CS 4518 Mobile and Ubiquitous Computing

Lecture 4: Data-Driven Views, Android Components & Android Activity Lifecycle

Emmanuel Agu

Announcements

- **Group formation:** Projects 2, 3 and final project will be done in groups
  - Form groups latest today
  - ALL members of the group should email me indicating their group
  - List all team members
  - Student unable to form groups, I will put you in groups
- **Project 1 due tomorrow 11.59PM**
  - Tuesday, January 23, 2018, 11.59PM
  - Test your final submissions in zoolab
  - Submit via InstructAssist!

Data-Driven Layouts

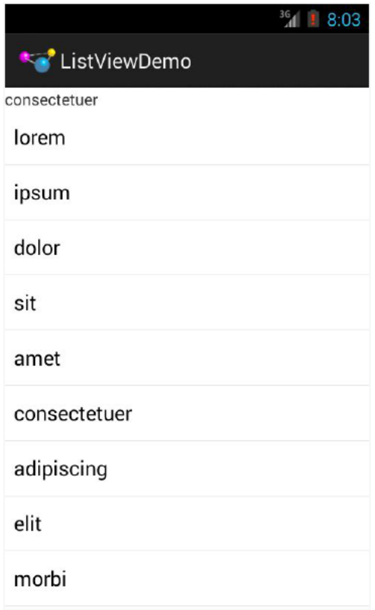
Data-Driven Layouts

- LinearLayout, RelativeLayout, TableLayout, GridLayout useful for positioning UI elements
  - UI data is **hard coded**

- Other layouts dynamically composed from data (e.g. database)
  - ListView, GridView, GalleryView
  - Tabs with TabHost, TabControl

Generate widgets from data source

lorem
ipsum
dolor
amet
consectetuer
adipiscing
elit
morbi

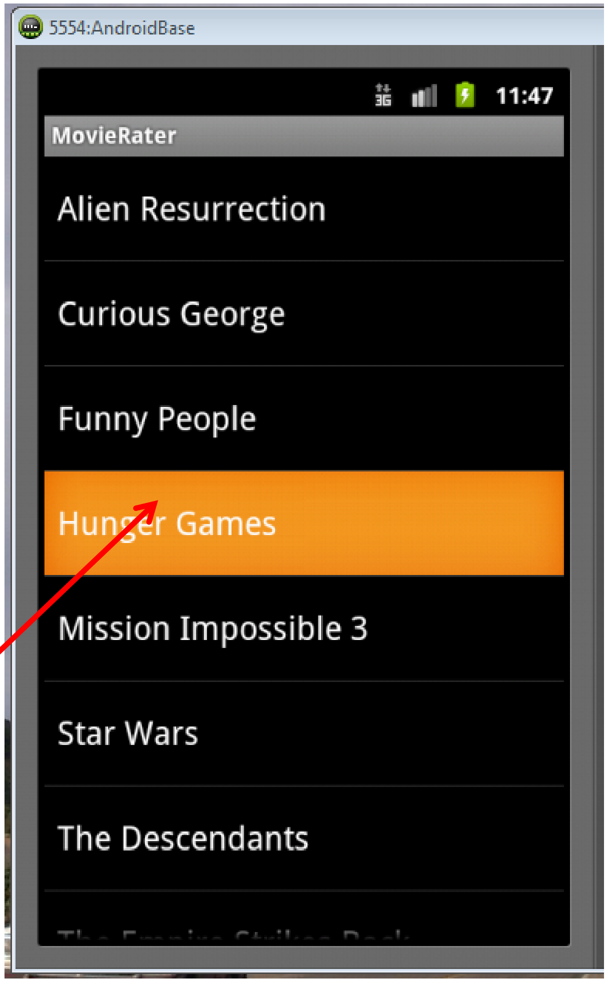


Data Driven Layouts

- May want to populate views from a data source (XML file or database)

- Layouts that display repetitive child Views from data source
  - ListView
  - GridView
  - GalleryView

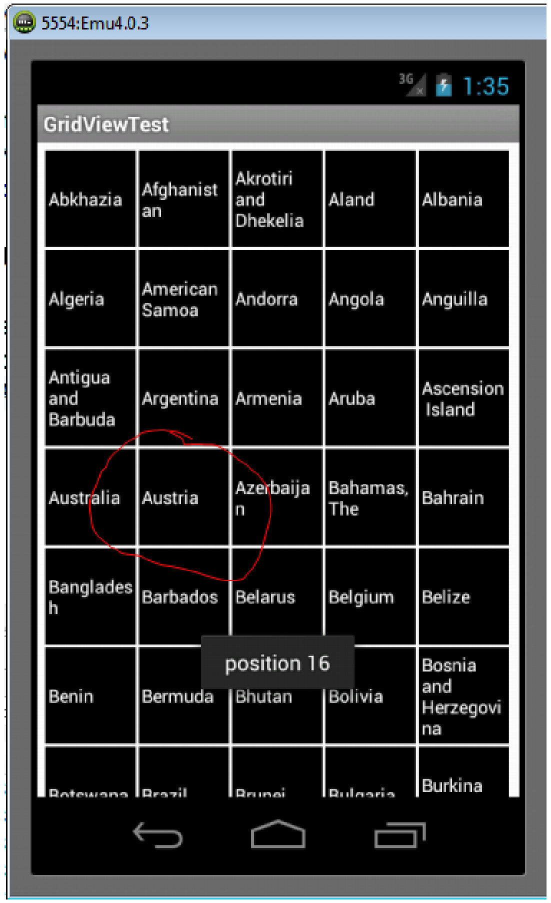
- ListView
  - Rows of entries, pick item, vertical scroll



Alien Resurrection
Curious George
Funny People
Hunger Games
Mission Impossible 3
Star Wars
The Descendants

Data Driven Containers

- GridView
  - List of items arranged in a number of rows and columns
- GalleryView
  - List with horizontal scrolling, typically images

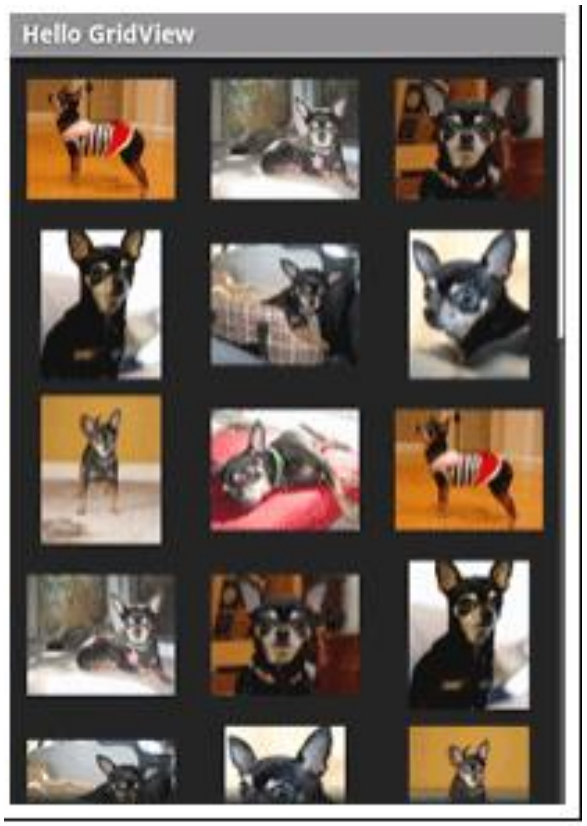

5554:Emu4.0.3

3G 1:35

GridViewTest

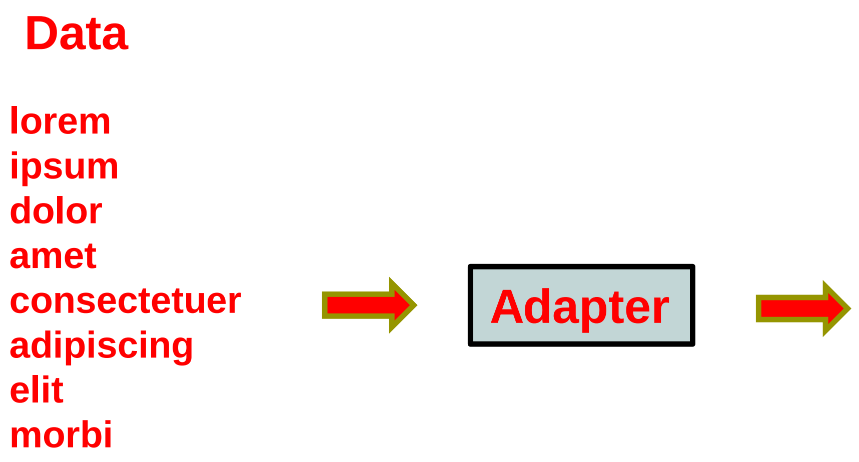
Abkhazia	Afghanistan	Akrotiri and Dhekelia	Aland	Albania
Algeria	American Samoa	Andorra	Angola	Anguilla
Antigua and Barbuda	Argentina	Armenia	Aruba	Ascension Island
Australia	Austria	Azerbaijan	Bahamas, The	Bahrain
Bangladesh	Barbados	Belarus	Belgium	Belize
Benin	Bermuda	Bhutan	Bolivia	Bosnia and Herzegovina
Botswana	Brazil	Brunei	Bulgaria	Burkina

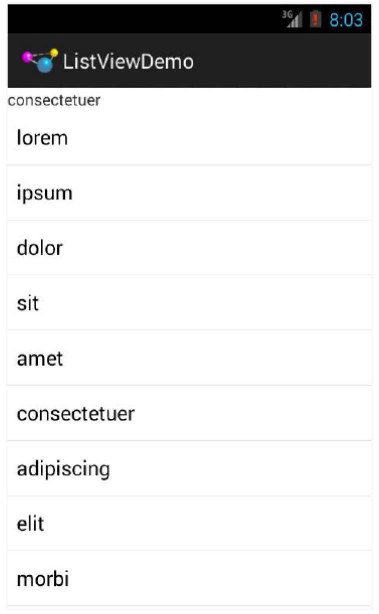
position 16



AdapterView

- ListView, GridView, and GalleryView are sub classes of AdapterView (variants)
- **Adapter:** generates widgets from a data source, populates layout
  - E.g. Data is adapted into cells of GridView





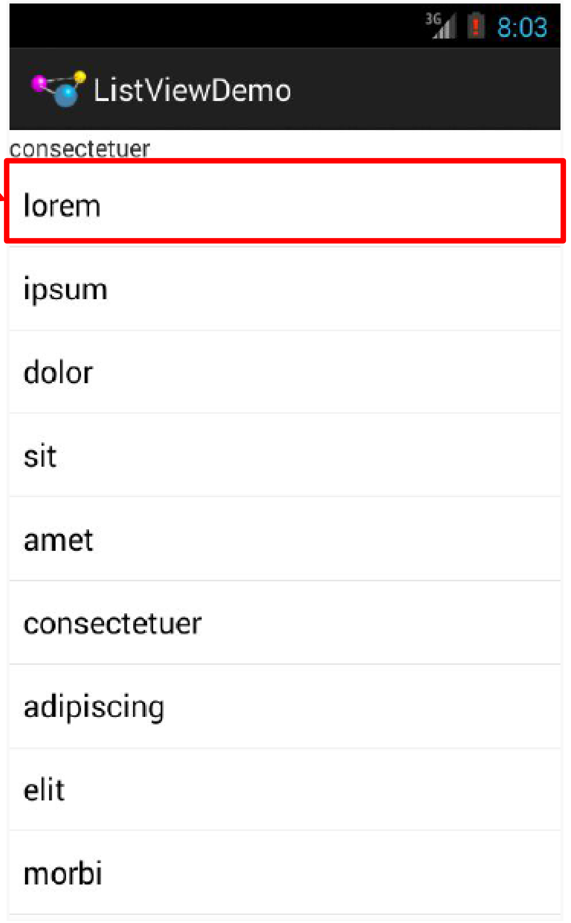
consectetuer

lorem
ipsum
dolor
sit
amet
consectetuer
adipiscing
elit
morbi

- Most common Adapter types:
  - **CursorAdapter:** read from database
  - **ArrayAdapter:** read from resource (e.g. XML file)

Adapters

- When using Adapter, a layout (XML format) is defined for each child element (View)
- The adapter
  - Reads in data (list of items)
  - Creates Views (widgets) using layout for each element in data source
  - Fills the containing layout (List, Grid, Gallery) with the created Views
- Child Views can be as simple as a TextView or more complex layouts / controls
  - simple views can be declared in a layout XML file (e.g. android.R.layout)

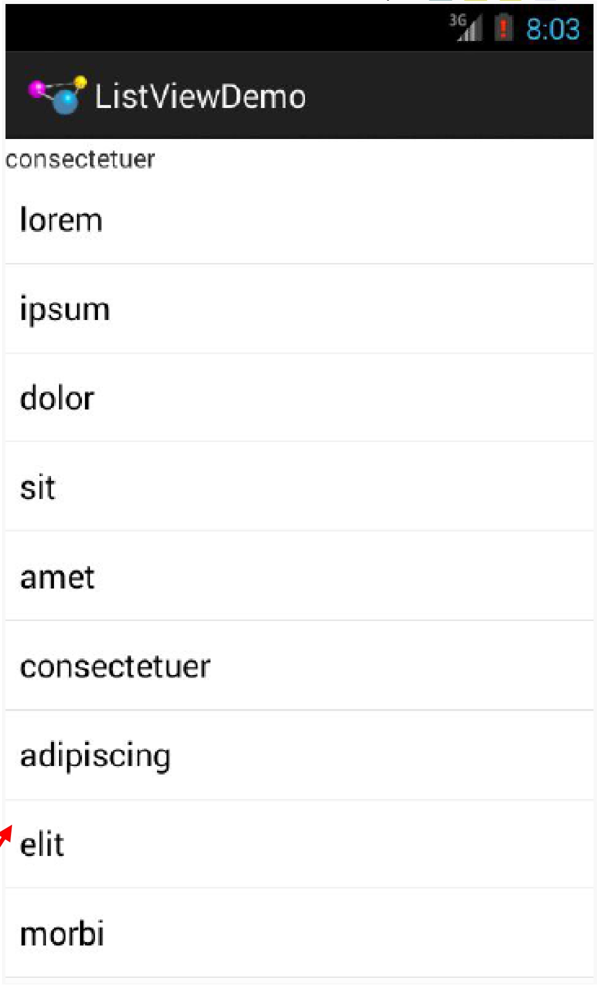


Example: Creating ListView using AdapterArray

- **Task:** Create listView (on right) from strings below

```
private static final String[] items={"lorem", "ipsum", "dolor",
        "sit", "amet",
        "consectetuer", "adipiscing", "elit", "morbi", "vel",
        "ligula", "vitae", "arcu", "aliquet", "mollis",
        "etiam", "vel", "erat", "placerat", "ante",
        "porttitor", "sodales", "pellentesque", "augue", "purus"};
```

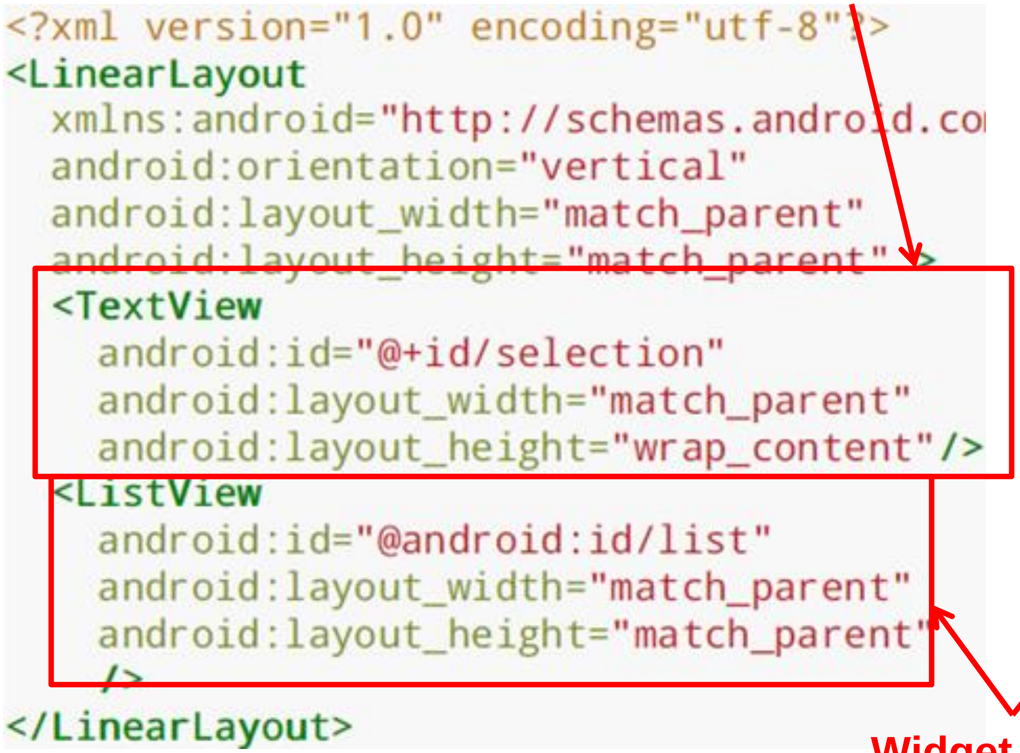
Enumerated list


8:03
ListViewDemo
consectetuer
lorem
ipsum
dolor
sit
amet
consectetuer
adipiscing
elit
morbi

ListView of items

Example: Creating ListView using AdapterArray

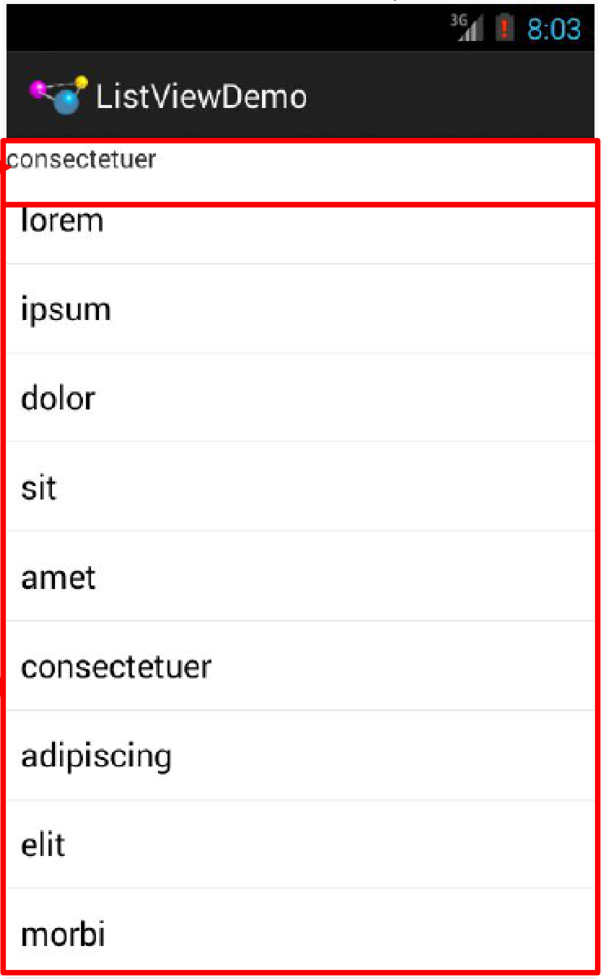
- First create Layout file (e.g. LinearLayout)



```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
  xmlns:android="http://schemas.android.co
  android:orientation="vertical"
  android:layout_width="match_parent"
  android:layout_height="match_parent" >
  <TextView
    android:id="@+id/selection"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"/>
  <ListView
    android:id="@android:id/list"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    />
</LinearLayout>
```

TextView Widget for selected list item

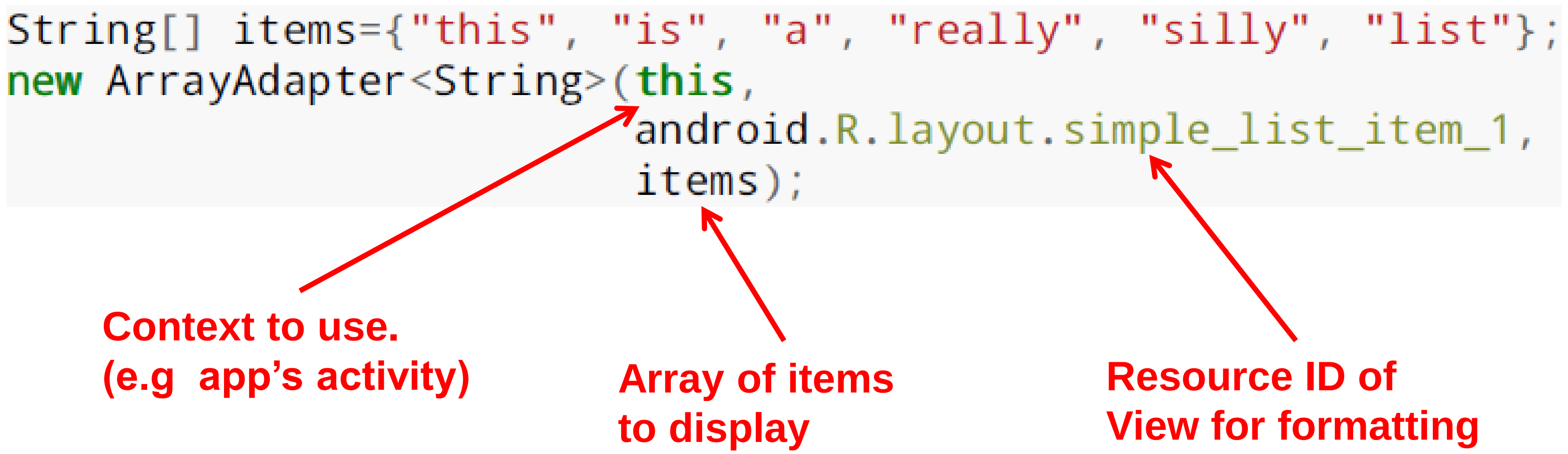
Widget for list of options



consectetuer
lorem
ipsum
dolor
sit
amet
consectetuer
adipiscing
elit
morbi

Using ArrayAdapter

- Command used to wrap adapter around array of menu items or **java.util.List** instance



```
String[] items={"this", "is", "a", "really", "silly", "list"};
new ArrayAdapter<String>(this,
                         android.R.layout.simple_list_item_1,
                         items);
```

- E.g. **android.R.layout.simple_list_item_1** turns strings into textView objects (widgets)

Example: Creating ListView using AdapterArray

```
package com.commonsware.android.list;

import android.app.ListActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.ArrayAdapter;
import android.widget.ListView;
import android.widget.TextView;
public class ListViewDemo extends ListActivity {
  private TextView selection;
  private static final String[] items={"lorem", "ipsum", "dolor",
          "sit", "amet",
          "consectetuer", "adipiscing", "elit", "morbi", "vel",
          "ligula", "vitae", "arcu", "aliquet", "mollis",
          "etiam", "vel", "erat", "placerat", "ante",
          "porttitor", "sodales", "pellentesque", "augue", "purus"};

  @Override
  public void onCreate(Bundle icicle) {
    super.onCreate(icicle);
    setContentView(R.layout.main);
    setListAdapter(new ArrayAdapter<String>(this,
                        android.R.layout.simple_list_item_1,
                        items));
    selection=(TextView)findViewById(R.id.selection);
  }

  @Override
  public void onListItemClick(ListView parent, View v, int position,
                               long id) {
    selection.setText(items[position]);
  }
}
```

Set list adapter (Bridge Data source and views)

Get handle to TextView of Selected item

Change Text at top to that of selected view when user clicks on selection

Android App Components

Android App Components

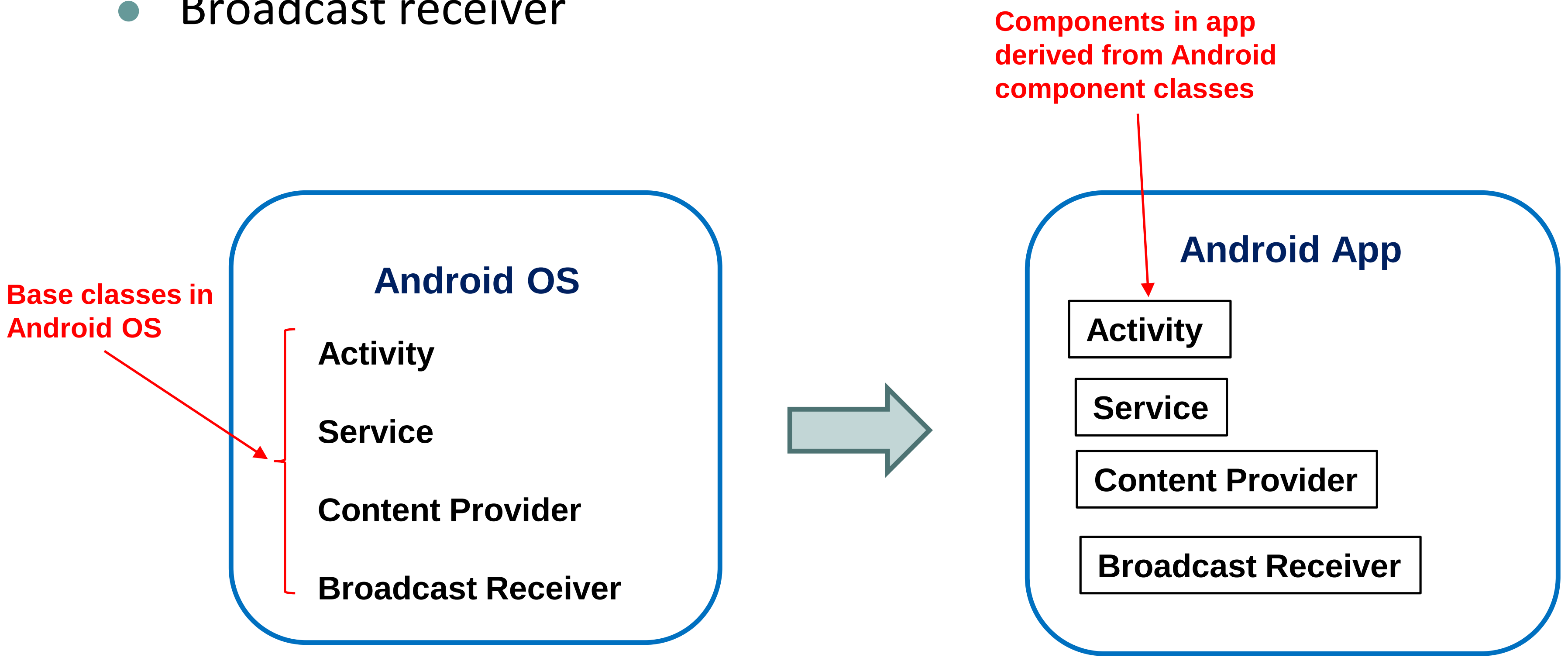
- Typical Java program starts from main( )

```
class SillyApp {
    public static void main(String[] args) {
        System.out.println("Hello World!");
    }
}
```

- Android app: No need to write a main
- Just define app components derived from base classes already defined in Android

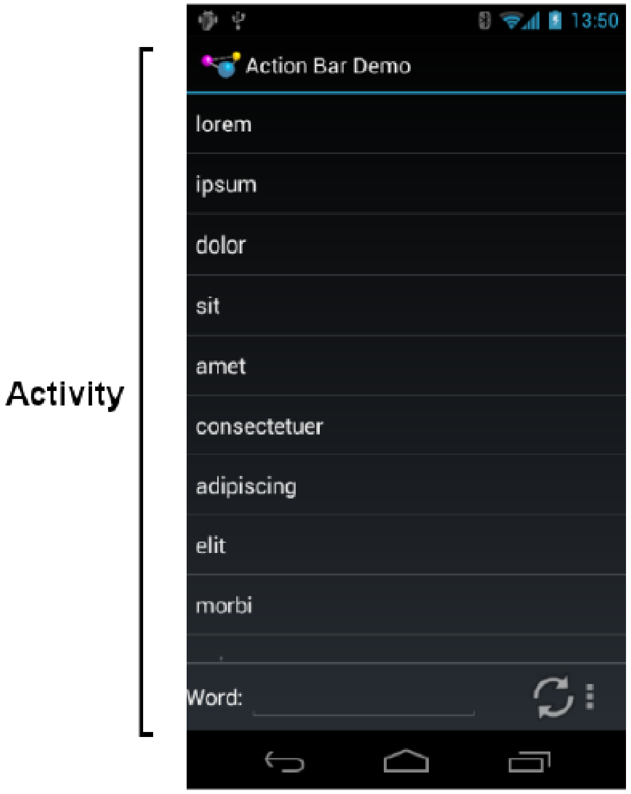
Android App Components

- 4 main types of Android app components:
  - Activity (already seen this)
  - Service
  - Content provider
  - Broadcast receiver



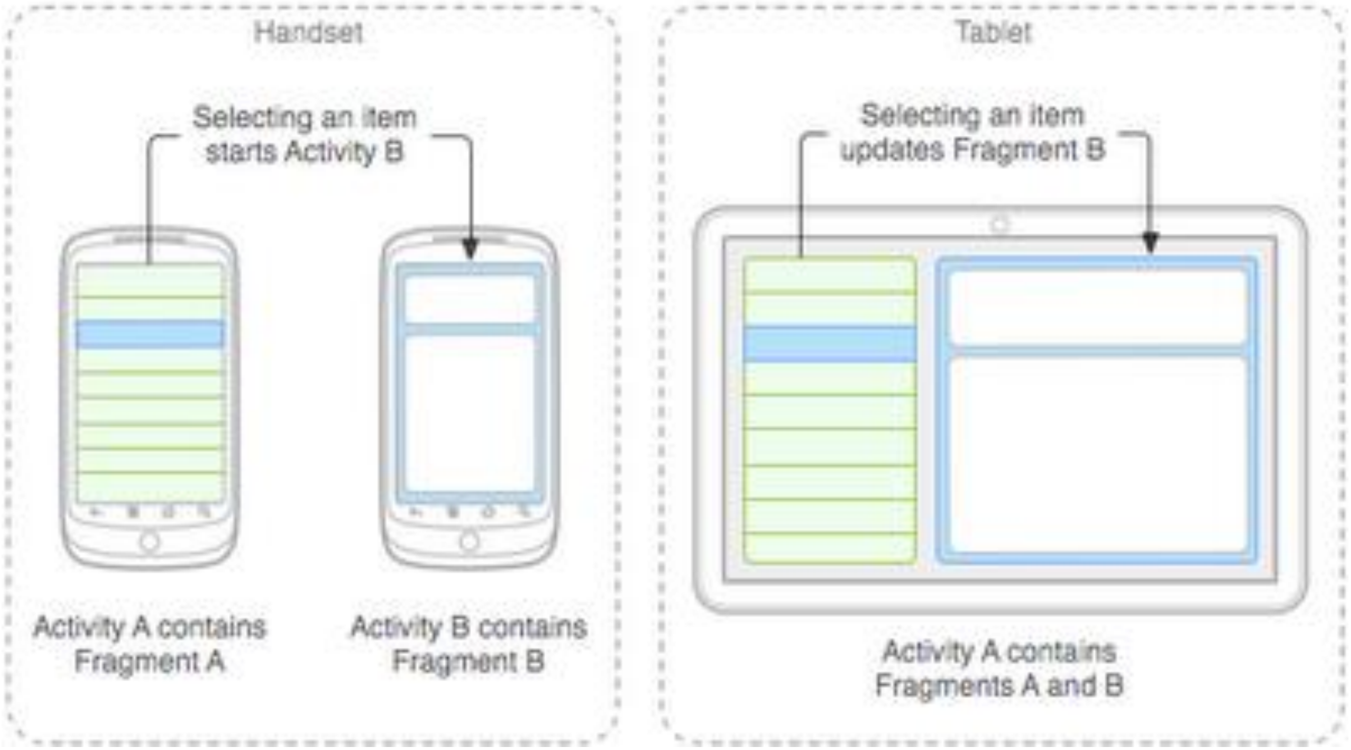
Recall: Activities

- Activity: main building block of Android UI
- Analogous to a window or dialog box in a desktop application
- Apps
  - have at least 1 activity that deals with UI
  - Entry point of app similar to **main( )** in C
  - typically have multiple activities
- Example: A camera app
  - **Activity 1:** to focus, take photo, start activity 2
  - **Activity 2:** to present photo for viewing, save it



Fragments

- Fragments
  - UI building blocks (pieces), can be arranged in Activities in different ways.
  - Enables app to look different on different devices (e.g. phone vs tablet)
- An activity can contain multiple fragments that are organized differently for phone vs tablet
- More later


Handset
Selecting an item starts Activity B
Activity A contains Fragment A
Activity B contains Fragment B
Tablet
Selecting an item updates Fragment B
Activity A contains Fragments A and B

Services

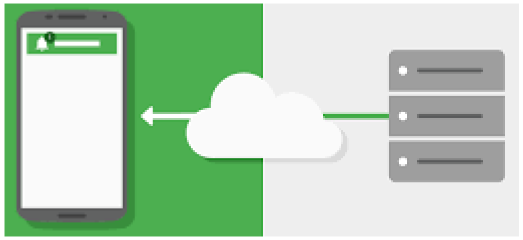
- Activities are short-lived, can be shut down anytime (e.g when user presses back button)
- Services keep running in background
- Similar to Linux/Unix CRON job
- Example uses of services:
  - Periodically check device's GPS location
  - Check for updates to RSS feed
- Minimal interaction with (independent of) any activity
- Typically an activity will control a service -- start it, pause it, get data from it
- App Services are sub-class of **Services** class

Android Platform Services

- Android Services can either be on:
  - Android Platform (local, on smartphone)
  - Google (remote, in Google server)

- Android platform services examples (on smartphone):
  - **LocationManager:** location-based services.
  - **ClipboardManager:** access to device's clipboard, cut-and-paste content
  - **DownloadManager:** manages HTTP downloads in background
  - **FragmentManager:** manages the fragments of an activity.
  - **AudioManager:** provides access to audio and ringer controls.

Android services
on smartphone

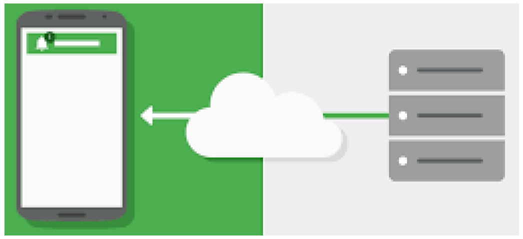


Android services
In Google cloud

Google Services (In Google Cloud)

- Maps
- Location-based services
- Game Services
- Authorization APIs
- Google Plus
- Play Services
- In-app Billing
- Google Cloud Messaging
- Google Analytics
- Google AdMob ads

Typically need Internet connection

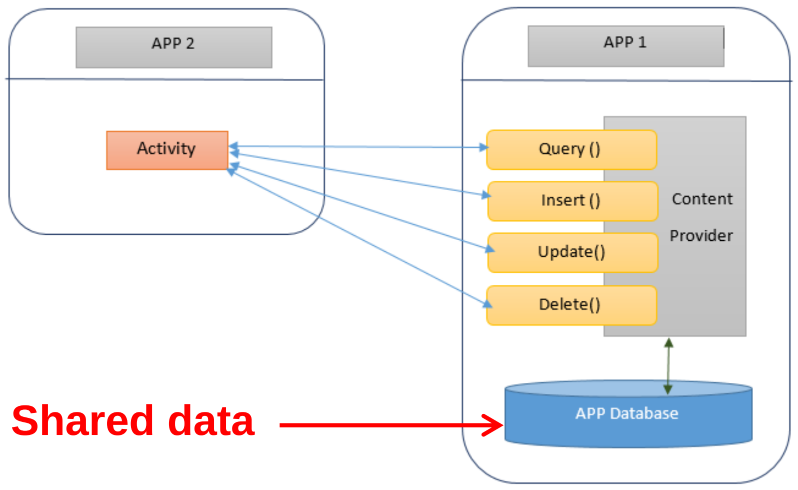


Android services on smartphone

Android services In Google cloud

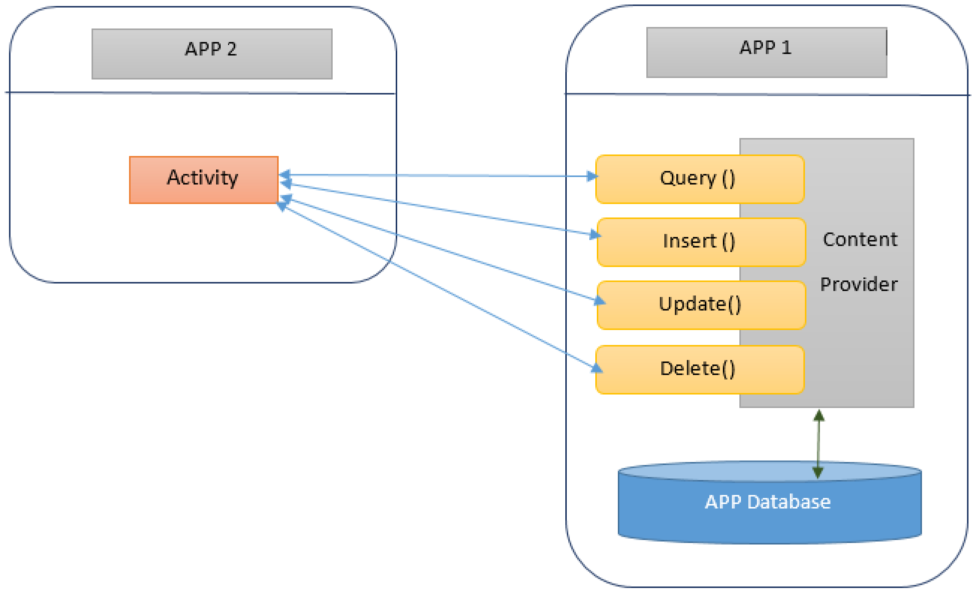
Content Providers

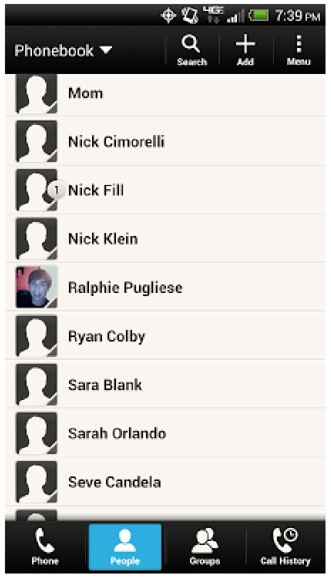
- Android apps can share data (e.g. User's contacts) as content provider
- Content Provider:
  - Abstracts shareable data, makes it accessible through methods
  - Applications can access that shared data by calling methods for the relevant **content provider**
  - E.g. Can query, insert, update, delete shared data (see below)



Content Providers

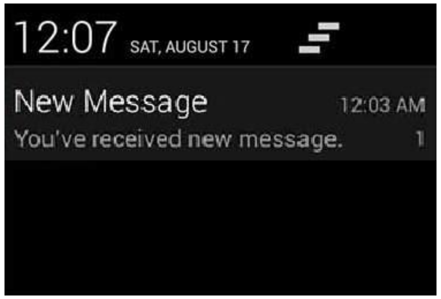
- **E.g.** Data stored in Android Contacts app can be accessed by other apps
- **Example:** We can write an app that:
  - Retrieve's contacts list from contacts content provider
  - Adds contacts to social networking (e.g. Facebook)
- Apps can also **ADD** to data through content provider. E.g. Add contact
- E.g. Our app can also share its data
- App Content Providers are sub-class of **ContentProvider** class





Broadcast Receivers

- The system, or applications, periodically *broadcasts* events
- Example broadcasts:
  - Battery getting low
  - Download completed
  - New email arrived
- Any app can create broadcast receiver to listen for broadcasts, respond
- Our app can also initiate broadcasts
- Broadcast receivers typically
  - Doesn't interact with the UI
  - Creates a status bar notification to alert the user when broadcast event occurs
- App Broadcast Receivers are sub-class of **BroadcastReceiver** class



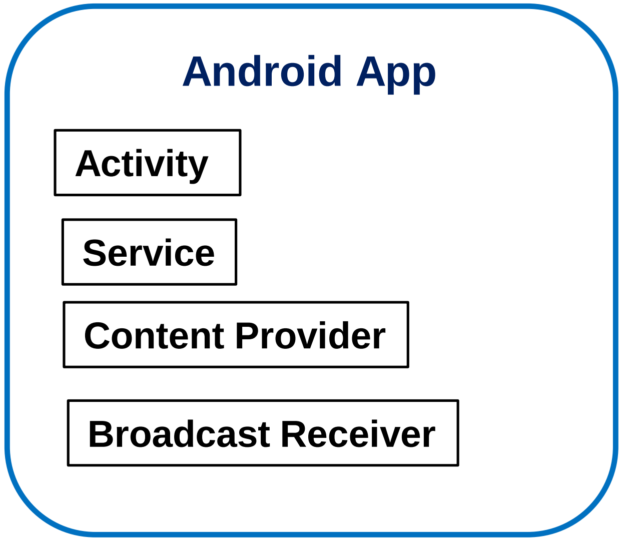
12:07 SAT, AUGUST 17

New Message 12:03 AM
You've received new message. 1

Quiz

- Pedometer App
  - **Component A:** continously counts user's steps even when user closes app, does other things on phone (e.g. youtube, calls)
  - **Component B:** Displays user's step count
  - **Component C:** texts user's friends (from contacts list) every day with their step totals

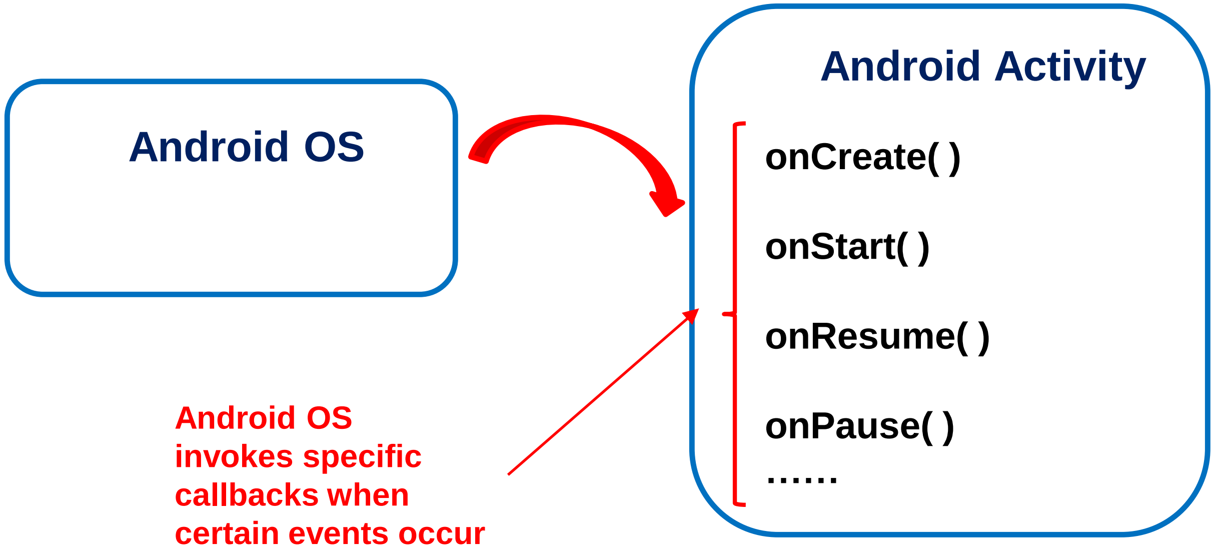
- What should component A be declared as (Activity, service, content provider, broadcast receiver)
- What of component B?
- Component C?



Android Activity LifeCycle

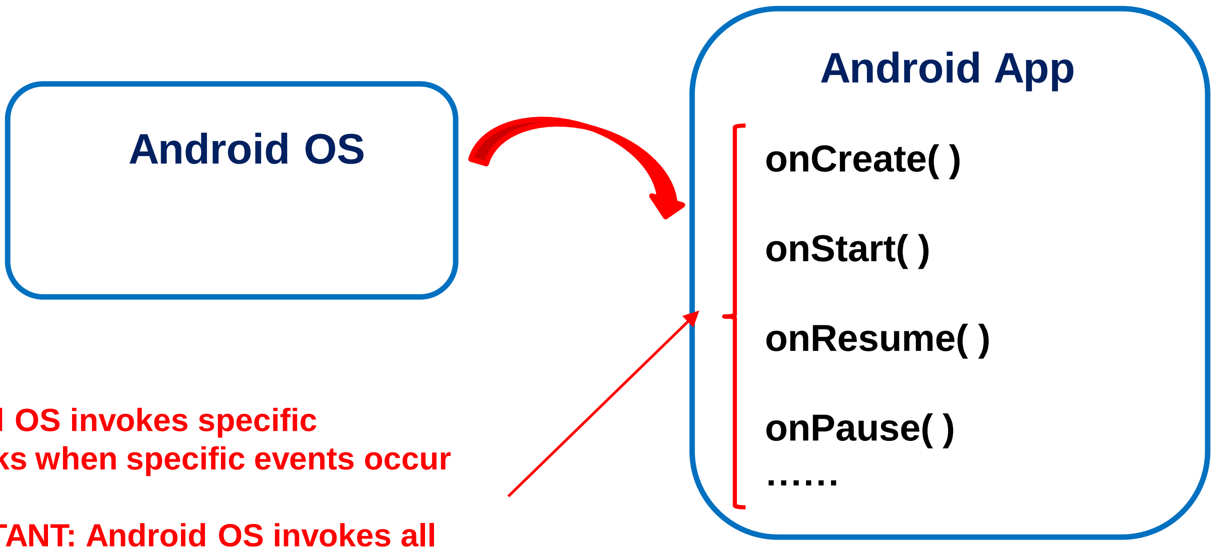
Starting Activities

- Android Activity callbacks invoked corresponding to app state.
- Examples:
  - When activity is created, its **onCreate( )** method invoked (like constructor)
  - When activity is paused, its **onPause( )** method invoked
- Callback methods also invoked to destroy Activity /app



Activity Callbacks

- onCreate() ← **Already saw this (initially called)**
- onStart()
- onResume()
- onPause()
- onStop()
- onRestart()
- onDestroy()



Android OS invokes specific callbacks when specific events occur

IMPORTANT: Android OS invokes all callbacks!!

Understanding Android Lifecycle

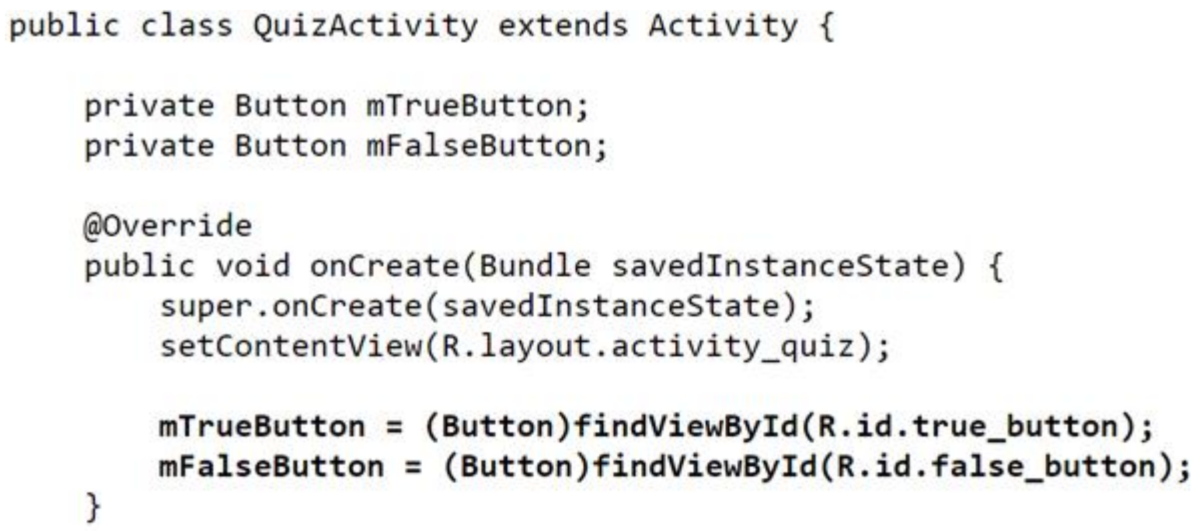
- Many **disruptive** things could happen while app is running
  - Incoming call or text message, user switches to another app, etc

- Well designed app should NOT:
  - Crash if interrupted, or user switches to other app
  - Lose the user's state/progress (e.g state of chess game app) if they leave your app and return later
  - Crash or lose the user's progress when the screen rotates between landscape and portrait orientation.
    - E.g. Youtube video should continue at correct point after rotation
- To handle these situations, appropriate callback methods must be invoked appropriately to “tidy up” before app gets bumped

https://developer.android.com/guide/components/activities/activity-lifecycle.html

OnCreate( )

- Initializes activity once created
- Operations typically performed in onCreate() method:
    - Inflate widgets and place them on screen
        - (e.g. using layout files with setContentView( ))
    - Getting references to inflated widgets (using findViewbyId( ))
    - Setting widget listeners to handle user interaction
- E.g.

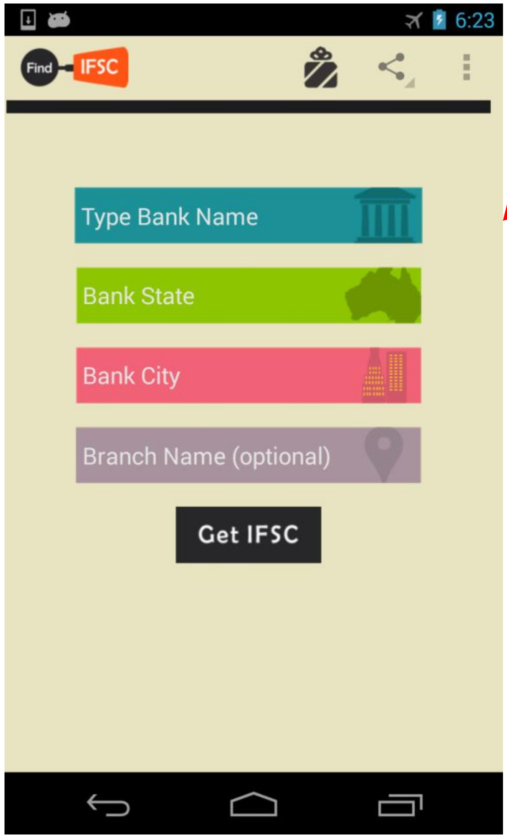


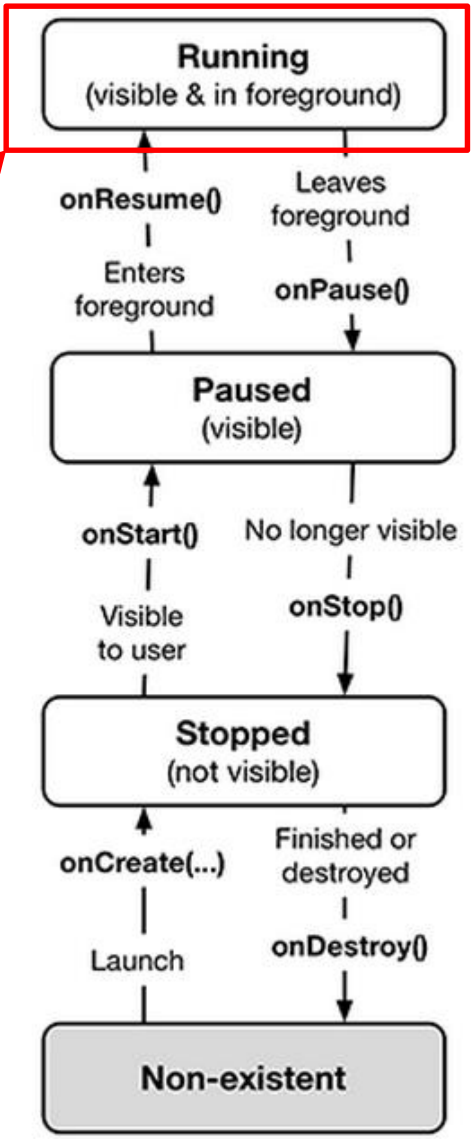
```
public class QuizActivity extends Activity {

    private Button mTrueButton;
    private Button mFalseButton;

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_quiz);

        mTrueButton = (Button)findViewById(R.id.true_button);
        mFalseButton = (Button)findViewById(R.id.false_button);
    }
```

- **Note:** Android OS calls apps' onCreate( ) method

Running App

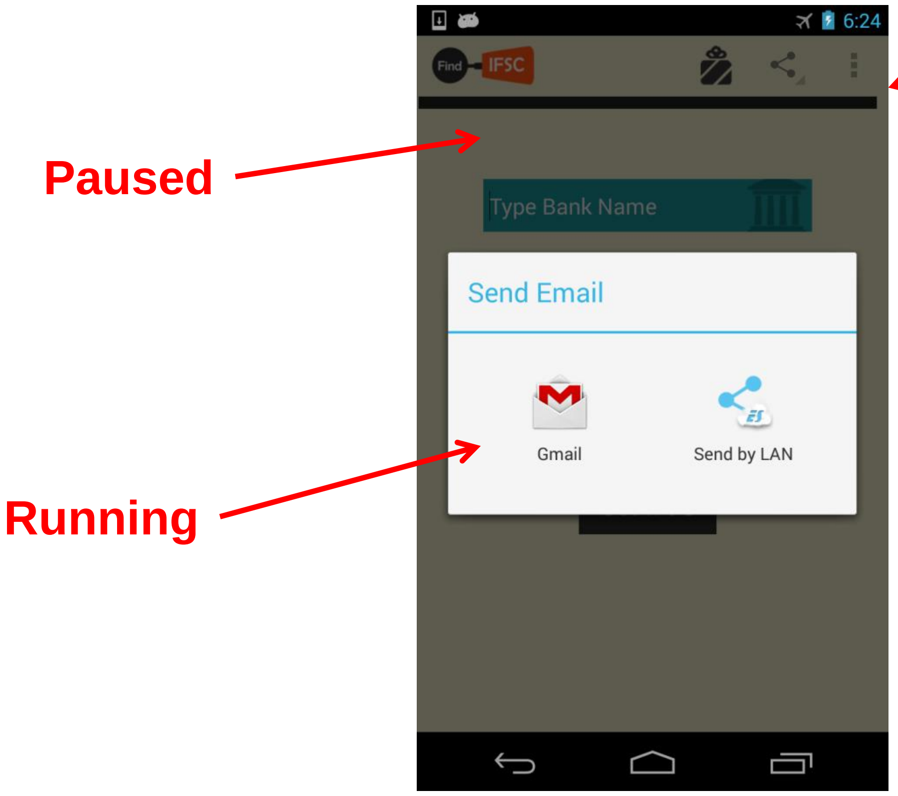
- A running app is one that user is currently using or interacting with
  - Visible, in foreground

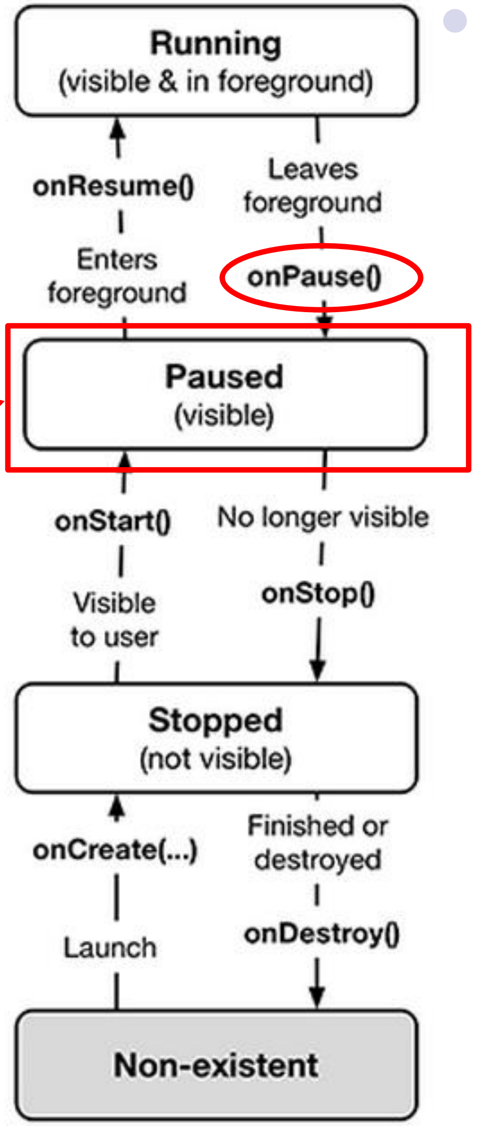




Paused App

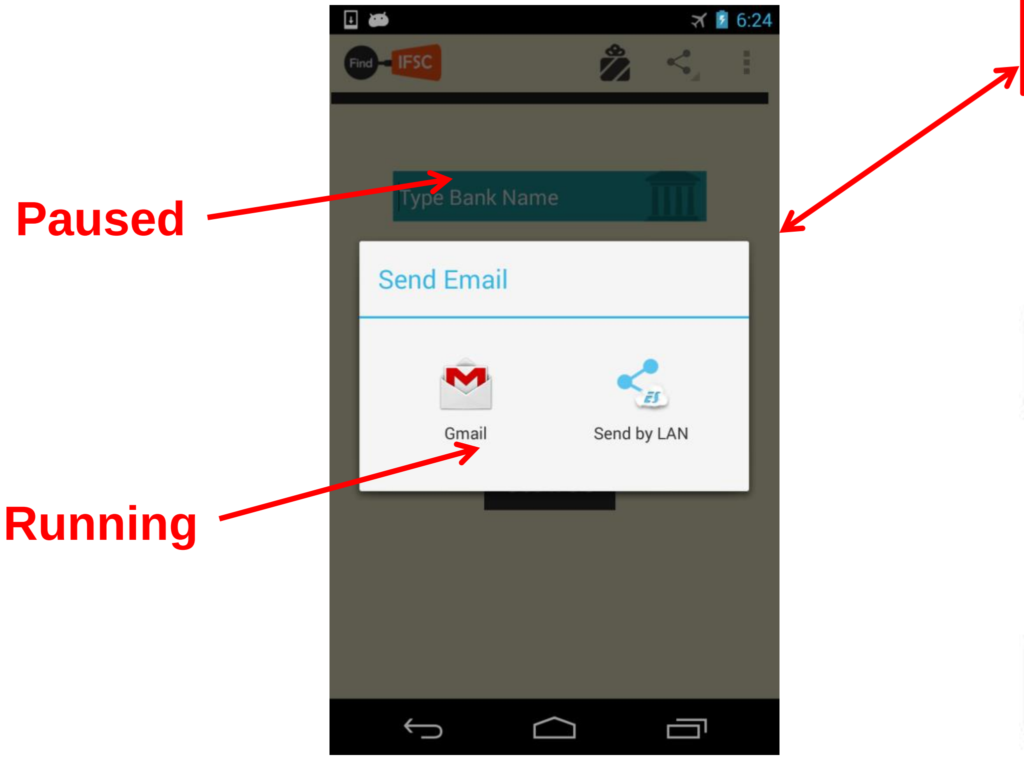
- An app is **paused** if it is **visible but no longer in foreground**
- E.g. blocked by a pop-up dialog box
- App's **onPause( )** method is called during transition from running to paused state

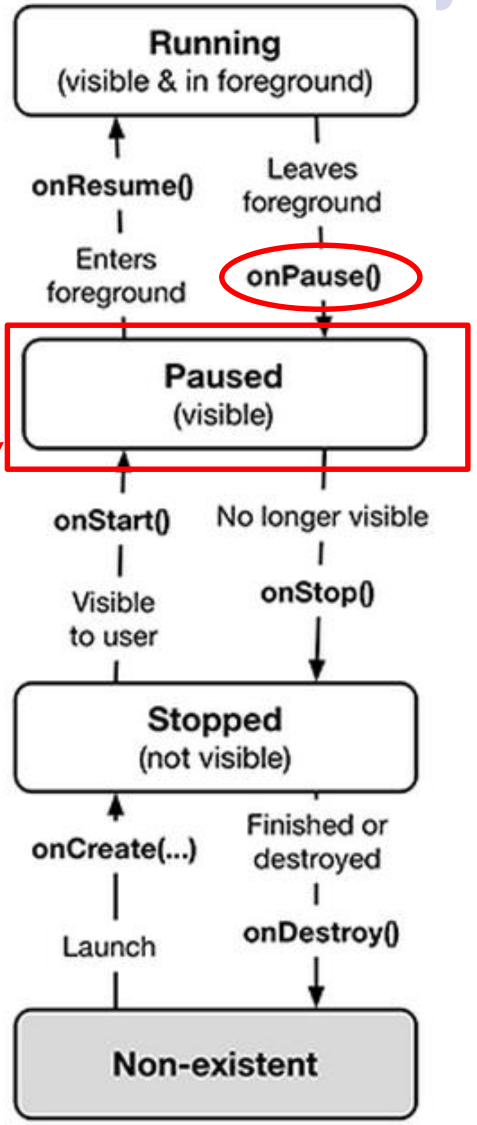




onPause( ) Method

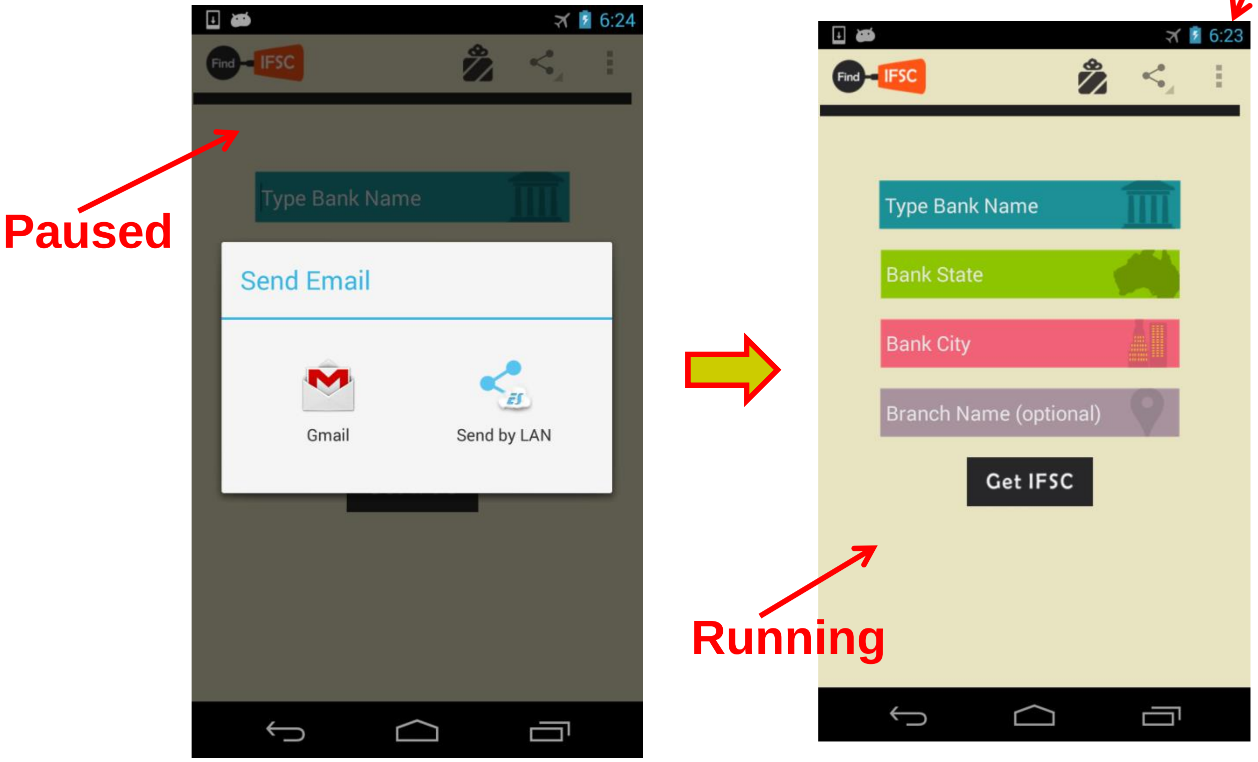
- Typical actions taken in onPause( ) method
  - Stop animations or CPU intensive tasks
  - Stop listening for GPS, broadcast information
  - Release handles to sensors (e.g GPS, camera)
  - Stop audio and video if appropriate

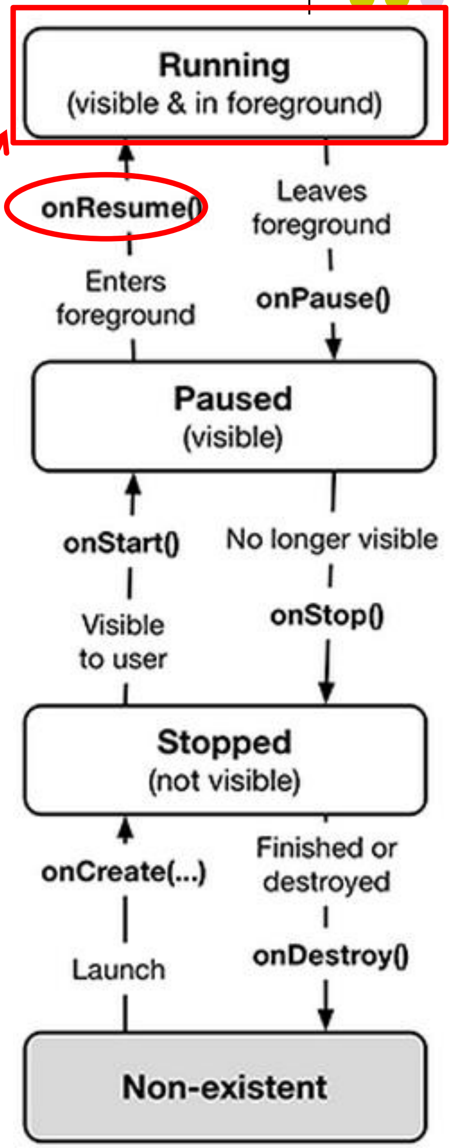




onResume( ): Resuming Paused App

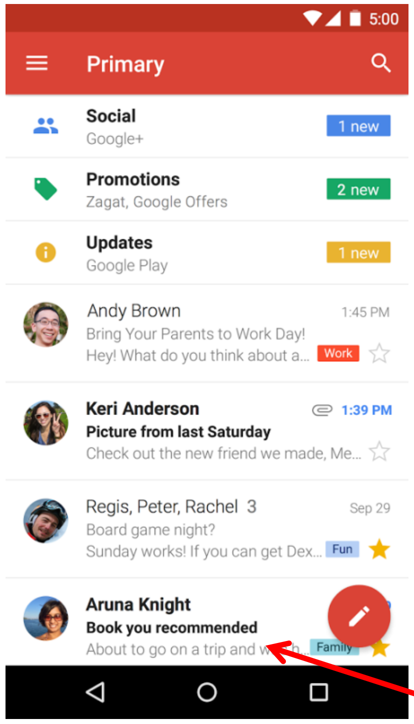
- A **paused** app resumes **running** if it becomes fully visible and in foreground
  - E.g. pop-up dialog box blocking it goes away
- App's **onResume( )** method is called during transition from **paused** to **running** state
  - Restart videos, animations, GPS checking, etc

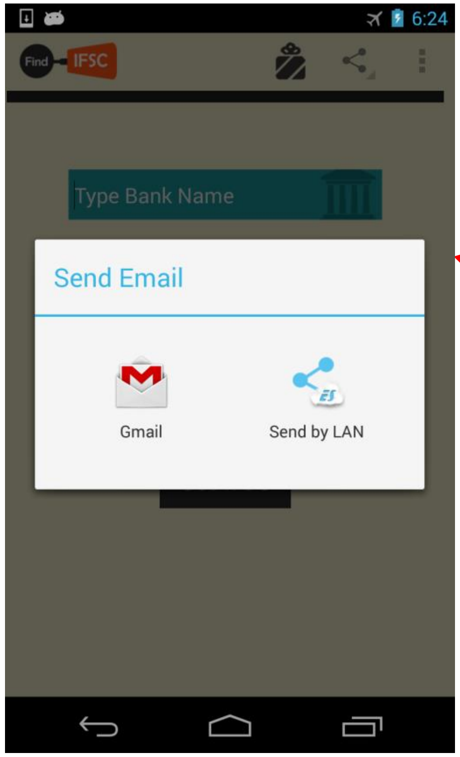




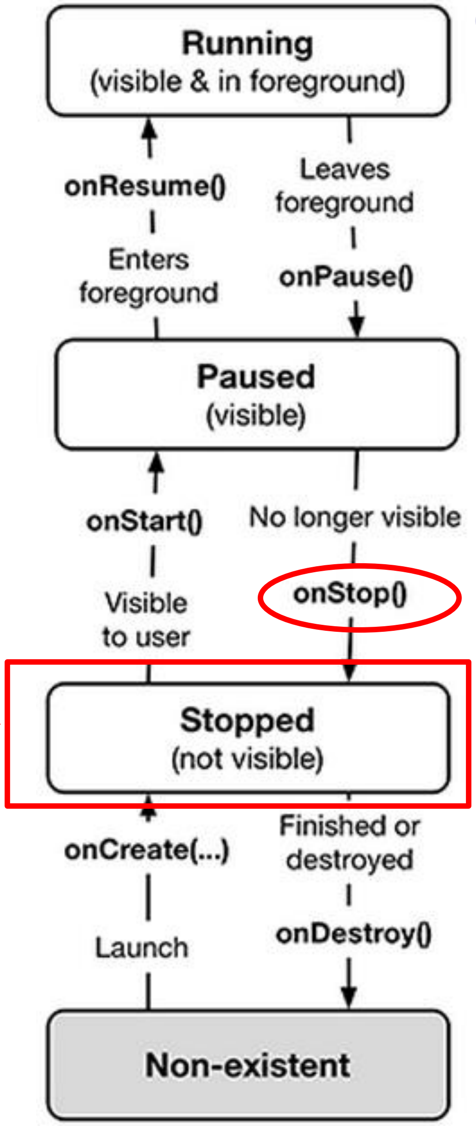
Stopped App

- An app is **stopped** if it **no longer visible + no longer in foreground**
- E.g. user starts using another app
- App's **onStop( )** method is called during transition from paused to stopped state





Running

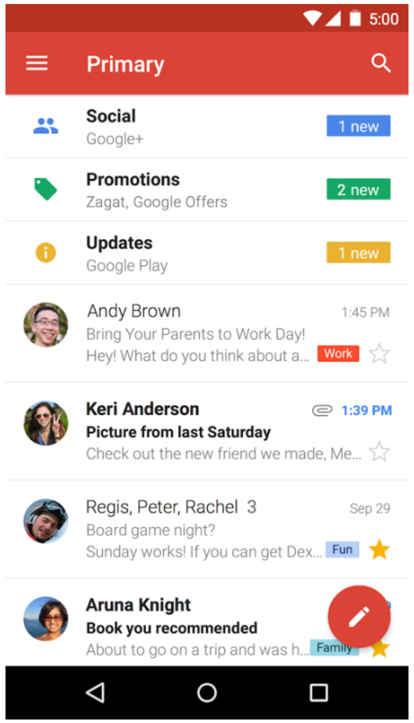


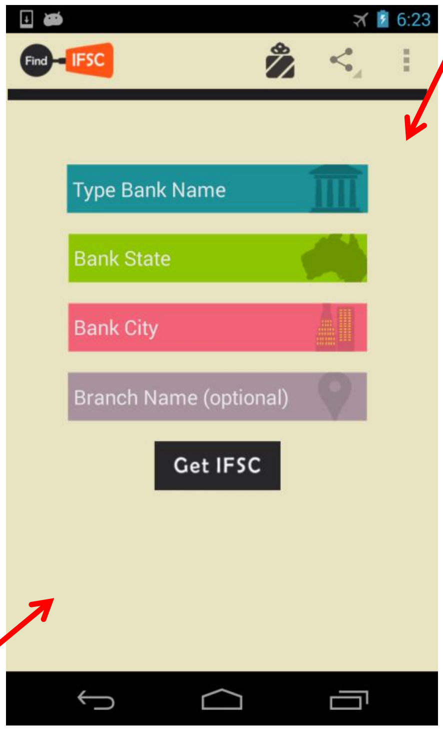
onStop() Method

- An activity is stopped when:
  - User receives phone call
  - User starts another app
  - Activity 1 launches new Activity 2
- Activity instance and variables of stopped app are retained but no code is being executed by the activity
- If activity is stopped, in onStop( ) method, well behaved apps should
  - save progress to enable seamless restart later
  - Release all resources, save info (persistence)

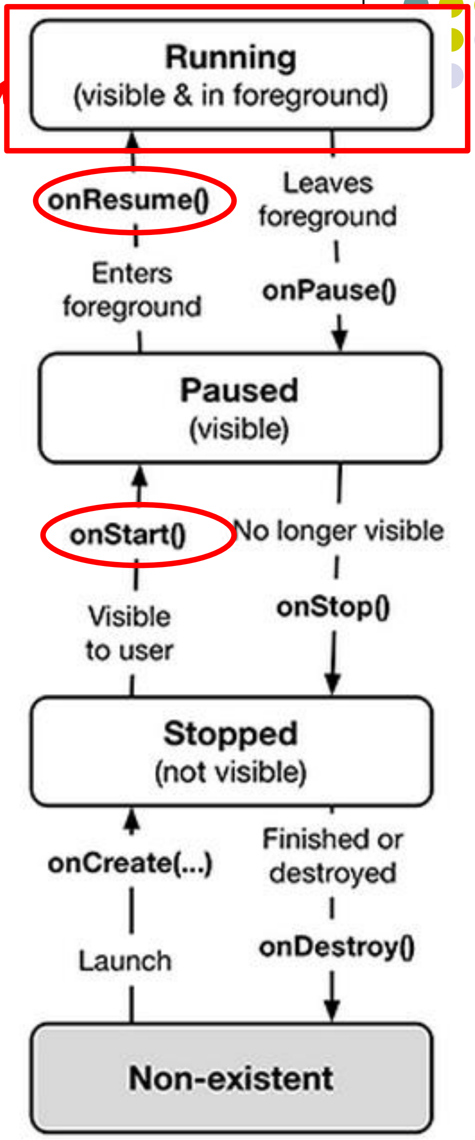
Resuming Stopped App

- A **stopped** app can go back into **running** state if becomes visible and in foreground
- App's **onStart( )** and **onResume( )** methods called to transition from **stopped** to **running** state



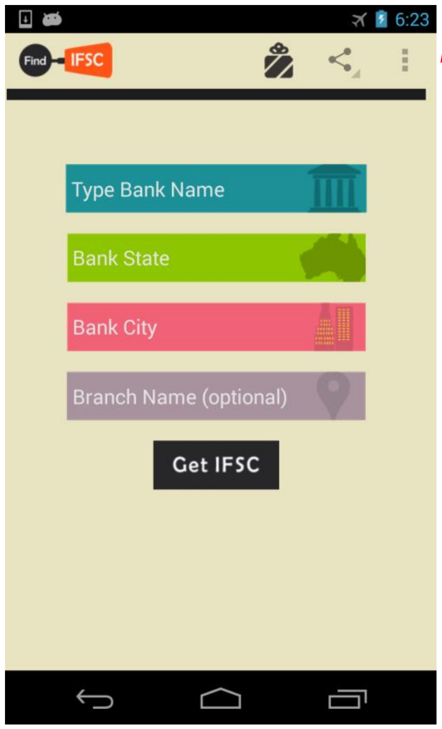


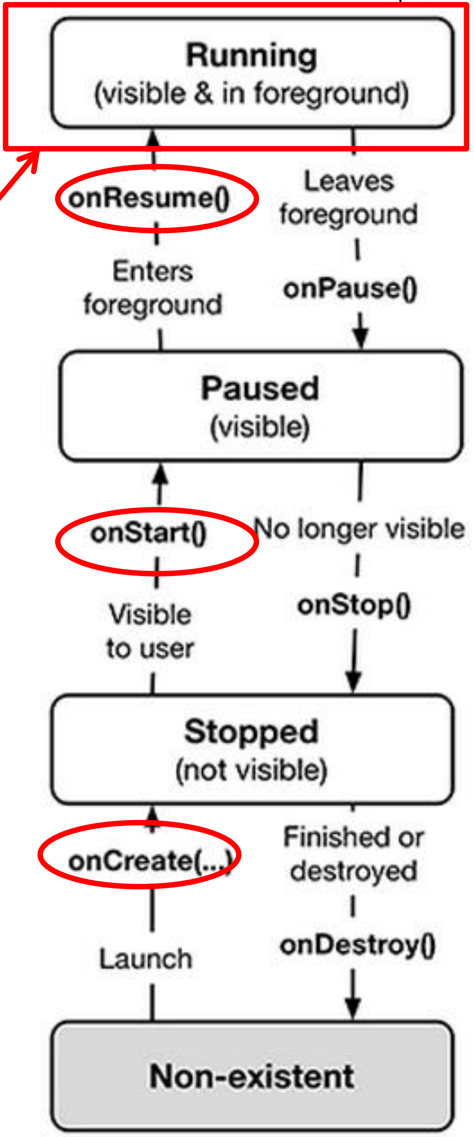
Running



Starting New App

- To start new app, app is launched
- App's **onCreate( ), onStart( )** and **onResume( )** methods are called
- Afterwards new app is **running**

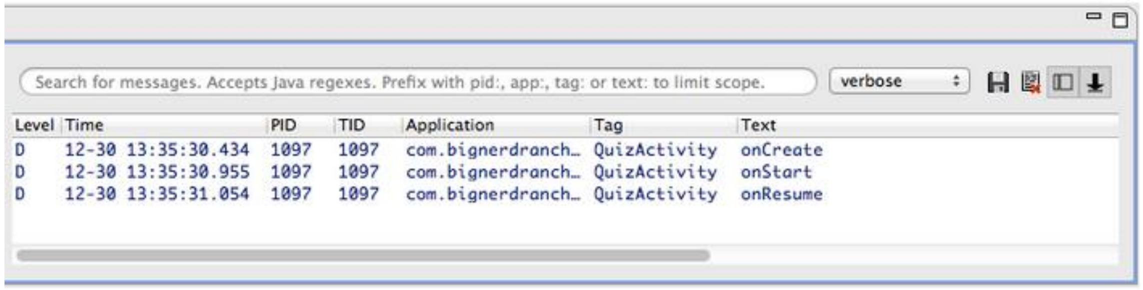




Logging Errors in Android

Logging Errors in Android

- Android can log and display various types of errors/warnings



Search for messages. Accepts Java regexes. Prefix with pid:, app:, tag: or text: to limit scope.
verbose

```
Level Time                PID   TID   Application         Tag           Text
D     12-30 13:35:30.434  1097  1097  com.bignerdranch…   QuizActivity  onCreate
D     12-30 13:35:30.955  1097  1097  com.bignerdranch…   QuizActivity  onStart
D     12-30 13:35:31.054  1097  1097  com.bignerdranch…   QuizActivity  onResume
```

- Error logging is in **Log** class of **android.util** package

  import android.util.Log;

- Turn on logging of different message types by calling appropriate method
- Logged errors/warnings displayed in Android Studio window

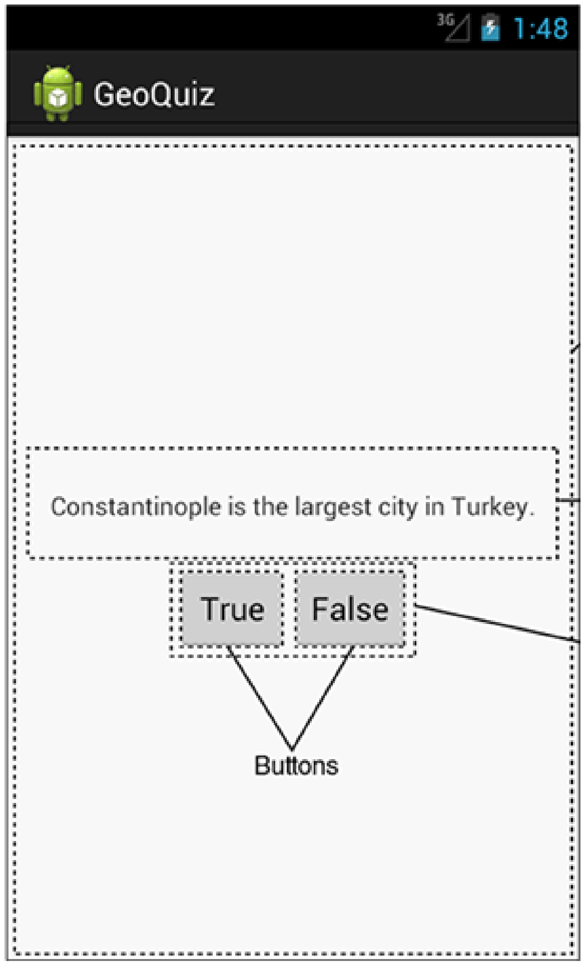
Method	Purpose
`Log.e()`	Log errors
`Log.w()`	Log warnings
`Log.i()`	Log informational messages
`Log.d()`	Log debug messages
`Log.v()`	Log verbose messages

Ref: Introduction to Android Programming, Annuzzi, Darcey & Conder

QuizActivity.java

- A good way to understand Android lifecycle methods is to print debug messages when they are called
- E.g. print debug message from onCreate method below

```
package com.bignerdranch.android.geoquiz;

import android.app.Activity;
import android.os.Bundle;
import android.view.Menu;

public class QuizActivity extends Activity {

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



QuizActivity.java

- Debug (d) messages have the form

```
public static int d(String tag, String msg)
```

- E.g.

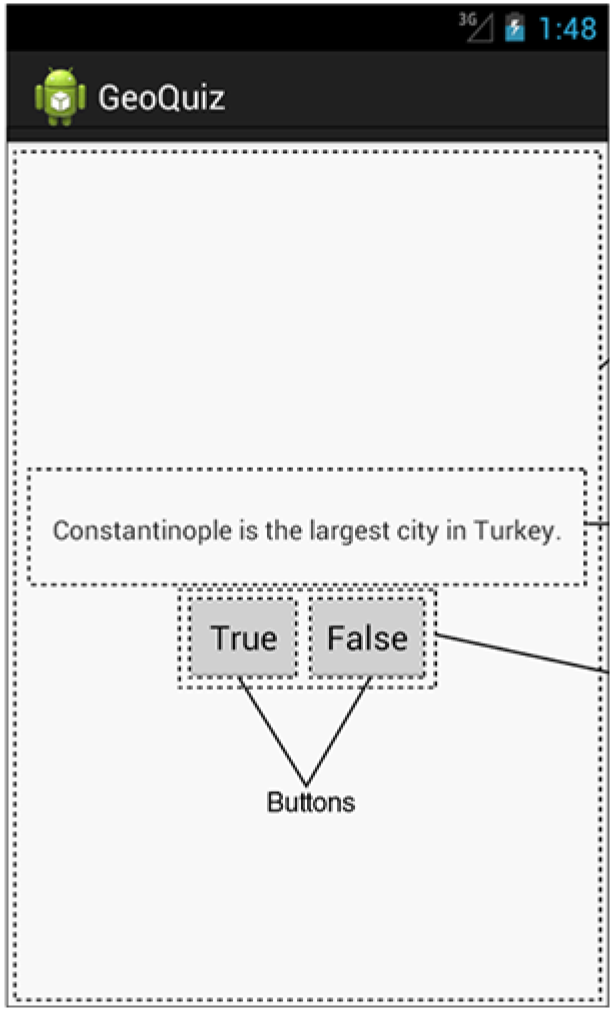
Tag → QuizActivity: Message → onCreate(Bundle) called

- Example declaration:

```
Log.d(TAG, "onCreate(Bundle) called");
```

- Then declare string for **TAG**

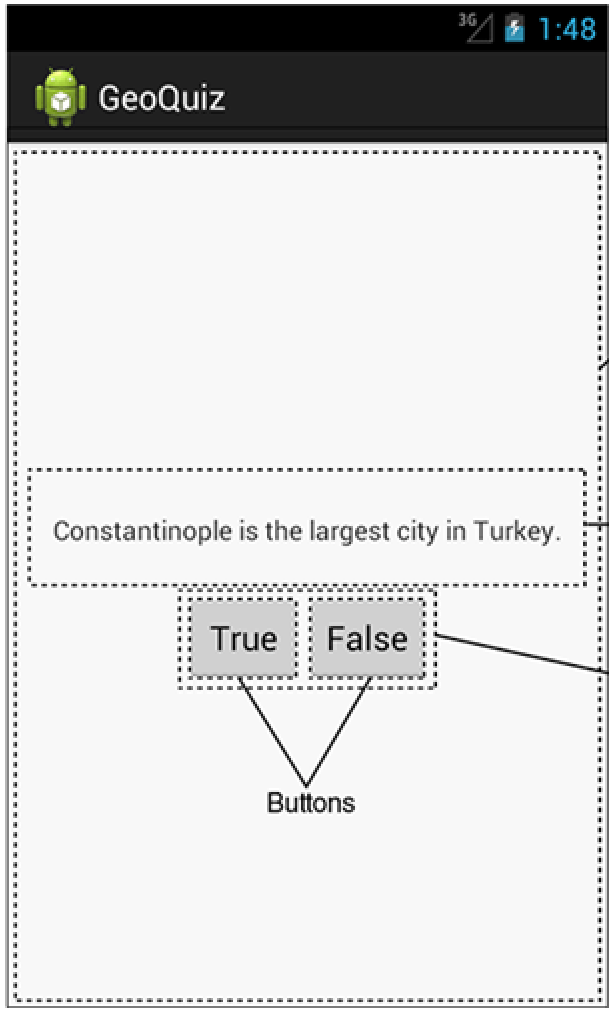
```
public class QuizActivity extends Activity {

    private static final String TAG = "QuizActivity";

    ...

}
```



QuizActivity.java

- Putting it all together

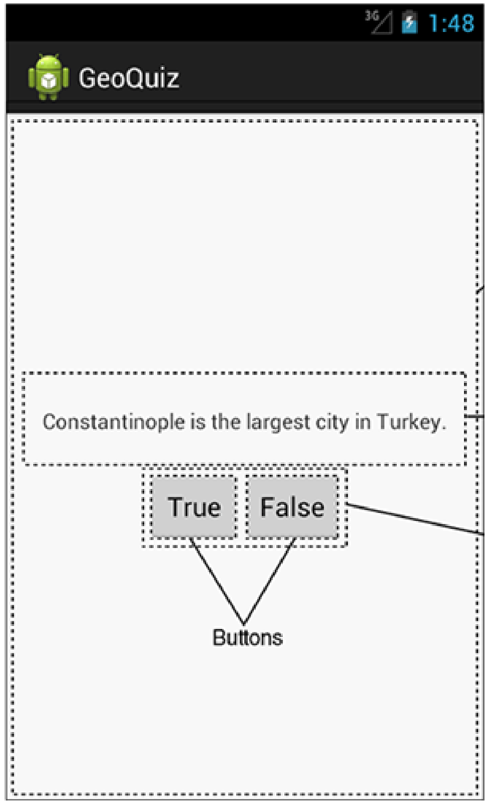
```
public class QuizActivity extends Activity {

    ...

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        Log.d(TAG, "onCreate(Bundle) called");
        setContentView(R.layout.activity_quiz);

        ...
```



QuizActivity.java

- Can overide more lifecycle methods
- Print debug messages from each method
- Superclass calls called in each method

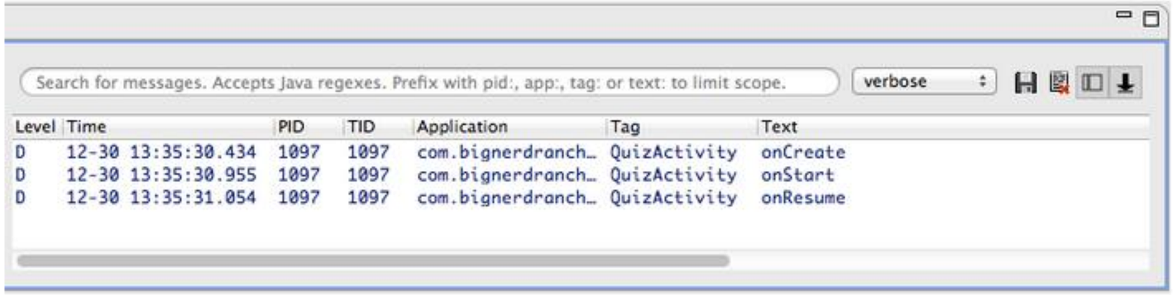
```
    } // End of onCreate(Bundle)

    @Override
    public void onStart() {
        super.onStart();
        Log.d(TAG, "onStart() called");
    }

    @Override
    public void onPause() {
        super.onPause();
        Log.d(TAG, "onPause() called");
    }

    @Override
    public void onResume() {
        super.onResume();
        Log.d(TAG, "onResume() called");
    }

    @Override
    public void onStop() {
        super.onStop();
        Log.d(TAG, "onStop() called");
    }

    @Override
    public void onDestroy() {
        super.onDestroy();
        Log.d(TAG, "onDestroy() called");
    }

}
```



Constantinople is the largest city in Turkey.

QuizActivity.java Debug Messages

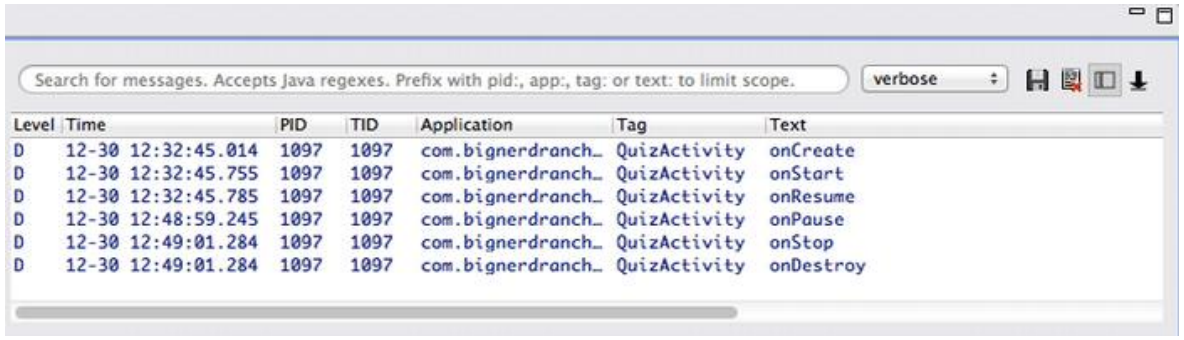
- Launching GeoQuiz app **creates, starts and resumes** an activity



Search for messages. Accepts Java regexes. Prefix with pid:, app:, tag: or text: to limit scope.   verbose

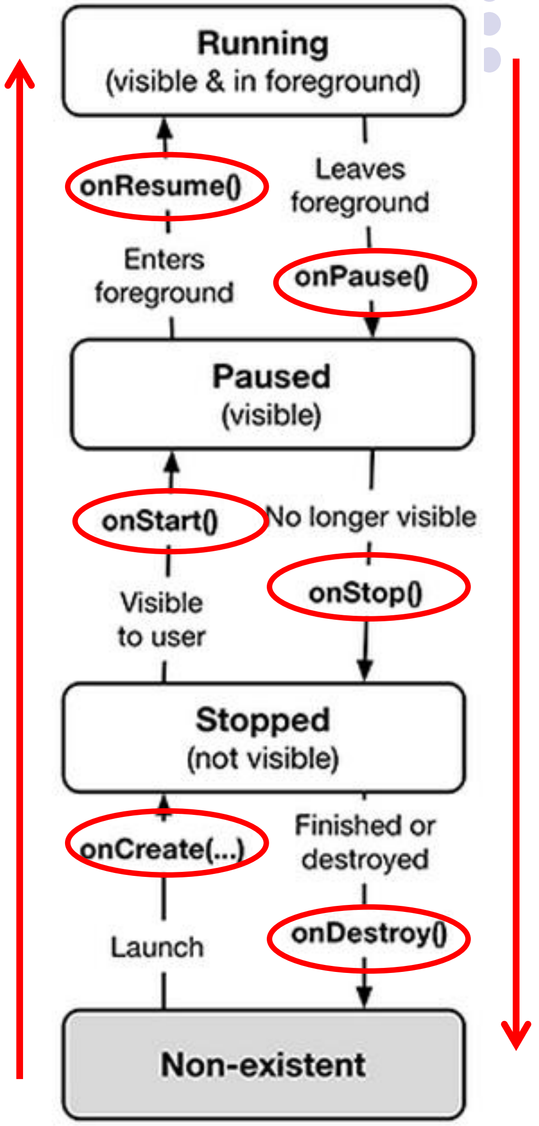
```
Level Time                PID   TID   Application         Tag           Text
D     12-30 13:35:30.434  1097  1097  com.bignerdranch…   QuizActivity  onCreate
D     12-30 13:35:30.955  1097  1097  com.bignerdranch…   QuizActivity  onStart
D     12-30 13:35:31.054  1097  1097  com.bignerdranch…   QuizActivity  onResume
```

- Pressing **Back** button destroys the activity (calls onPause, onStop and onDestroy)



Search for messages. Accepts Java regexes. Prefix with pid:, app:, tag: or text: to limit scope.   verbose

```
Level Time                PID   TID   Application         Tag           Text
D     12-30 12:32:45.014  1097  1097  com.bignerdranch…   QuizActivity  onCreate
D     12-30 12:32:45.755  1097  1097  com.bignerdranch…   QuizActivity  onStart
D     12-30 12:32:45.785  1097  1097  com.bignerdranch…   QuizActivity  onResume
D     12-30 12:48:59.245  1097  1097  com.bignerdranch…   QuizActivity  onPause
D     12-30 12:49:01.284  1097  1097  com.bignerdranch…   QuizActivity  onStop
D     12-30 12:49:01.284  1097  1097  com.bignerdranch…   QuizActivity  onDestroy
```



References

- Busy Coder's guide to Android version 4.4
- CS 65/165 slides, Dartmouth College, Spring 2014
- CS 371M slides, U of Texas Austin, Spring 2014