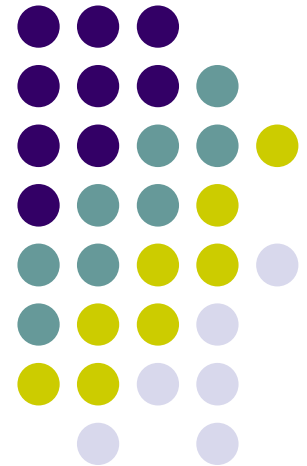


CS 4518 Mobile and Ubiquitous Computing

Lecture 5: Data-Driven Views and Android Components

Emmanuel Agu





Announcements

- Slight modifications to course timeline posted
 - No class February 16 (Advising day)
 - Class on March 2 and 3, 2017
- Today is deadline to form groups for final project
 - Projects 2 & 3 will also be done in these groups
- Final projects from teams D term 2016 are also posted

http://web.cs.wpi.edu/~emmanuel/courses/cs403x/D16/projects/final_project/final_project.html



Data-Driven Layouts



Data-Driven Layouts

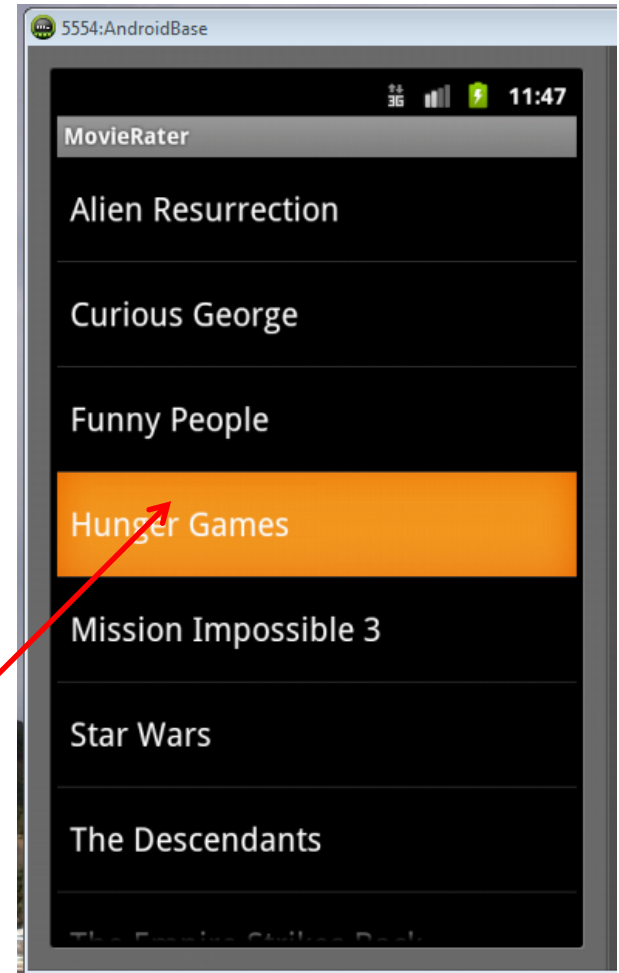
- LinearLayout, RelativeLayout, TableLayout, GridLayout useful for positioning UI elements
 - Data is literally hard coded

- Other layouts dynamically composed from data
 - ListView, GridView, GalleryView
 - Tabs with TabHost, TabControl



Data Driven Layouts

- May want to populate views from a data source (XML file or database)
- Layouts that display repetitive child Views from data
 - ListView
 - GridView
 - GalleryView
- ListView
 - vertical scroll, horizontal row entries, pick item





Data Driven Containers

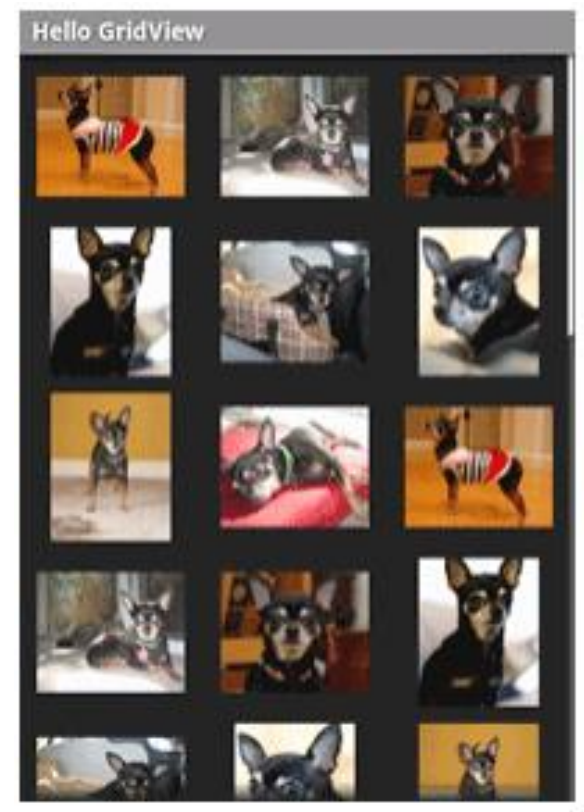
- GridView

- List with specified number of rows and columns



- GalleryView

- List with horizontal scrolling, typically images



AdapterView



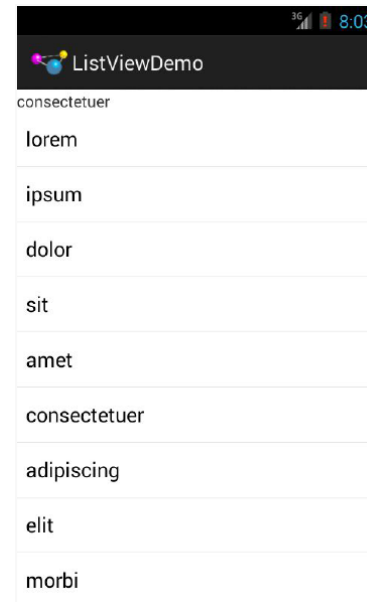
- ListView, GridView, and GalleryView are sub classes of AdapterView (variants)
- **Adapter:** generates widgets from a data source, populates layout
 - E.g. Data is adapted into cells of GridView

Data

lorem
ipsum
dolor
amet
consectetuer
adipiscing
elit
morbi



Adapter

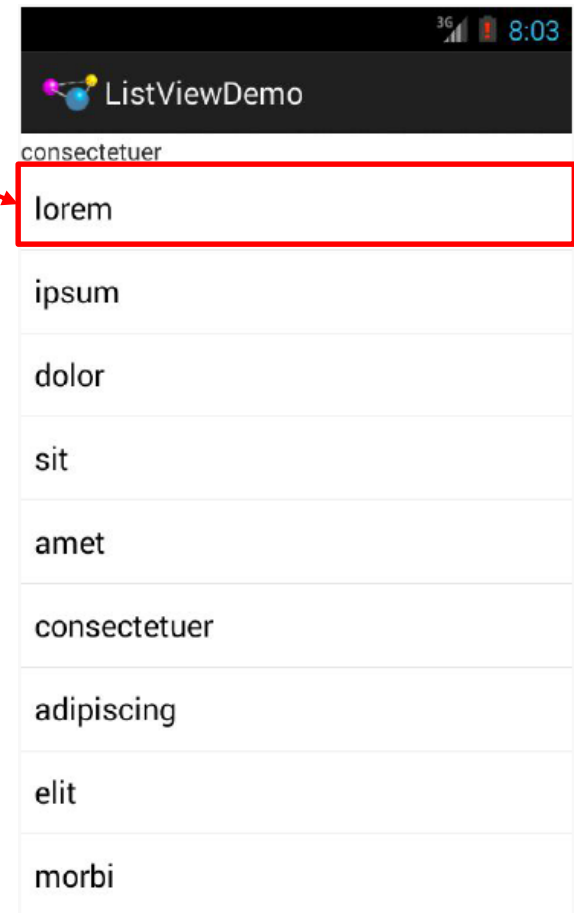


- Most common Adapters
 - **CursorAdapter:** read from database
 - **ArrayAdapter:** read from resource (e.g. XML file)

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 android.R.layout



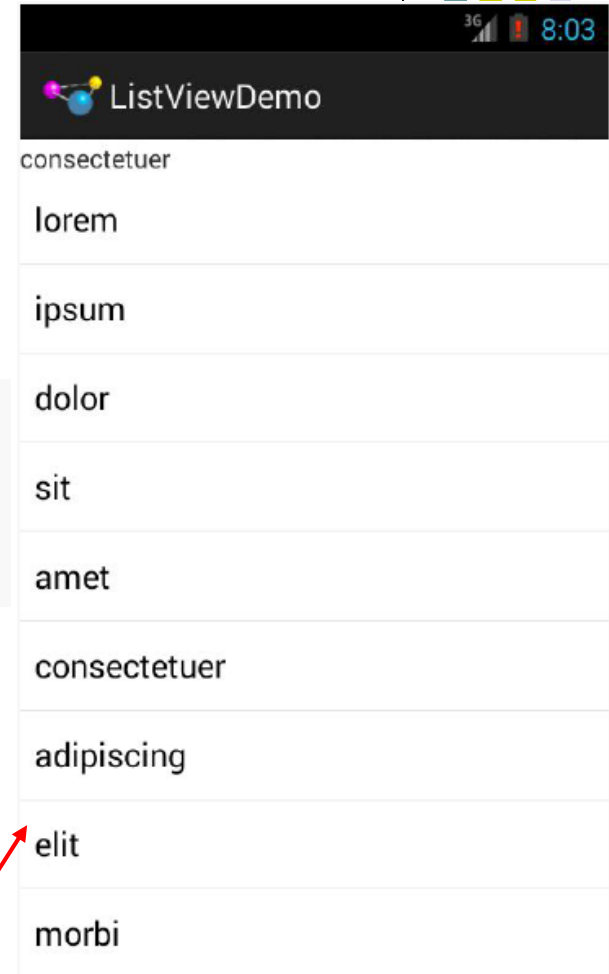


Example: Creating ListView using ArrayAdapter

- **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



**ListView
of items**

Example: Creating ListView using ArrayAdapter

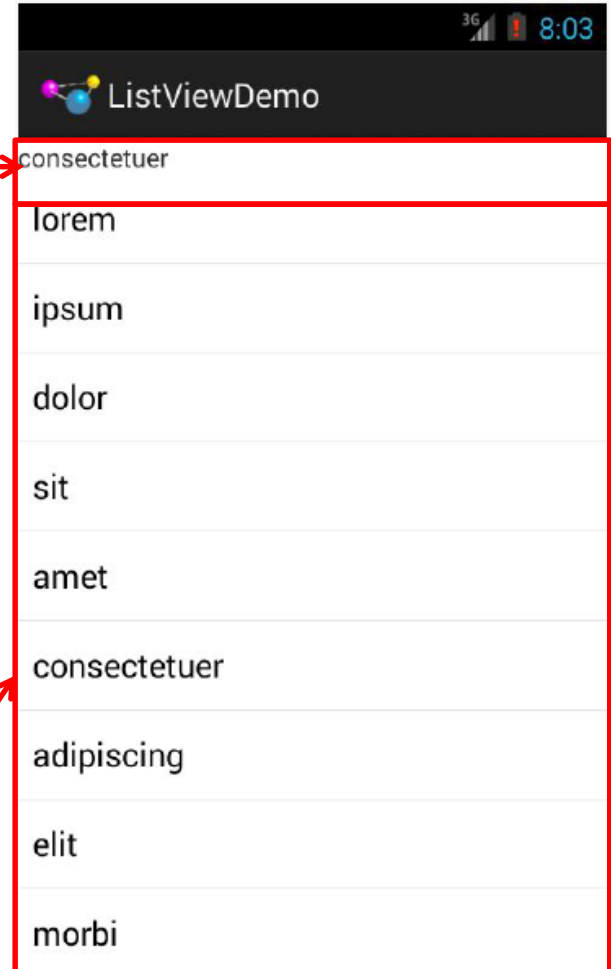


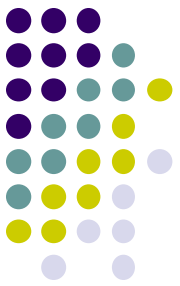
- First create Layout file (e.g. LinearLayout)

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
  xmlns:android="http://schemas.android.com/apk/res/android"
  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 main list of activity





Using ArrayAdapter

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

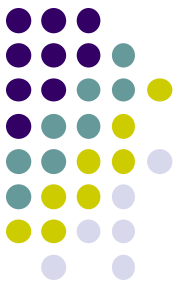
```
String[] items={"this", "is", "a", "really", "silly", "list"};  
new ArrayAdapter<String>(this,  
    android.R.layout.simple_list_item_1,  
    items);
```

Context to use.
(e.g app's activity)

Array of items
to display

Resource ID of
View for formatting

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



Example: Creating ListView using AdapterArray

```
package com.commonware.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 icle) {  
    super.onCreate(icle);  
    setContentView(R.layout.main);  
    setListAdapter(new ArrayAdapter<String>(this,  
        android.R.layout.simple_list_item_1,  
        items));  
    selection=(TextView)findViewById(R.id.selection);  
}
```

Set list adapter (Bridge
Data source and views)

Get handle to TextView
of Selected item

```
@Override
```

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

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
- 4 main types of Android app components:
 - Activities (already seen this)
 - Services
 - Content providers
 - Broadcast receivers



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

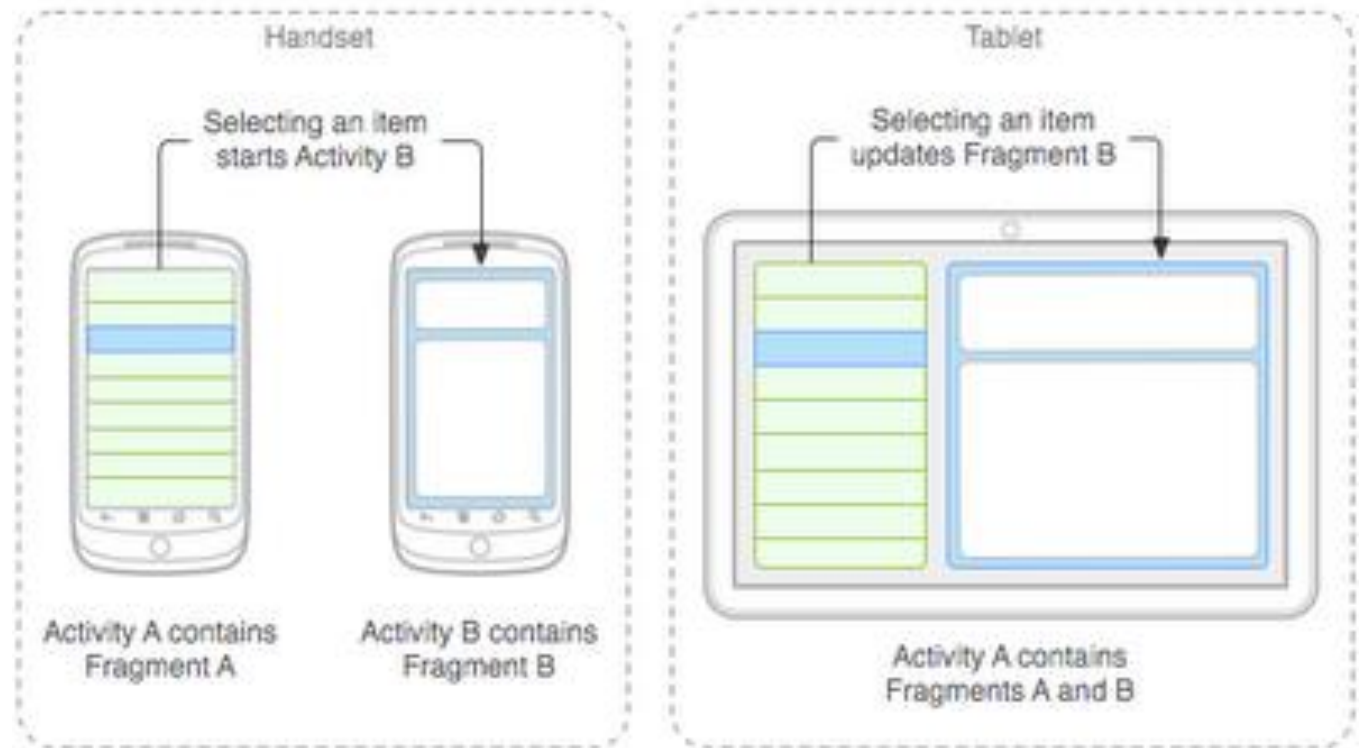
Activity



Fragments



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





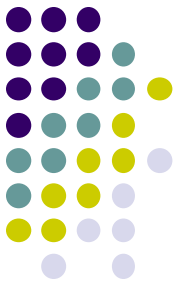
Services

- 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, for cutting and pasting content.
 - **DownloadManager:** manages HTTP downloads in background
 - **FragmentManager:** manages the fragments of an activity.
 - **AudioManager:** provides access to audio and ringer controls.

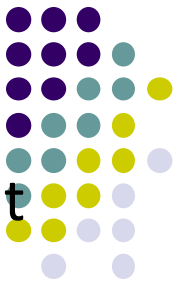


Google Services (In Google Cloud)

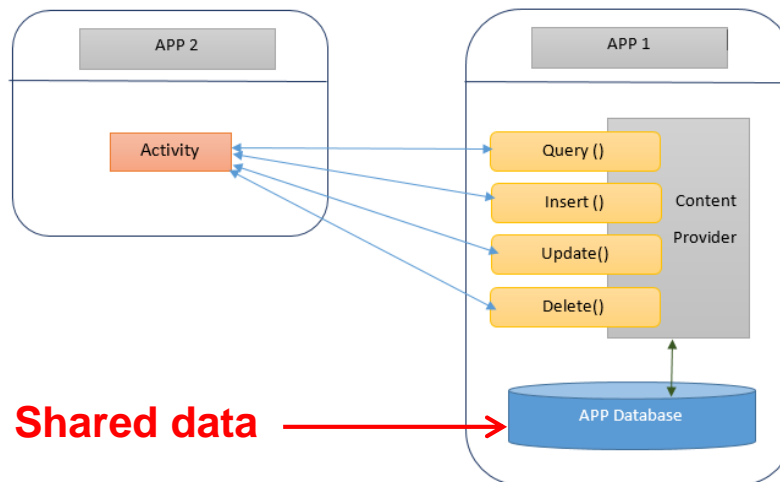
- Maps
- Location-based services
- Game Services
- Authorization APIs
- Google Plus
- Play Services
- In-app Billing
- Google Cloud Messaging
- Google Analytics
- Google AdMob ads

Typically need Internet connection

Content Providers



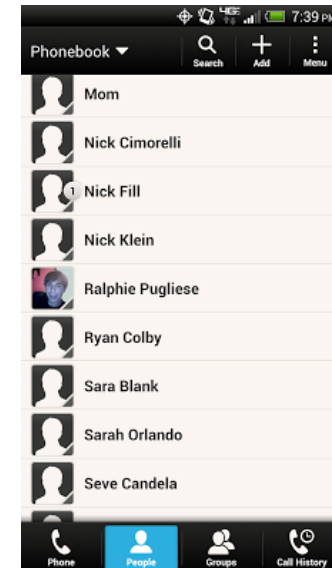
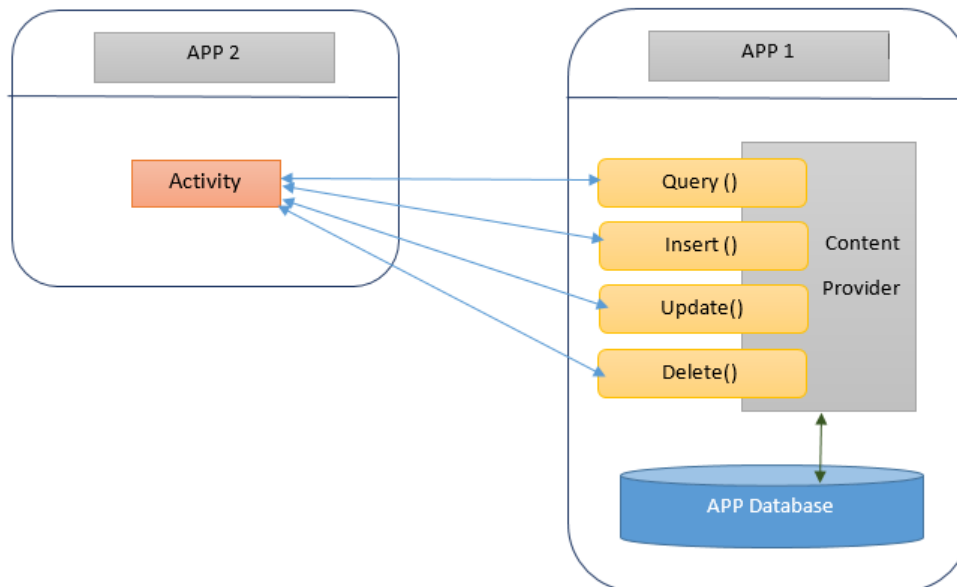
- Android apps can share data (e.g. User's contacts) as content provider
- Content Provider:
 - Abstracts shareable data, makes it accessible through methods
 - Applications can access that shared data by calling methods for the relevant **content provider**
 - E.g. Can query, insert, update, delete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



Content Providers



- 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 don't interact with the UI
 - Commonly create a status bar notification to alert the user when broadcast event occurs
- App Broadcast Receivers are sub-class of **BroadcastReceiver** class

Quiz



- Pedometer App
 - **Component A:** continuously 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 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?



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