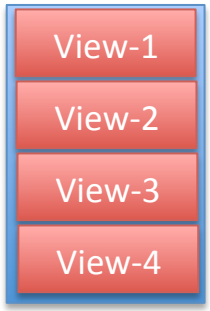
Android

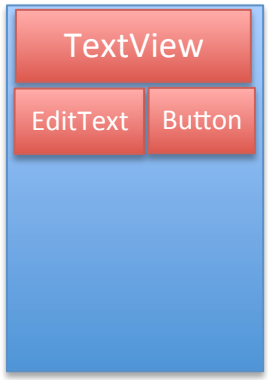
UI: Overview

- All UI elements are either
  - View
    - Base class for widgets: Button, TextView, EditText, CheckBox, RadioButton
  - ViewGroup
    - Subclass of View
    - Invisible container of Views or VieGroups
- Layouts
  - Subclasses of ViewGroup
  - Linear Layout
  - Relative Layout
  - Frame Layout
  - List View
  - Grid View
  - Table Layout

Linear Layout

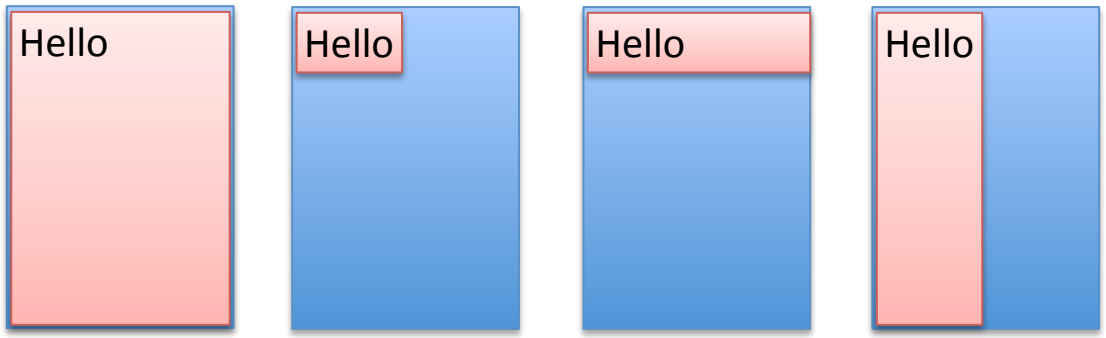
- Puts child views one after another, vertically or horizontally
  - <LinearLayout android:orientation="horizontal"|"vertical">
    <*view-1*…/>
    …
    <*view-N*…/>
    </LinearLayout>
- Nested LinearLayouts for complex a layout
  - <LinearLayout *vertical*>
    <TextView … />
    <LinearLayout *horizontal*>
    <EditText … />
    <Button …/>
    </LinearLayout>
    </LinearLayout>





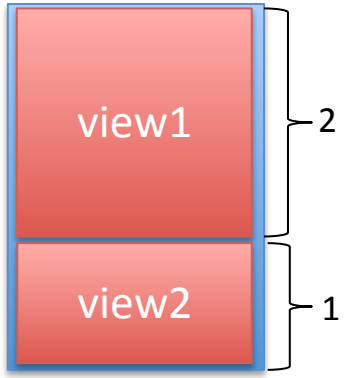
Controlling View(Group) Size

- <*viewname*
  android:layout_width=*size*
  android:layout_height=*size*
  android:layout_weight=*weight* />
- *size*= match_parent (fill up the parent's area)
  | wrap_content (only as much as needed for the content)
  | # px (# pixels)
  | # dp (# density-independent pixel (dip)
  1 dp = 1 px for 160 dpi,
  thus 160 dp=1 inch for any screen density)

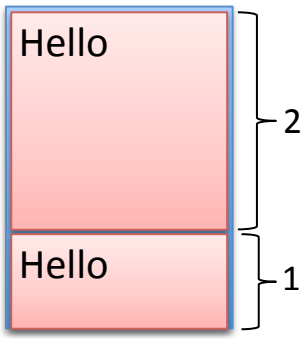


Proportional View(Group) Size

- *weight*: number defining the proportion of the view relative to sibling views' weights
  - If view1's weight=2, view2's weight=1
    then view1:view2 = 2:1, thus view1 is 2/3 wide of the parent size
  - <LinearLayout ..orientation=vertical>
    <*view1* ..height=wrap_cont/>
    <*view2* ..height=wrap_cont/>
    </LInearLayout>



View1: hei=wrap
View2: hei=wrap



View1: weight=2
View2: weight=1

View1: weight=1
View2: [weight=0]

LogCat & Log class

- LogCat
  - Command line tool to print system & app log messages
  - App can output log messages to LogCat via Log class
- LogCat window in Android Studio
  - Shows logcat output to a dedicated window
- Log class
  - Log.v (…) Less priority
  - Log.d (...)
  - Log.i (...)
  - Log.w (...)
  - Log.e (...) Higher priority
- Output format
  - Call Log.<priority>(<tag>, <message>) outputs:
    <date> <time> <PID>-<TID>/<package> <priority>/<tag>: <message>

Button Click Handling

1. Implement OnClickListener in MainActivity
2. Inner-class OnClickListener in MainActivity
3. Through XML layout

<View … android:id="@+id/*view_id*>
- Create an "ID" resource with id=*view_id*
- + means: If doesn't exist already, create

Event Handling by Listener Interface

- Make MainActivity class a View.OnClickListener
  - Implement interface View.OnClickListner
- Implement (override) onClick method
  - Call Log.d()
- Add MainActivity to the onClickListener of the button
  - Get button object using findViewById()

```
public class MainActivity extends AppCompatActivity implements OnClickListener {
   @Override
   protected void onCreate(…) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.activity_main);
      findViewById(R.id.<id>).setOnClickListener(this);
   }
   @Override
   public void onClick(View v) {…}    // for multiple buttons, distinguish views by v.getId()
 }
```

Event Handling by Inner-Class Listener

- Use inner-class listener (anonymous class instance) to handle button click event

```
public class MainActivity extends AppCompatActivity {
   @Override
   protected void onCreate(…) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.activity_main);
      findViewById(R.id.<id>).setOnClickListener( new OnClickListener() {
            @Override
            public void onClick(View v) {…}
      }
  }
}
```

Event Handling by XML & Handler

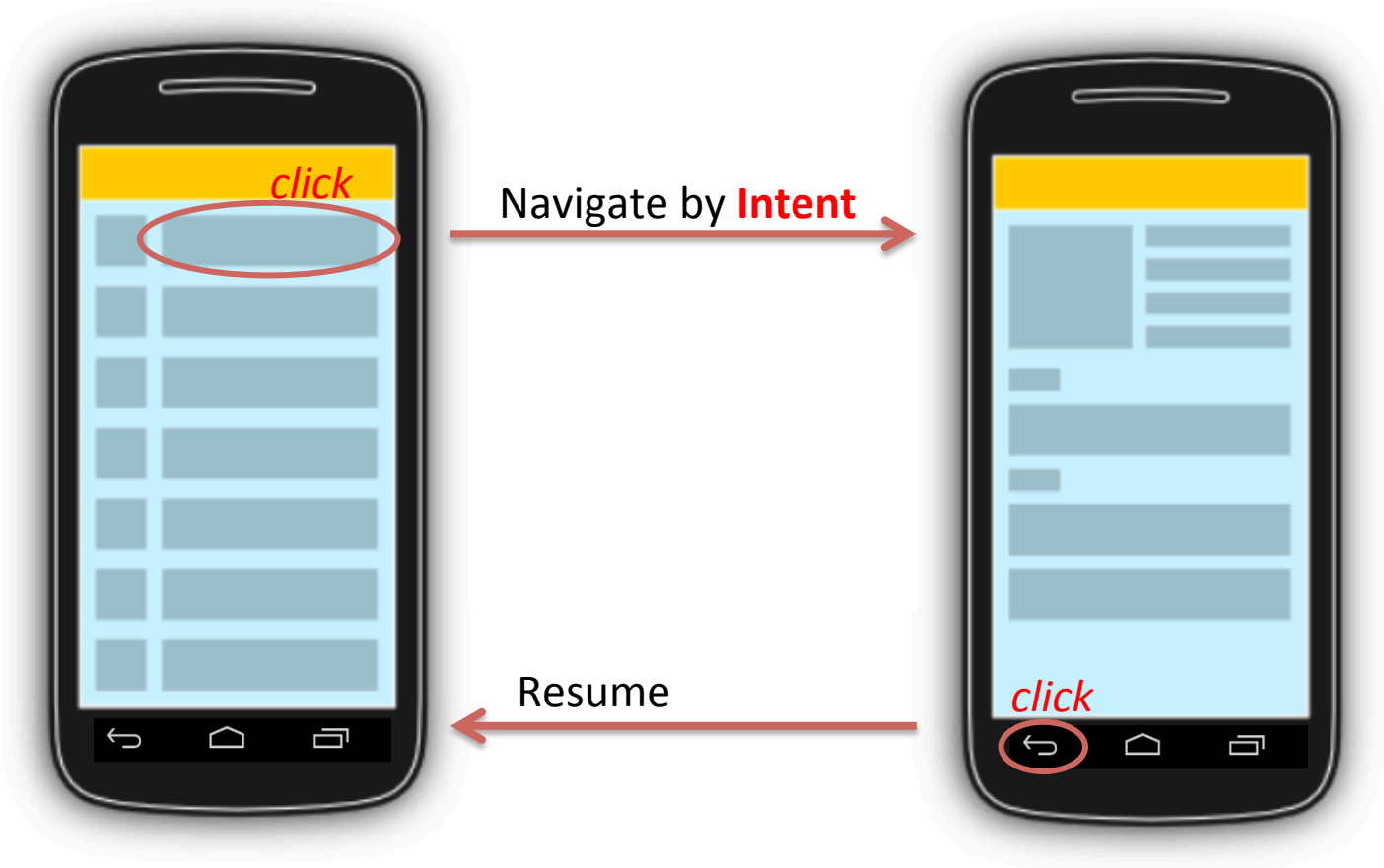
- In layout XML file, indicate handler of a view
  - <Button … android:onClick="clickHandler"/>
- In activity class, define handler method

```
public class MainActivity extends AppCompatActivity {
    @Override
    protected void onCreate(…) {
            super.onCreate(savedInstanceState);
            setContentView(R.layout.activity_main);
    }
    @Override
    public void clickHandler(View v) {
    }
}
```

Android Application Components

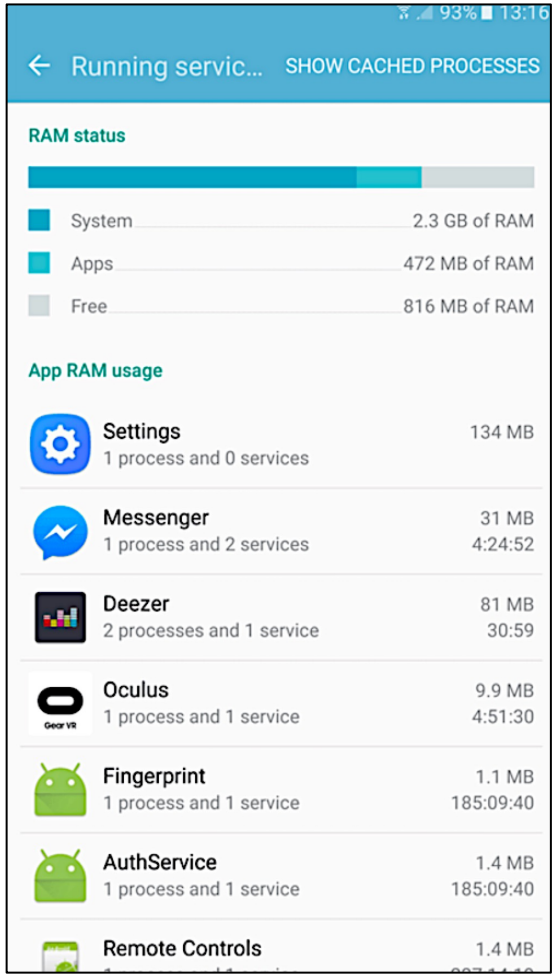
Activity Navigation via Intent

- Activity represents a single screen with UI



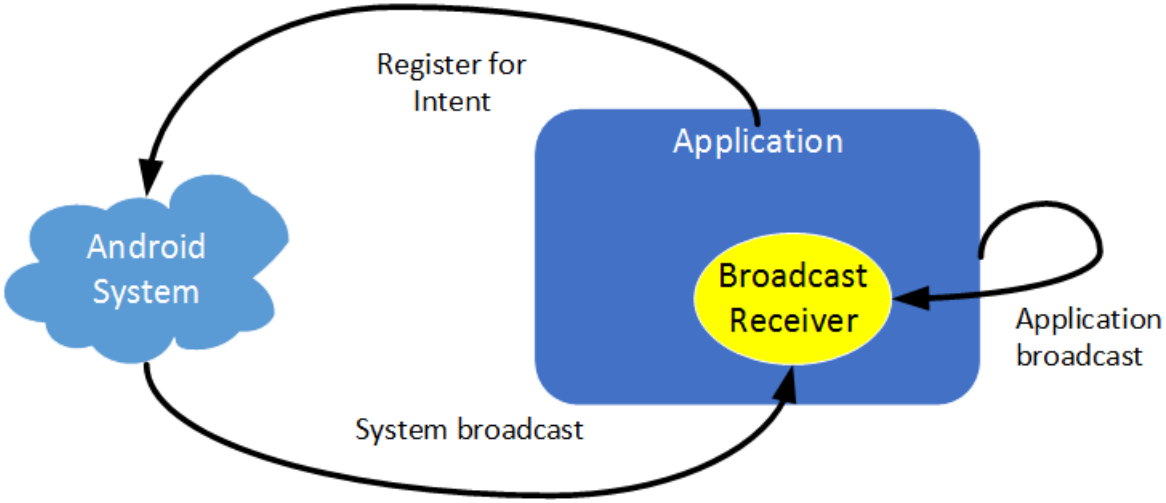
Services

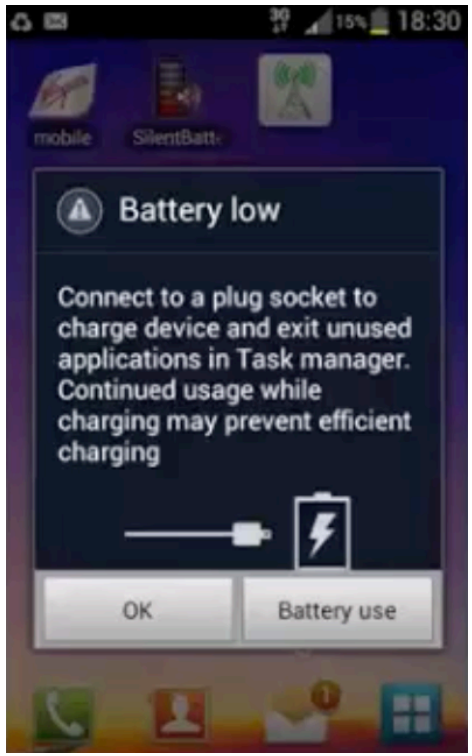
- Background component
- No user interface
- Can be started by an activity
- Examples
  - Play music
  - Fetch data over the internet
  - Alert the user for an event
  - …



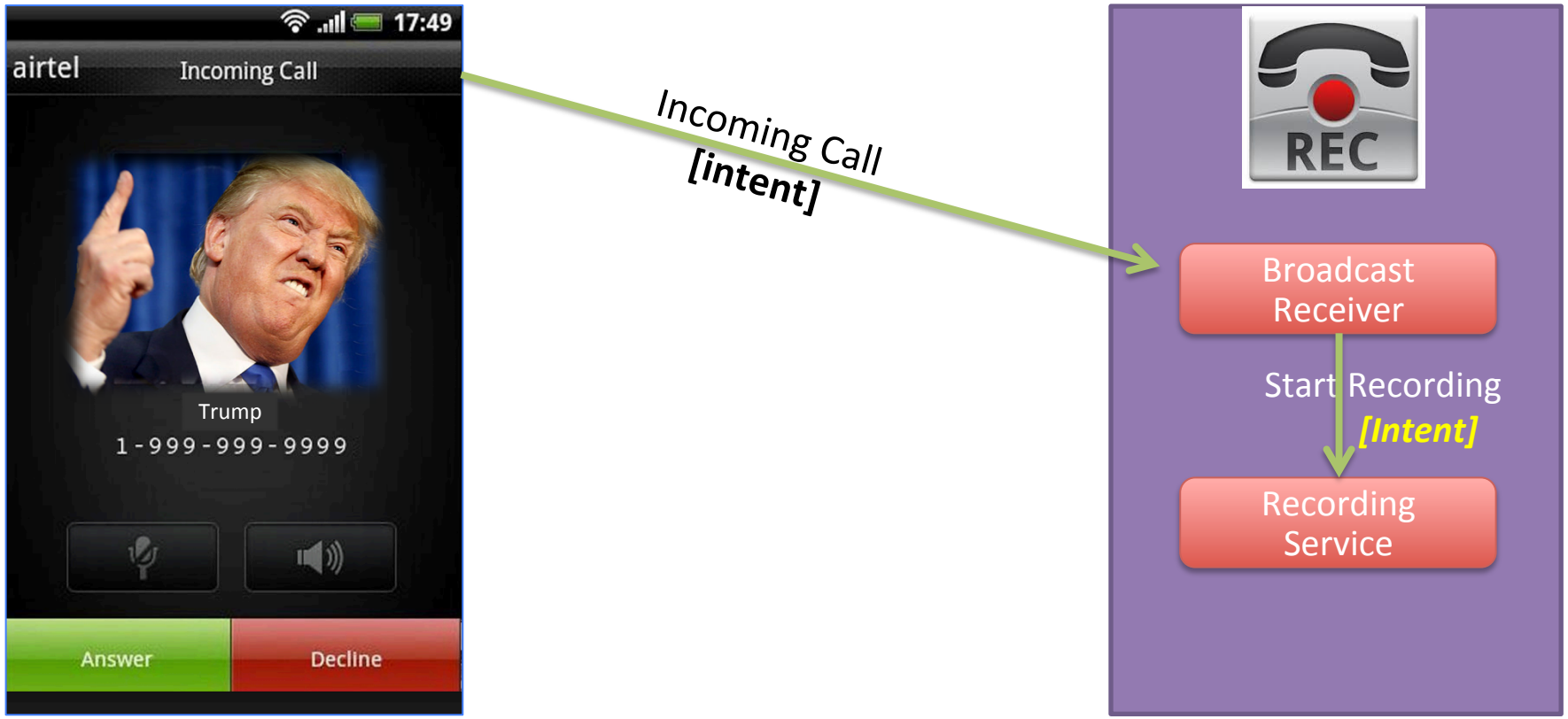
Broadcast Receiver

- Intent-based Publish-Subscribe Mechanism
- System events are broadcast to interested Apps
- App can broadcast, too





B.R. example: Call Recorder




I'm very intent
on camping.
BRAINLESS TALES.com

Intent

- An abstract description of an operation to be performed
- Uses
  - Launch an activity: startActivity(*)
  - Send to broadcast receivers: sendBroadcast(*)
  - Use Services: startService(*), bindService(*)
- Explicit intent
  - Specify target component with full class name
  - Used to interact with same-app components
- Implicit intent
  - Specify the action to be performed
  - Any component from any App can handle it IF intent filter matches
  - Intent filter: condition on intents to receive
  - Multiple matches → user decides

Create new Activity

- Right click on App→New→Activity→Empty Activity
- In Configure Activity, set Activity Name

  →Create class <ActivityName>.java with onCreate()

  →Create layout file activity_@#$.xml

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android=”…"
   xmlns:tools="http://schemas.android.com/tools"
   android:id="@+id/activity_second"
   android:layout_width="match_parent"
   android:layout_height="match_parent” …>
</RelativeLayout>
```

  →Add <activity> element in AndroidManifest.xml

```
<activity android:name=".SecondActivity"></activity>
```

Start an Activity (within an App)

java.lang.Object
↳ android.content.Context
↳ android.content.ContextWrapper
↳ android.view.ContextThemeWrapper
↳ android.app.Activity
↳ android.support.v4.app.FragmentActivity
↳ android.support.v7.app.AppCompatActivity

- Use Explicit Intent
  - New Intent(<Context>, <Activity class>)
    - Activity is a subclass of Context class
    - <Context>: this, <ActivityClass>.this
  - Call startActivity(<intent>)

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

```
OR
startActivity(new Intent(this, SecondActivity.class);
```

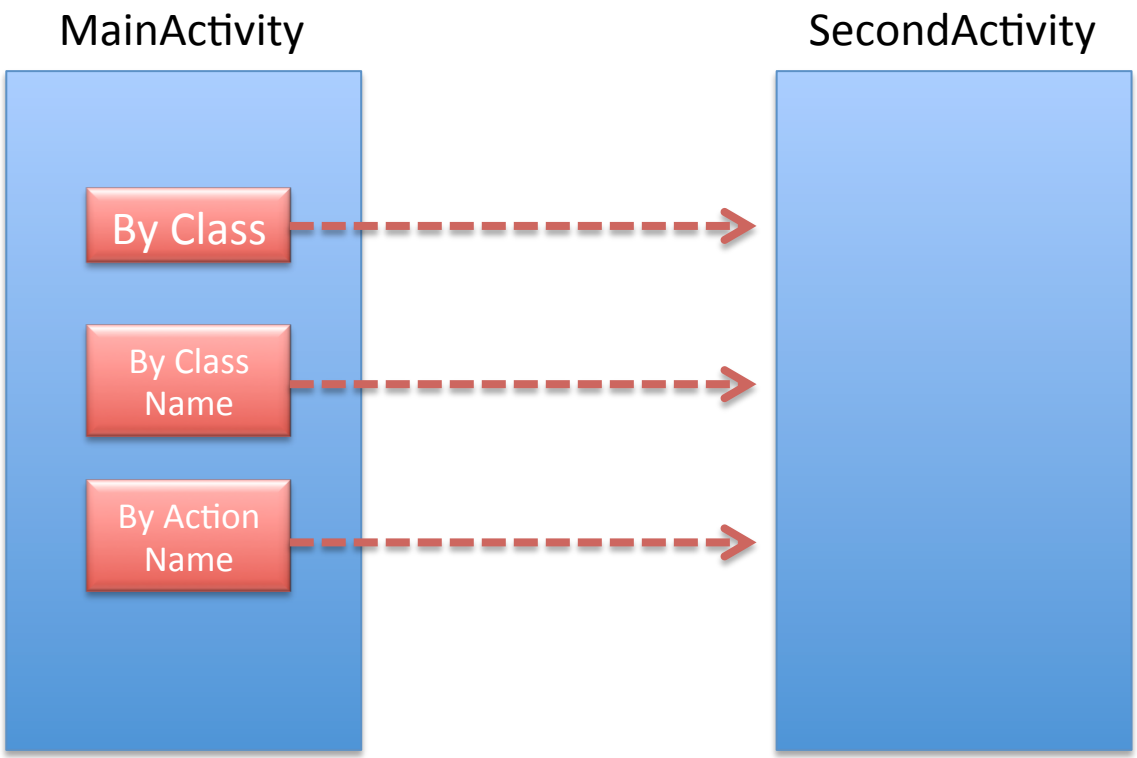
- Adding a button click event (XML method)
  - <Button … android:onClick="*handler_name*">

Starting Activity: Three Methods

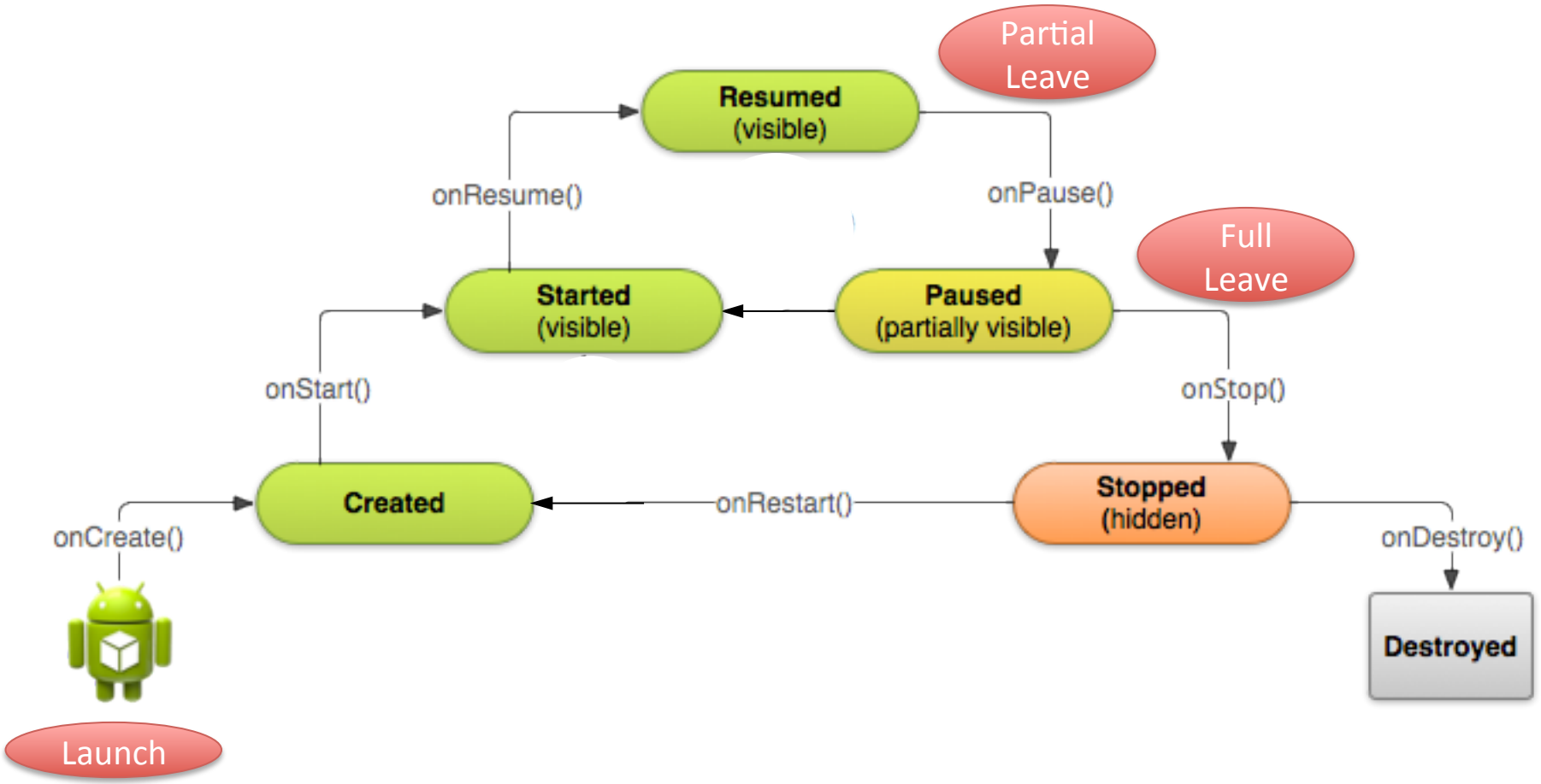
- (1st) Start by class
  - startActivity(new Intent(this, <ActivityName>.class);
- (2nd) Start by class name
  - Intent i = new Intent();
    i.setClassName(<pkg_name>, <full class name>);
    startActivity(i);
- (3rd) Start by Action Name
  - startActivity(new Intent(<action_name>));
  - In AndroidManifest.xml
    - <activity android:name="*ActivityName*">
      <intent-filter>
      <action android:name="*action_name*" />
      <category android:name="android.intent.category.DEFAULT" />
      </intent-filter>
      </activity>
  - Tip: Set action name as <package_name>.<activity_name>

<activity android:name="…">
</activity>

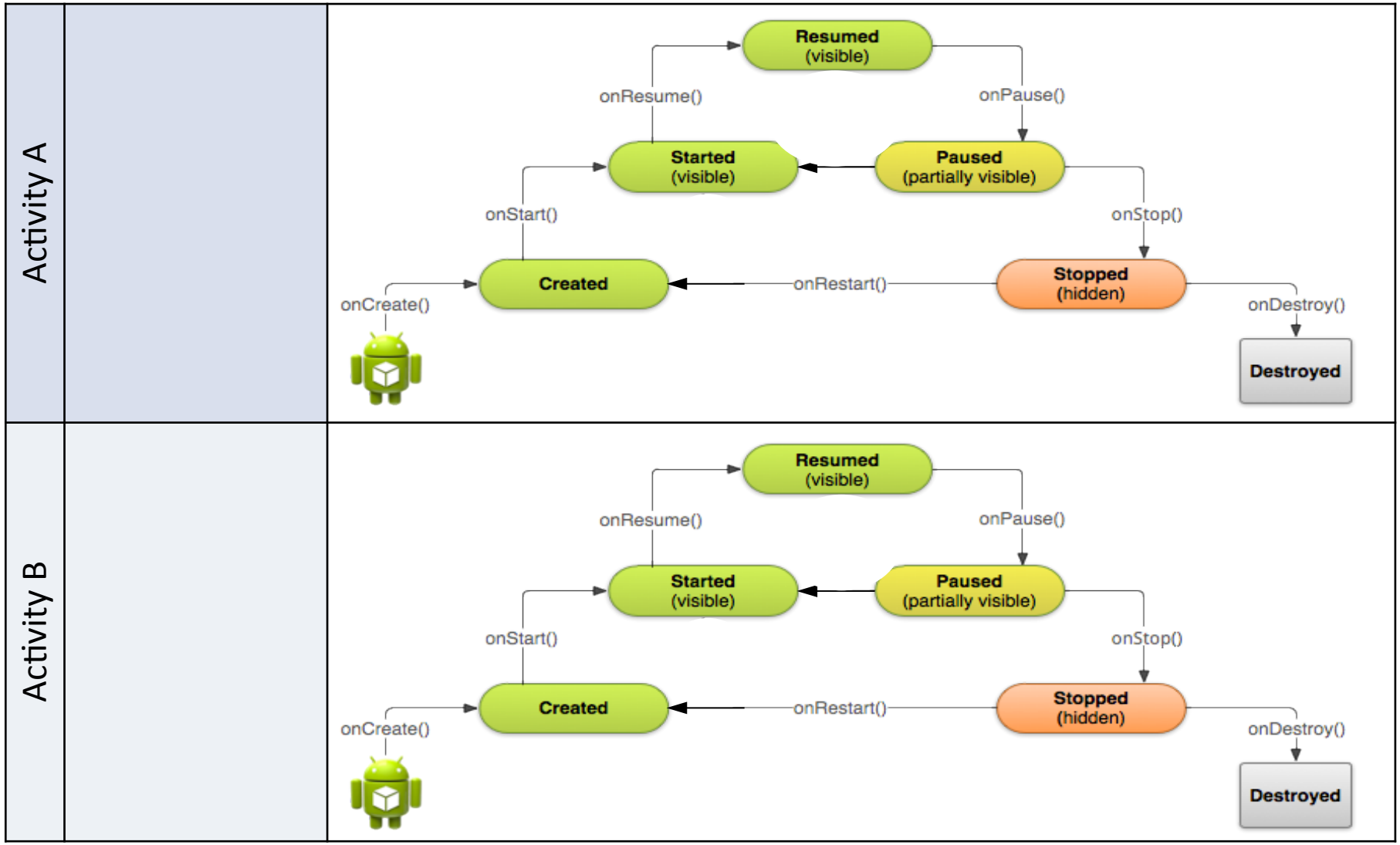
Drill: Start an Activity by 3 methods



Activity Lifecycle



Activity Lifecycle with Two Activities



Showing Toast Message

- Toast
  - A floating small message disappearing soon
  - Toast t=Toast.makeText(<Context>, <msg>, <duration>)
  - t.show();

```
Toast.makeText(this, "Hello World", Toast.LENGTH_SHORT).show();
```

- Toast Positioning
  - t.setGravity(<location>, <x-offset>, <y-offset>)
  - <location> = Gravity.{TOP|BOTTOM|CENTER}
    [| Gravity.{LEFT|RIGHT}]

```
t.setGravity( Gravity.CENTER, 0, 0 )
t.setGravity( Gravity.TOP | Gravity.RIGHT, 0, 0 )
t.setGravity( Gravity.CENTER, 100, 200 )
```

Screen Orientation Handling

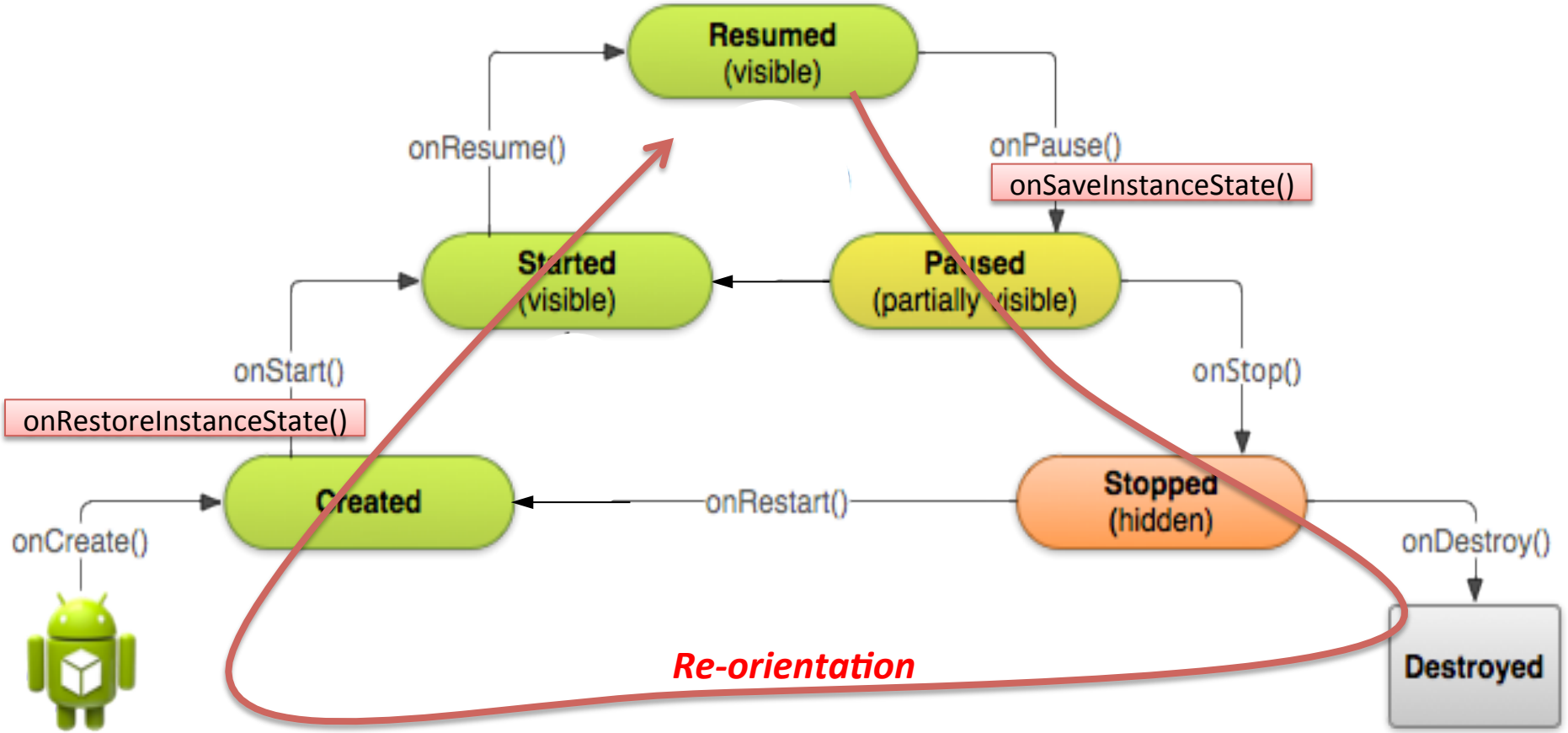
- Orientation
  - Default: both Portrait and Landscape
  - Fix to portrait or landscape: in AndroidManifes.xml
    - <activity .... android:screenOrientation="portrait|landscape">
- Orientation Callback
  - In AndroidManifest.xml
    - <activity … android:configChanges="orientation">
  - In Java
    - **public void** onConfigurationChanged(Configuration newConfig) {
      super.onConfigurationChanged(newConfig);
      if(newConfig.**orientation** == Configuration.***ORIENTATION_PORTRAIT***)
      {…}
      }

Re-orientation & Internal states

- Re-orientation problem
  - Destroy current Activity → Re-create it

  →All internal states will be reset
    - All changes after onResume()
    - All variables' values will be reset
    - All Views' values may be reset (true for old versions)
  - Drill: define variable, change it, toast it
- Solution
  - Save it before destroy: onSaveInstanceState()
  - and Restore it after resume: onRestoreInstanceState()
  - using class Bundle, for saving (key, value) pairs

Activity Lifecycle on Re-orientation



Saving and Restoring States

- In onSaveInstanceState(Bundle outState)
  - outState.put<type>(<key>, <value>)
  - Eg. putString("name", name)
  - Eg. putParcelable(), putSerializable(), …
- In onRestoreInstanceState(Bundle savedState)
  - savedState.get<type>(<key>)
  - Eg. getString("name")
- Drill
  - Make an activity with an EditText, integer var. score, Button for increasing score, Button for toasting score
  - Implement on{Save|Restore}InstanceState() for saving EditText's text and score
  - EditText: getText().toString(), setText(<string>)