your pc.
    4. restart the TCP connection
    5. Connect again.
    6. Disconnect the USB – your device will still be connected and ready for debug.



```
C:\WINDOWS\system32>adb devices -l
List of devices attached
06157df683f8ab1f       device product:zerofltevzw model:SM_G920V device:zerofltevzw
emulator-5570          host
emulator-5568          host
emulator-5574          host


C:\WINDOWS\system32>adb -s model:SM_G920V tcpip 5555
restarting in TCP mode port: 5555
```



```
C:\WINDOWS\system32>adb connect 10.0.0.14:5555
connected to 10.0.0.14:5555
```