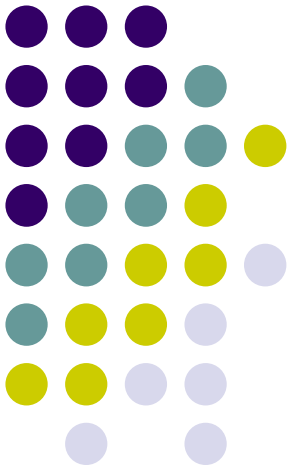
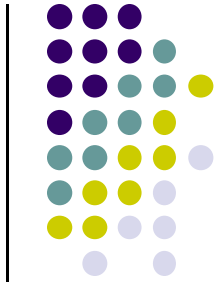
CS 528 Mobile and Ubiquitous Computing

Lecture 4: AdapterViews, Intents, Fragments Camera

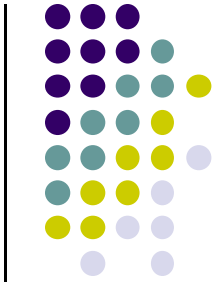
Emmanuel Agu





Layouts with More Interactivity & Data-Dependent

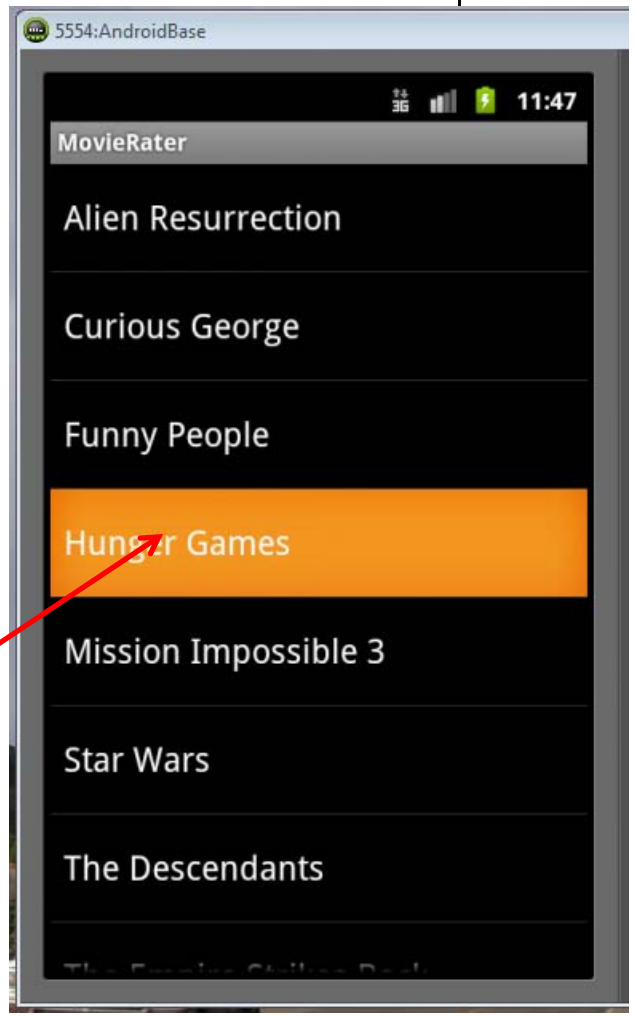
Container Control Classes



- LinearLayout, RelativeLayout, TableLayout, GridLayout useful for positioning UI elements
  - the layouts themselves are not interactive although the child Views may be
- Other available layouts have more interactivity between the user and the child Views
  - ListView, GridView, GalleryView
  - Tabs with TabHost, TabControl
  - ScrollView, HorizontalScrollView

Data Driven Containers

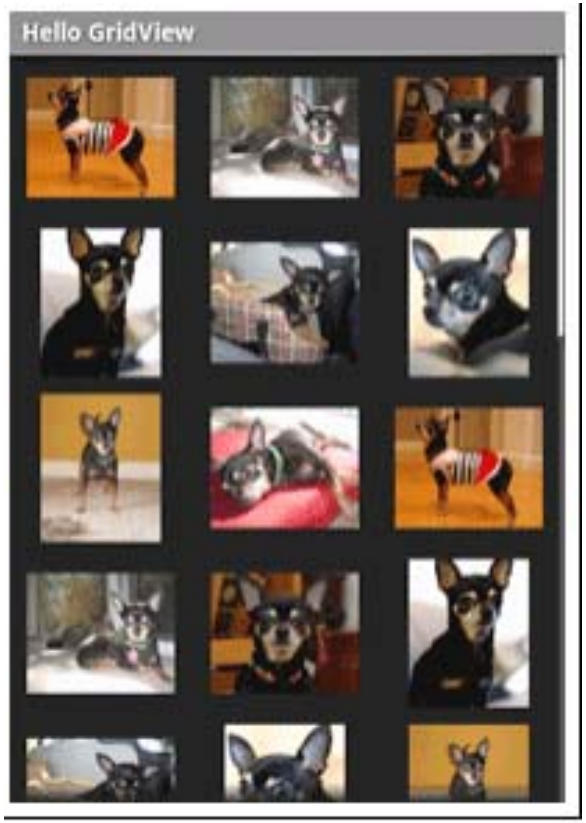
- May want to populate views from a data source (XML file or database)
- Containers that display repetitive child View controls in a given way
  - ListView
  - GridView
  - GalleryView
- ListView
  - vertical scroll, horizontal row entries, pick item



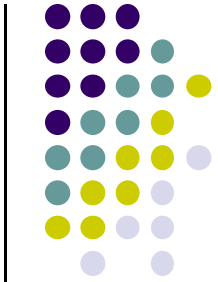
Data Driven Containers

- GridView
  - specified number of rows and columns

- GalleryView
  - horizontal scrolling list, typically images

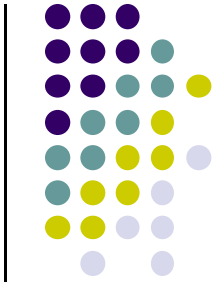


AdapterView

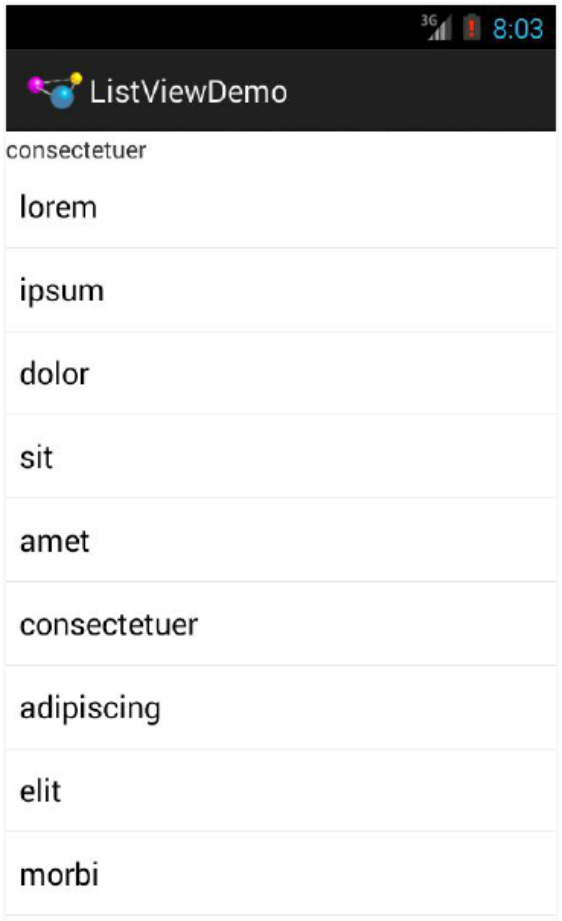


- ListView, GridView, and GalleryView are all sub classes of AdapterView
- Adapter generates child Views from some data source and populates the larger View.
  - E.g. Data is adapted into cells of GridView
- Most common Adapters
  - **CursorAdapter** used to read from database
  - Use **ArrayAdapter** to read from resource, typically an XML file

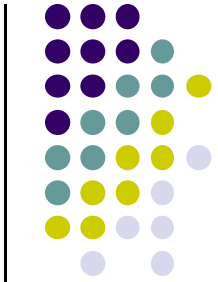
Adapters



- When using an Adapter a layout is defined for each child element (View)
- The adapter
  - Creates Views using layout for each element in data source
  - Fills the containing View (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



Using ArrayAdapter



- Wraps adapter around a Java 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. Typically app's activity instance

Array of Items to show

Resource ID of View to use

- In example, **android.R.layout.simple_list_item_1** turns strings into textView objects
- TextView widgets shown in list using this ArrayAdapter

Example: Creating ListView using AdapterArray

- Want to create the following listView from the following strings

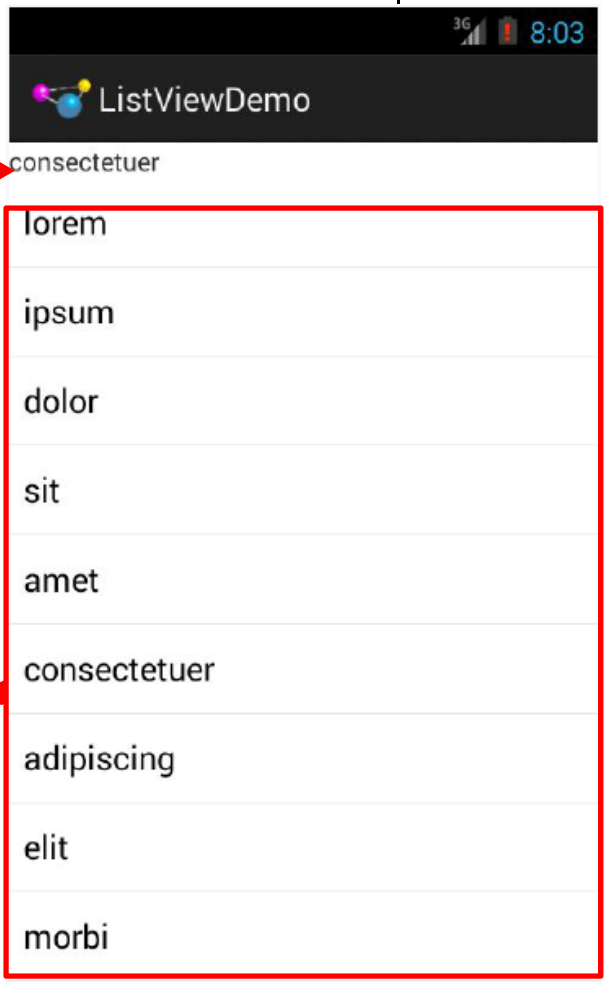
```
private static final String[] items={"lorem", "ipsum", "dolor",
        "sit", "amet",
        "consectetuer", "adipiscing", "elit", "morbi", "vel",
        "ligula", "vitae", "arcu", "aliquet", "mollis",
        "etiam", "vel", "erat", "placerat", "ante",
        "porttitor", "sodales", "pellentesque",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

Example: Creating ListView using AdapterArray

- First create LinearLayout

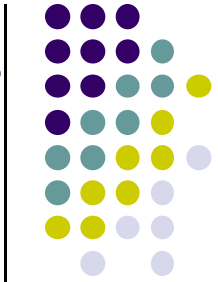
```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
  xmlns:android="http://schemas.android.co
  android:orientation="vertical"
  android:layout_width="match_parent"
  android:layout_height="match_parent" >
  <TextView
    android:id="@+id/selection"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"/>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

Example: Creating ListView using AdapterArray



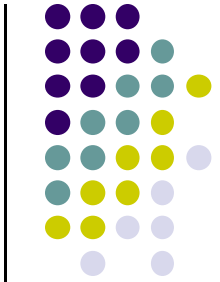
```
package com.commonsware.android.list;

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)

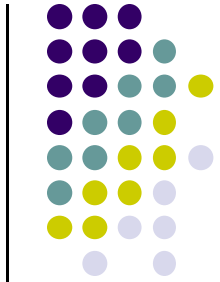
Get handle to TextView of Selected item

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

Selection Events



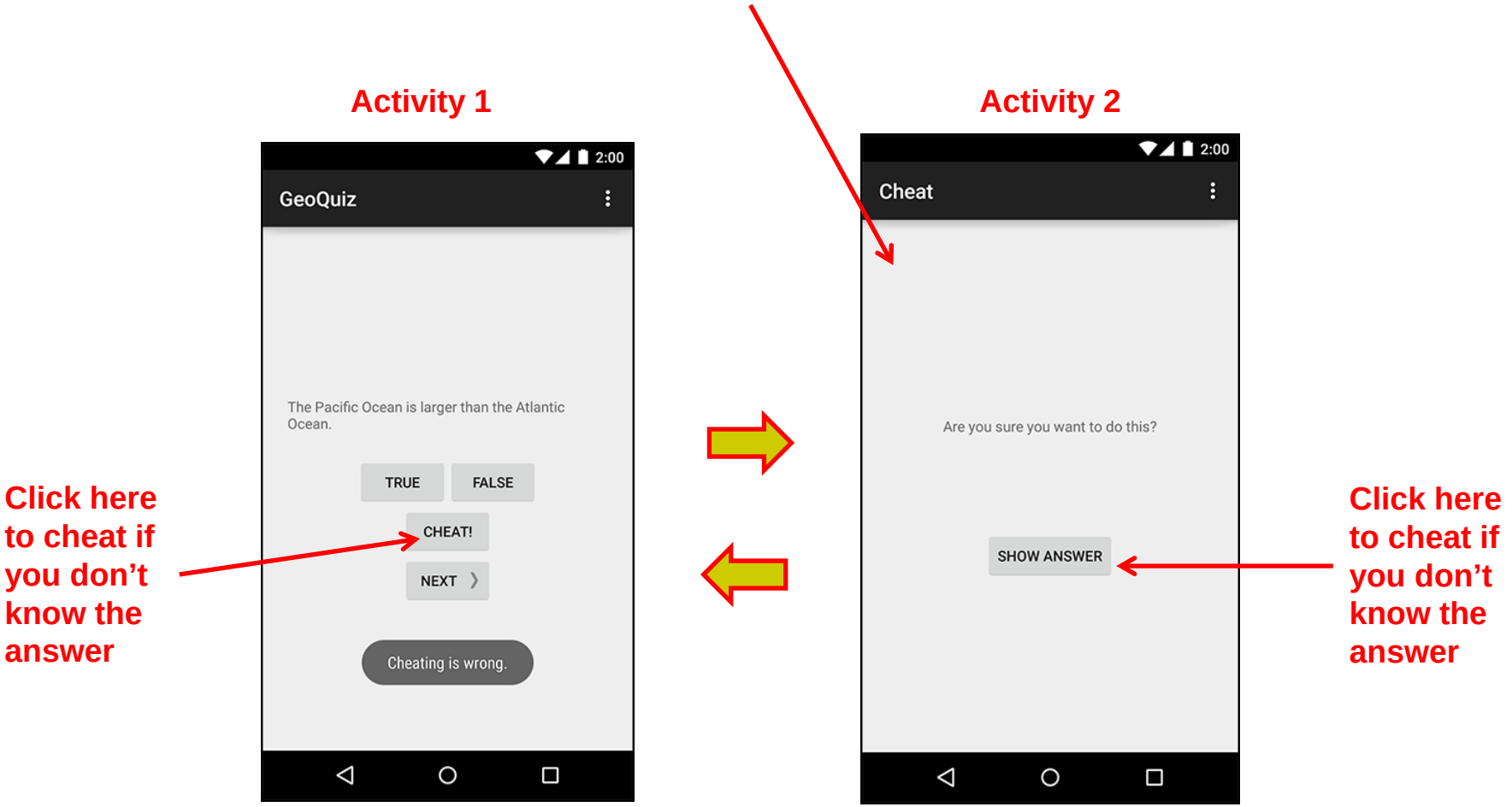
- ListView, GridView, GalleryView
- Typically user can select one item of data
- Implement the OnItemClickListener class, set it as the listener
- This approach is used a lot:
  - create a class that implements some kind of listener
  - register it with a control



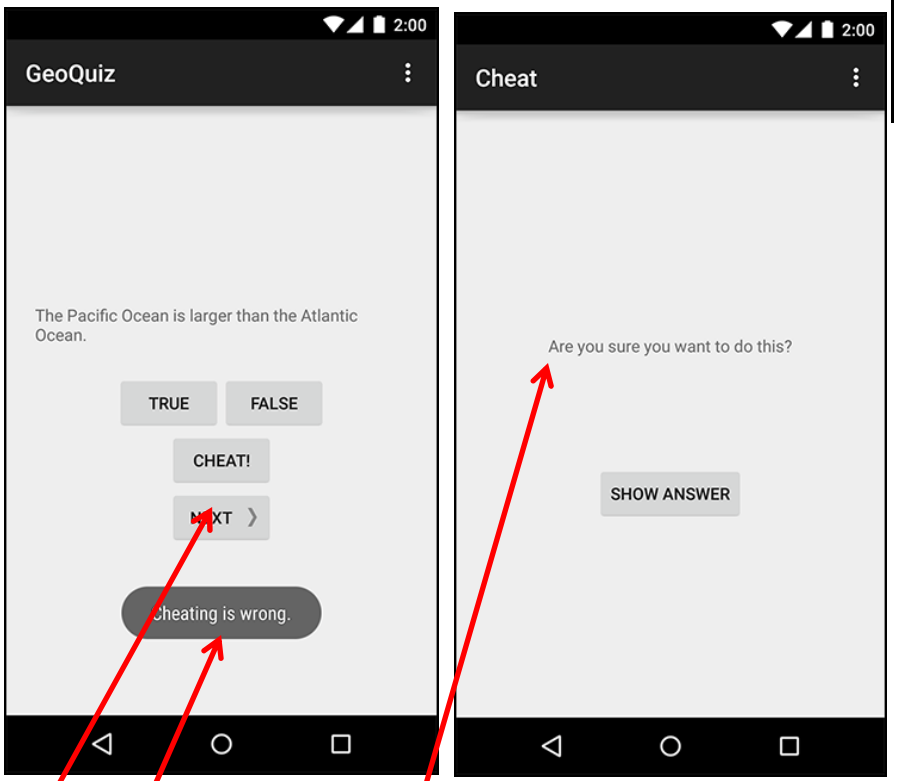
Starting Activity 2 from Activity 1

Why would we want to do this?
Ref: Android Nerd Ranch (2nd edition) pg 87

- May want to allow user to cheat by getting answer to quiz
- Second screen pops up to show Answer

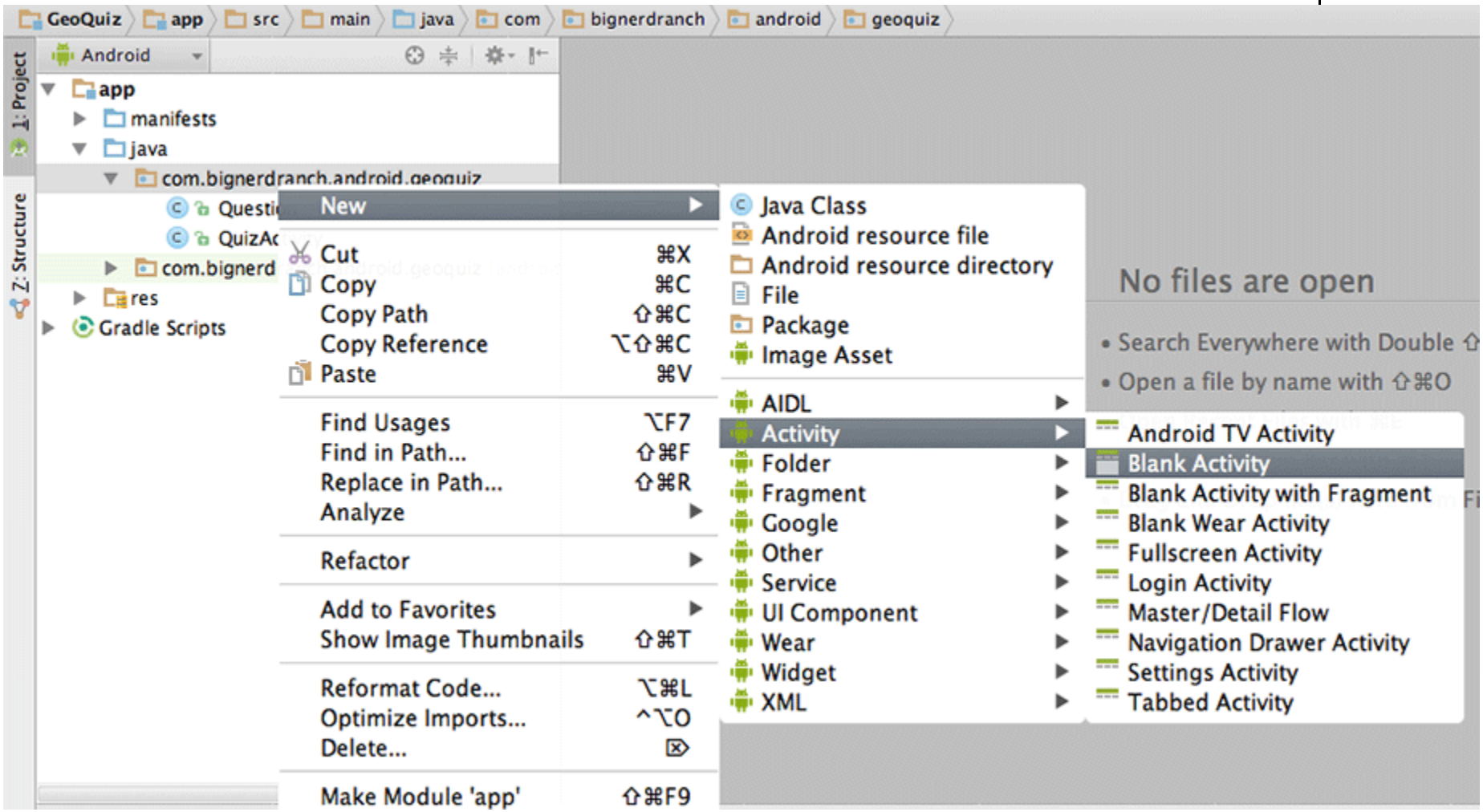


Add Strings for Activity 1 and Activity 2 to strings.xml

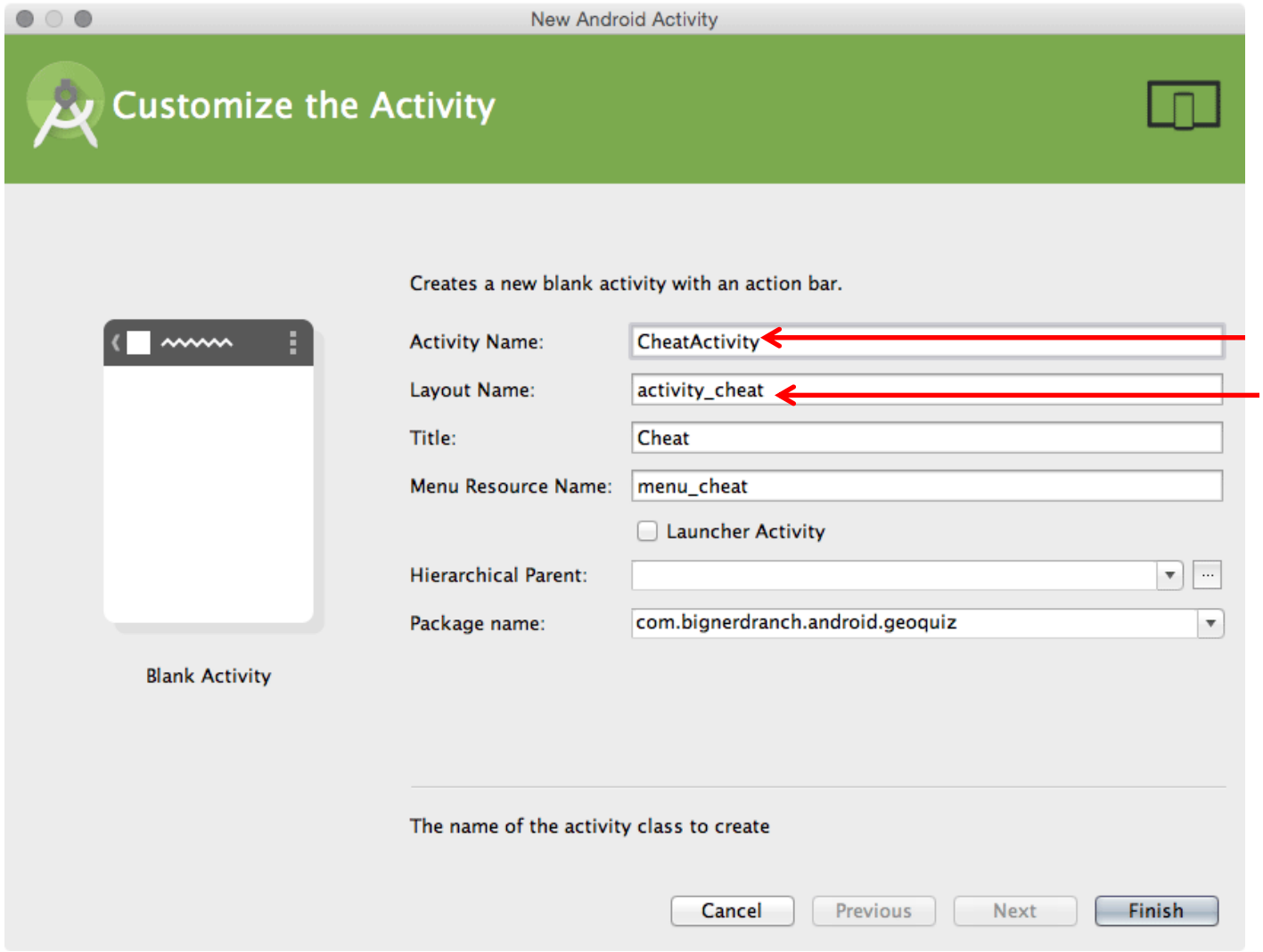



```
<?xml version="1.0" encoding="utf-8"?>
<resources>

    ...
    <string name="question_asia">Lake Baikal is the world\'s oldest and
deepest
      freshwater lake.</string>
    <string name="warning_text">Are you sure you want to do this?</string>
    <string name="show_answer_button">Show Answer</string>
    <string name="cheat_button">Cheat!</string>
    <string name="judgment_toast">Cheating is wrong.</string>

</resources>
```

Create Blank Activity (for Activity 2) in Android Studio



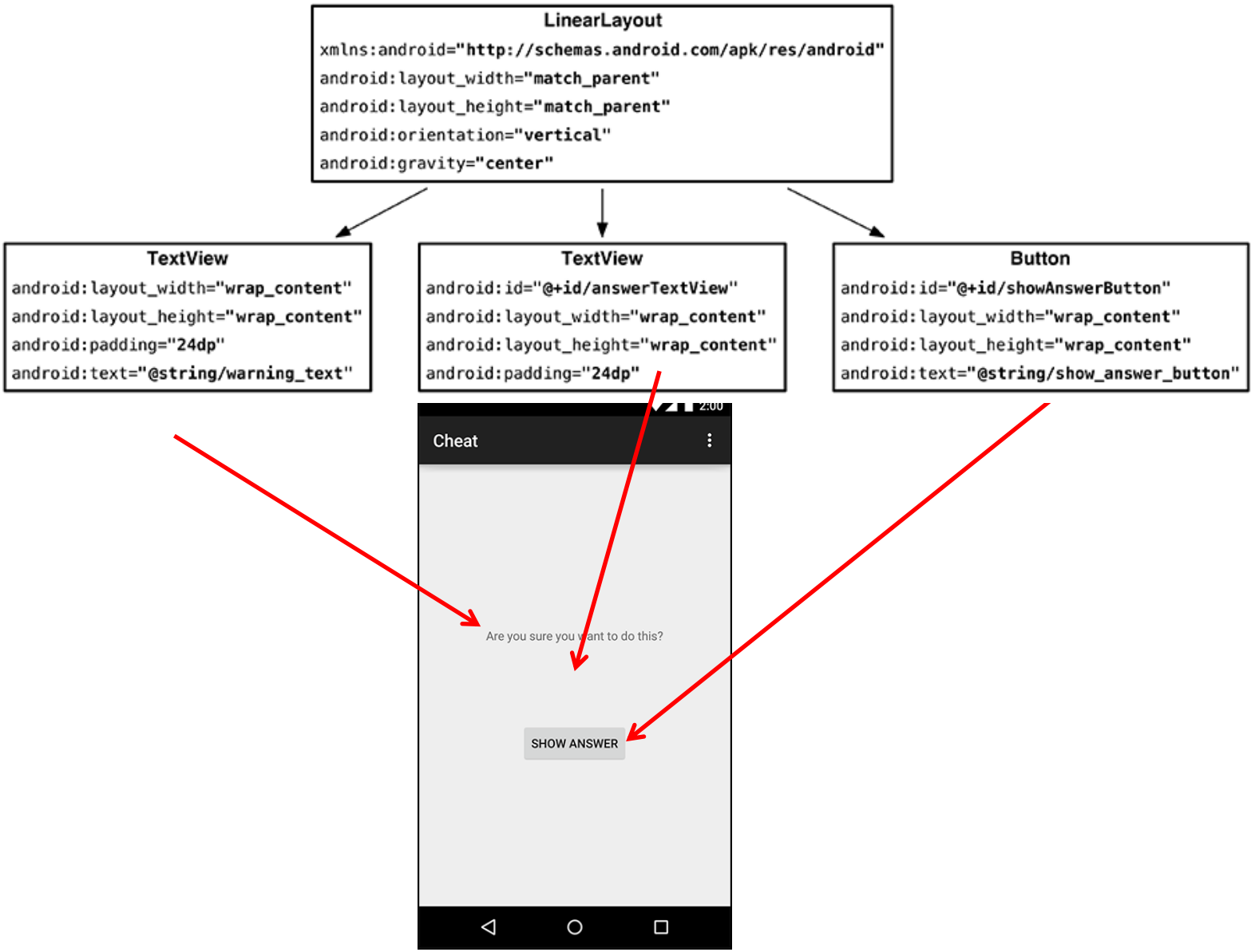
Specify Name and XML file for Activity 2



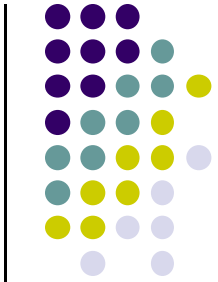
Code in CheatActivity.java

Uses activity_cheat.xml

Design Layout for Screen 2



Write XML Layout Code for Screen 2



```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
              xmlns:tools="http://schemas.android.com/tools"
              android:layout_width="match_parent"
              android:layout_height="match_parent"
              android:gravity="center"
              android:orientation="vertical"

tools:context="com.bignerdranch.android.geoquiz.CheatActivity">

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:padding="24dp"
        android:text="@string/warning_text"/>

    <TextView
        android:id="@+id/answer_text_view"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:padding="24dp"
        tools:text="Answer"/>

    <Button
        android:id="@+id/show_answer_button"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/show_answer_button"/>

</LinearLayout>
```

Activity 2

Declare New Activity in AndroidManifest.xml

- Create new activity (CheatActivity) in Android Studio

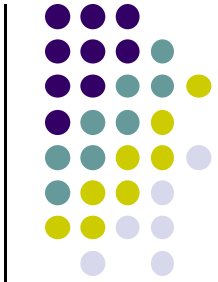
```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.bignerdranch.android.geoquiz" >

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:theme="@style/AppTheme" >
        <activity
            android:name=".QuizActivity"
            android:label="@string/app_name" >
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
        <activity
            android:name=".CheatActivity"
            android:label="@string/title_activity_cheat" >
        </activity>
    </application>

</manifest>
```

Activity 1

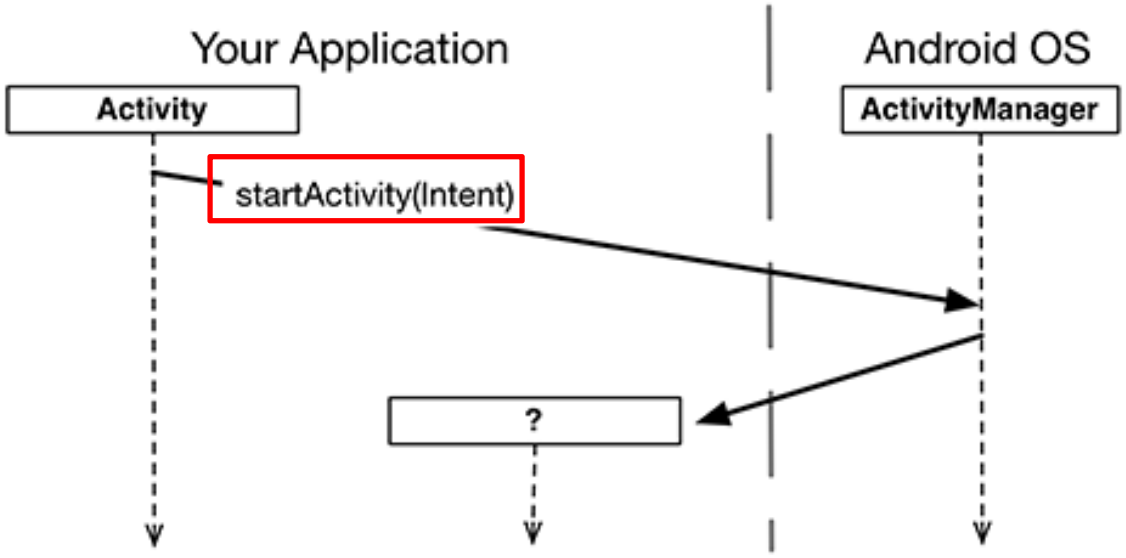
Activity 2 (CheatActivity)

Activity 2 (CheatActivity)

Starting Activity 2 from Activity 1



- Activity 1 starts activity 2 **through** the Android OS
- Activity 1 starts activity 2 by calling **startActivity(Intent)**
- Passes Intent (object for communicating with Android OS)

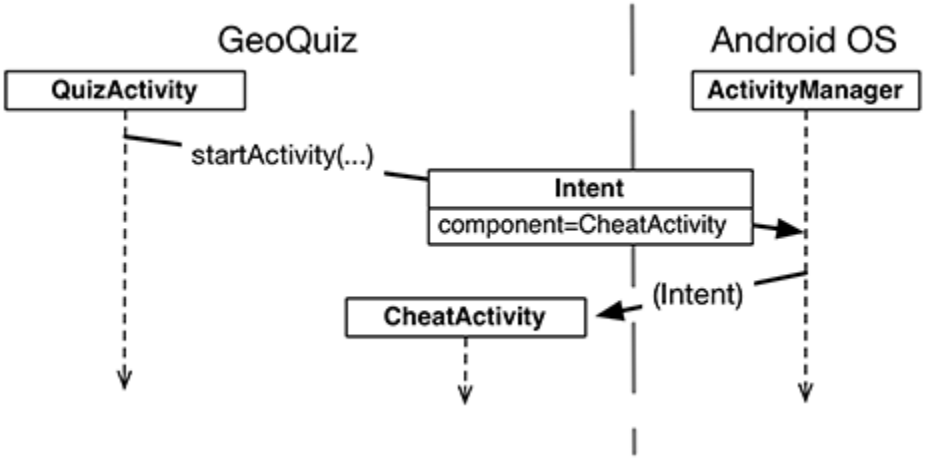


- Intent specifies which Activity OS ActivityManager should start

Starting Activity 2 from Activity 1

- Intents have many different constructors. We will use form:

```
public Intent(Context packageContext, Class<?> cls)
```



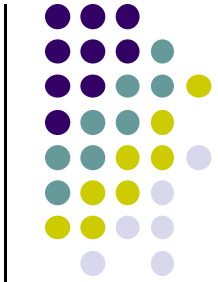
- Actual code looks like this

```
mCheatButton = (Button)findViewById(R.id.cheat_button);
mCheatButton.setOnClickListener(new View.OnClickListener() {

    @Override
    public void onClick(View v) {
        // Start CheatActivity
        Intent i = new Intent(QuizActivity.this, CheatActivity.class);
        startActivity(i);
    }
});

...
```

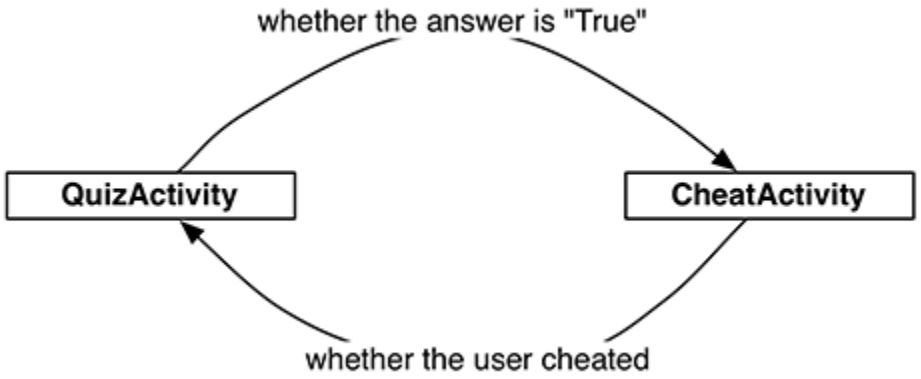
Parent Activity

Activity 2

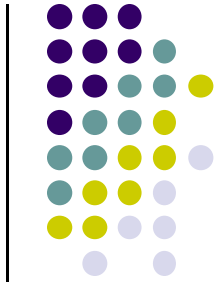
Final Words on Intents



- Previous example is called an **explicit intent** because Activity 1 and activity 2 are in same app
- If Activity 2 were in another app, an **implicit intent** would have to be created instead
- Can also pass data between Activities 1 or 2
  - E.g. New Activity 2 can tell activity 1 if user checked answer

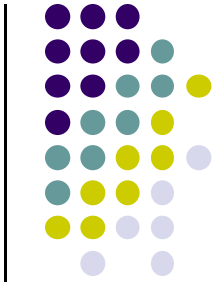


See Android Nerd Ranch for more details

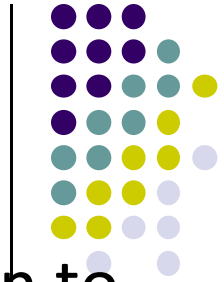


Intents

Intents



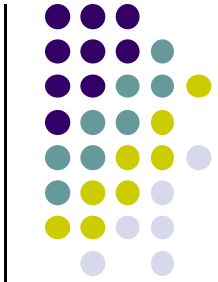
- Allows apps to use Android applications and components
  - start **activities**
  - start **services**
  - deliver **broadcasts**
- Also allows other apps to use components of our apps
- Examples of Google applications: http://developer.android.com/guide/appendix/g-app-intents.html



Intents

- "An intent is an abstract description of an operation to be performed"
- Intents consist of:
  - **Action** (what to do, example visit a web page)
  - **Data** (to perform operation on, example web page url)
- Commands related with Intents: **startActivity**, **startActivityForResult**, **startService**, **bindService**

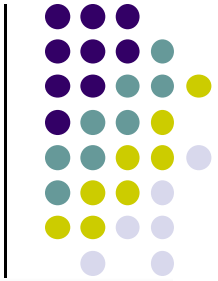
Intent Object Info



- data for the Android system
  - category of component to handle intent (activity, service, broadcast receiver)
  - instructions on how to launch component if necessary

- data for the component that receives the intent
  - action to take
  - data to act on

Recall: Inside AndroidManifest.xml



Your package name

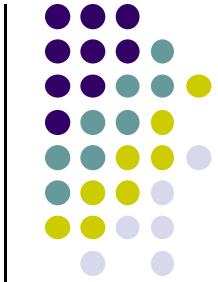
Android version

List of activities (screens) in your app

```
<?xml version="1.0"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
  package="com.commonsware.android.skeleton"
  android:versionCode="1"
  android:versionName="1.0">

  <application>
    <activity
      android:name="Now"
      android:label="Now">
      <intent-filter>
        <action android:name="android.intent.action.MAIN" />

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

</manifest>
```

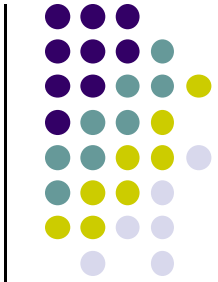
Action of intent

Intent Action



Constant	Target component	Action
ACTION_CALL	activity	Initiate a phone call.
ACTION_EDIT	activity	Display data for the user to edit.
ACTION_MAIN	activity	Start up as the initial activity of a task, with no data input and no returned output
ACTION_SYNC	activity	Synchronize data on a server with data on the mobile device.
ACTION_BATTERY_LOW	broadcast receiver	A warning that the battery is low.
ACTION_HEADSET_PLUG	broadcast receiver	A headset has been plugged into the device, or unplugged from it.
ACTION_SCREEN_ON	broadcast receiver	The screen has been turned on.
ACTION_TIMEZONE_CHANGED	broadcast receiver	The setting for the time zone has changed.

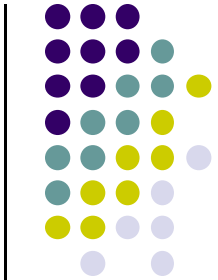
Intent Info - *Data*



- URI (uniform resource identifier) of data to work with / on
  - for content on device a content provider and identifying information, for example an audio file or image or contact

- MIME (Multipurpose Internet Mail Extension, now internet media type) initially for email types, but extended to describe type information in general about data / content
  - `image/png` or `audio/mpeg`

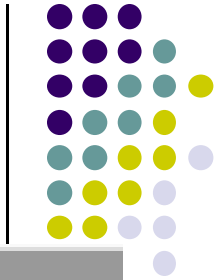
Intent Info - *Category*



- String with more information on what kind of component should handle Intent

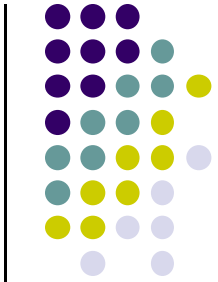
Constant	Meaning
`CATEGORY_BROWSABLE`	The target activity can be safely invoked by the browser to display data referenced by a link – for example, an image or an e-mail message.
`CATEGORY_GADGET`	The activity can be embedded inside of another activity that hosts gadgets.
`CATEGORY_HOME`	The activity displays the home screen, the first screen the user sees when the device is turned on or when the *Home* button is pressed.
`CATEGORY_LAUNCHER`	The activity can be the initial activity of a task and is listed in the top-level application launcher.
`CATEGORY_PREFERENCE`	The target activity is a preference panel.

Intent Constructors



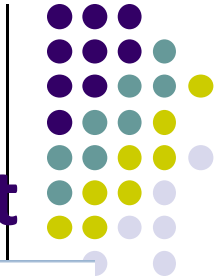
Public Constructors
Intent () Create an empty intent.
Intent (Intent o) Copy constructor.
Intent (String action) Create an intent with a given action.
Intent (String action, Uri uri) Create an intent with a given action and for a given data url.
Intent (Context packageContext, Class<?> cls) Create an intent for a specific component.
Intent (String action, Uri uri, Context packageContext, Class<?> cls) Create an intent for a specific component with a specified action and data.

AndroidManifest.xml



- describes app components:
  - activities, services, broadcast receivers, content providers
- **Intents:** Also describes *intent messages each component can handle*
- **Permissions:** declares permissions requested by app
- **Libraries:** libraries application to link to

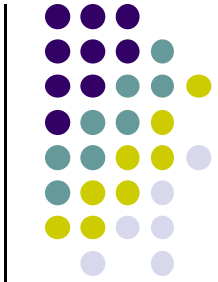
Recall: AndroidManifest.xml - Launcher Intent



```
1  <?xml version="1.0" encoding="utf-8"?>
2  <manifest xmlns:android="http://schemas.android.com/apk/res/android"
3      package="scott.examples.lifeCycleTest"
4      android:versionCode="1"
5      android:versionName="1.0" >
6
7      <uses-sdk android:minSdkVersion="10" />
8
9      <application
0          android:icon="@drawable/ic_launcher"
1          android:label="@string/app_name" >
2          <activity
3              android:name=".LifeCycleTestActivity"
4              android:label="@string/app_name" >
5              <intent-filter>
6                  <action android:name="android.intent.action.MAIN" />
7                  <category android:name="android.intent.category.LAUNCHER" />
8              </intent-filter>
9          </activity>
0                  <activity
1              android:name=".NameGetter"
2              android:label="@string/getName"/>
3      </application>
4
5  </manifest>
```

Declare this as Activity to start when app is started

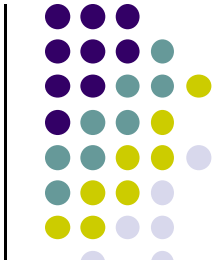
Intent - *Extras*



- A *Bundle* (like a map / dictionary, key-value pairs) of additional information to be given to the component handling the Intent

- Some Action will have specified extras
  - E.g. ACTION_TIMEZONE_CHANGED will have an extra with key of "time-zone"

From MyFirstActivity

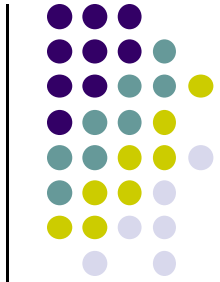


```
/** Called when the user clicks the Send button */
public void sendMessage(View view) {
    Intent intent = new Intent(this, DisplayMessageActivity.class);
    EditText editText = (EditText) findViewById(R.id.edit_message);
    String message = editText.getText().toString();
    intent.putExtra(EXTRA_MESSAGE,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

Create new Intent

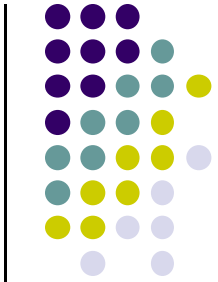
Put text typed in
by user into intent

Get message typed
in by user,
Convert to string

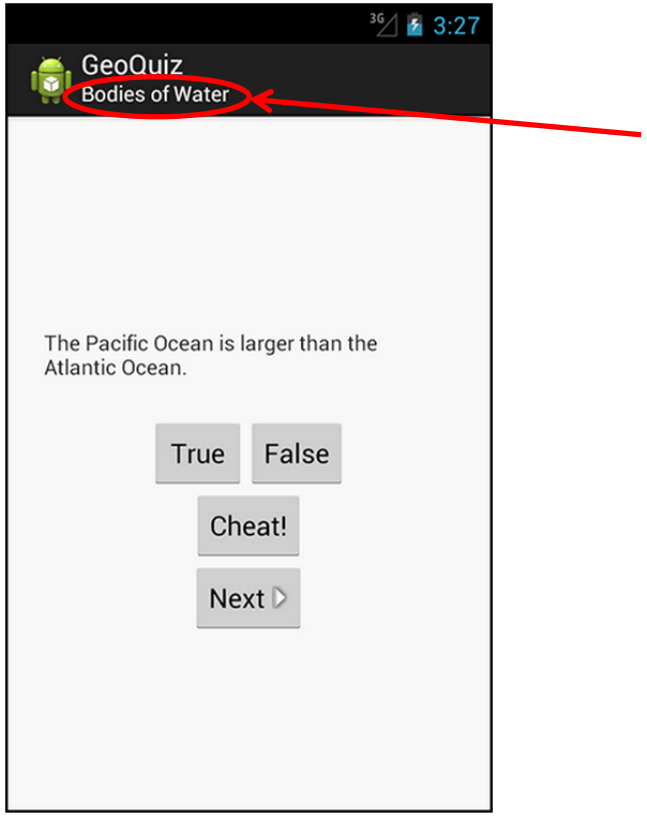


Action Bar

Action Bar (Ref: Android Nerd Ranch 1st Edition)



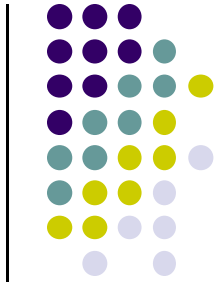
- Can add Action bar to the onCreate( ) method of GeoQuiz to indicate what part of the app we are in



Action bar

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

    ActionBar actionBar = getActionBar();
    actionBar.setSubtitle("Bodies of Water");
```

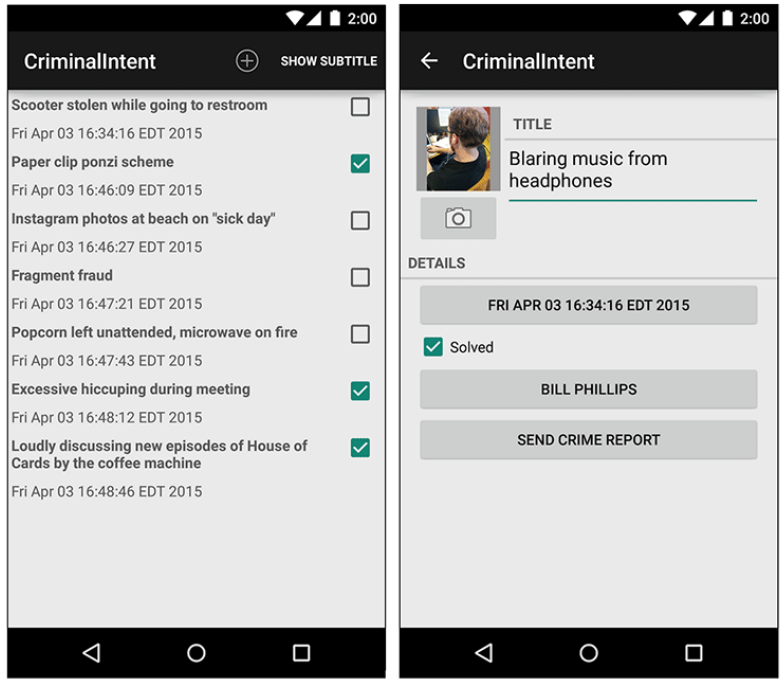
Code to add action bar

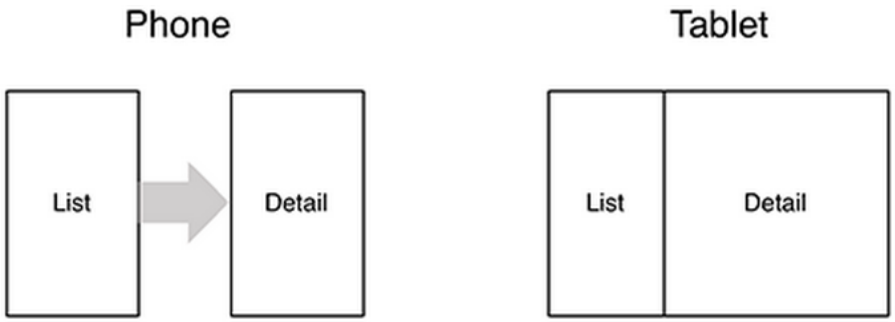


Fragments

Fragments

- To illustrate fragments, we create new app **CriminalIntent**
- Used to record “office crimes” e.g. leaving plates in sink, etc
- Record includes:
  - Title, date, photo
- List-detail app + Fragments

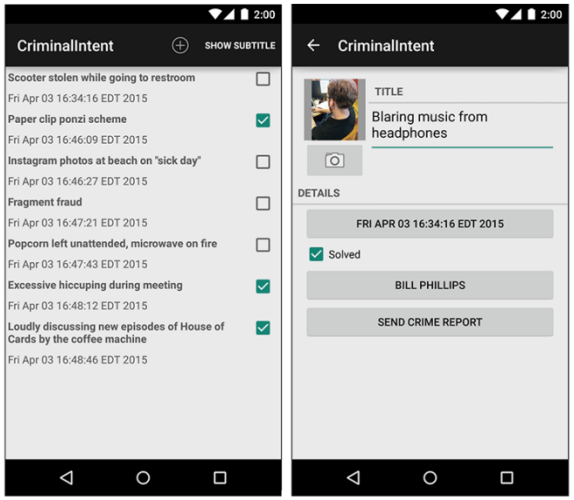


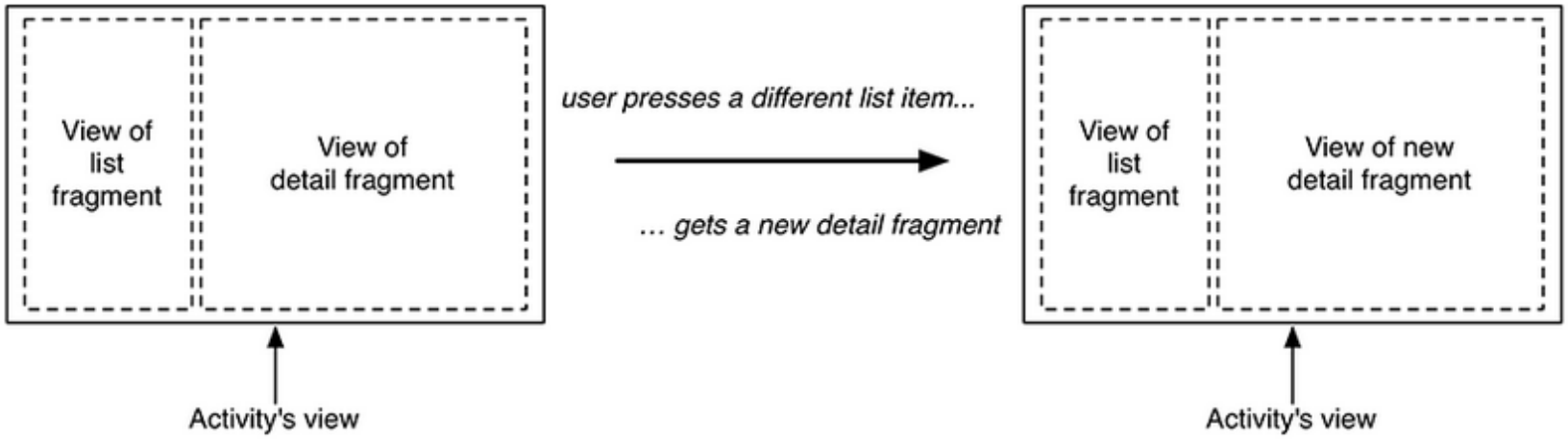


- **On tablet:** show list + detail
- **On phone:** swipe to show next crime

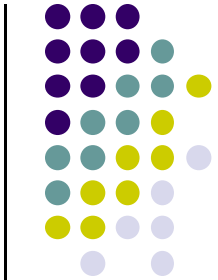
Fragments

- Activities can contain multiple fragments
- Fragment's views are inflated from a layout file
- Can rearrange fragments as desired on an activity
  - i.e. different arrangement on phone vs tablet

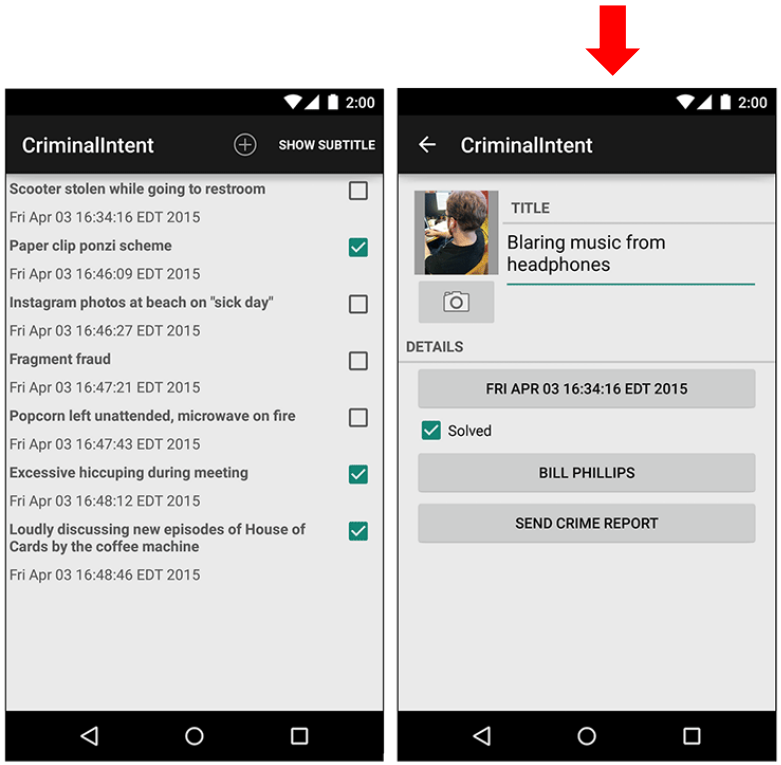




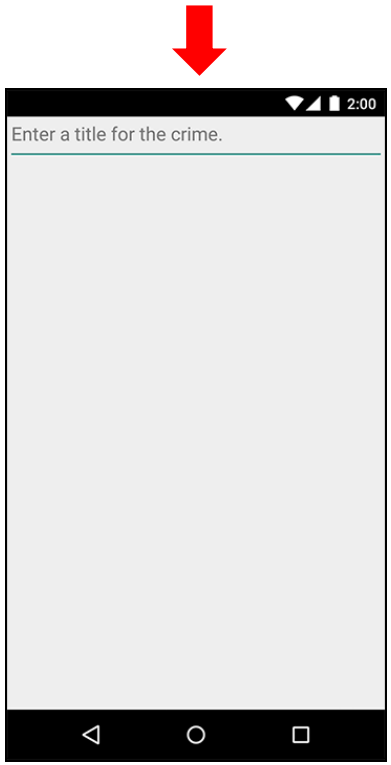
Starting Criminal Intent



- So, we will start by developing the detail view of **CriminalIntent** using Fragments

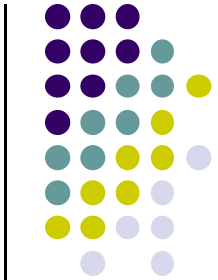


Final Look of CriminalIntent

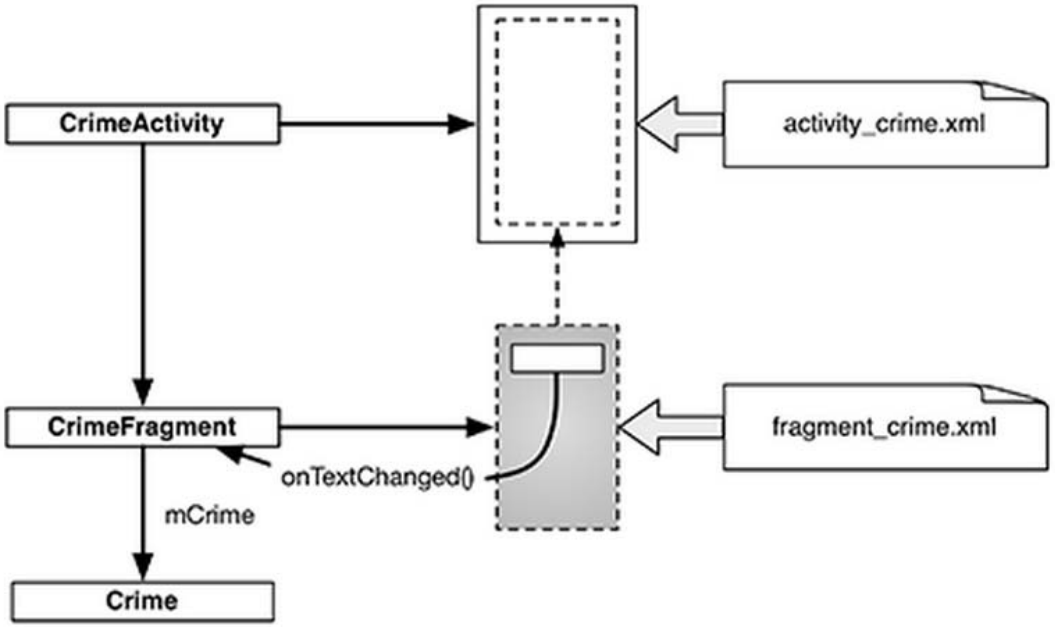


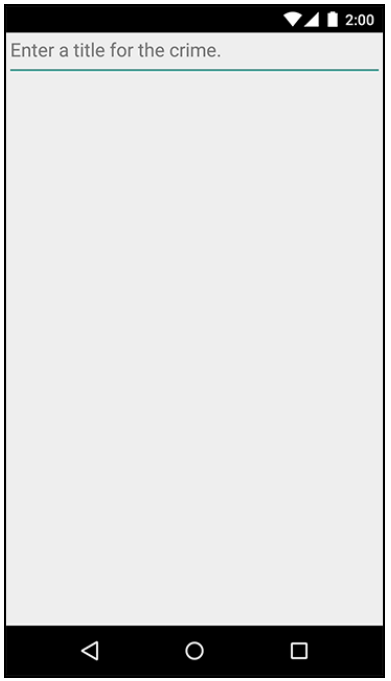
Start by Developing detail view using Fragments

Starting Criminal Intent

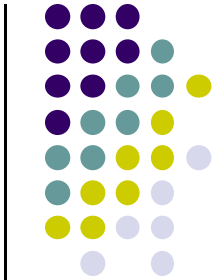


- Detail screen shown will be managed by a UI fragment called **CrimeFragment**
- An instance of **CrimeFragment** will be hosted by an activity called **CrimeActivity**
- **Hosted? CrimeActivity** provides a spot for **CrimeFragment** in its layout

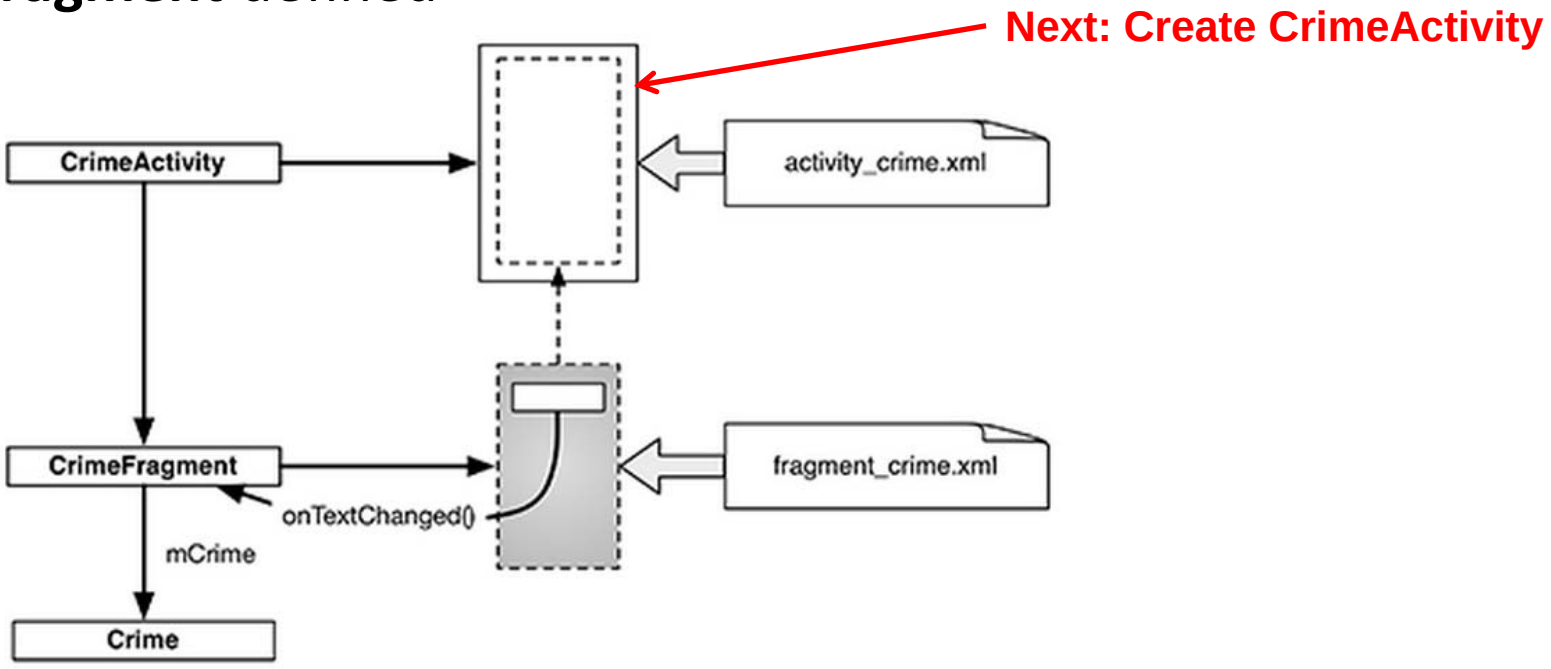




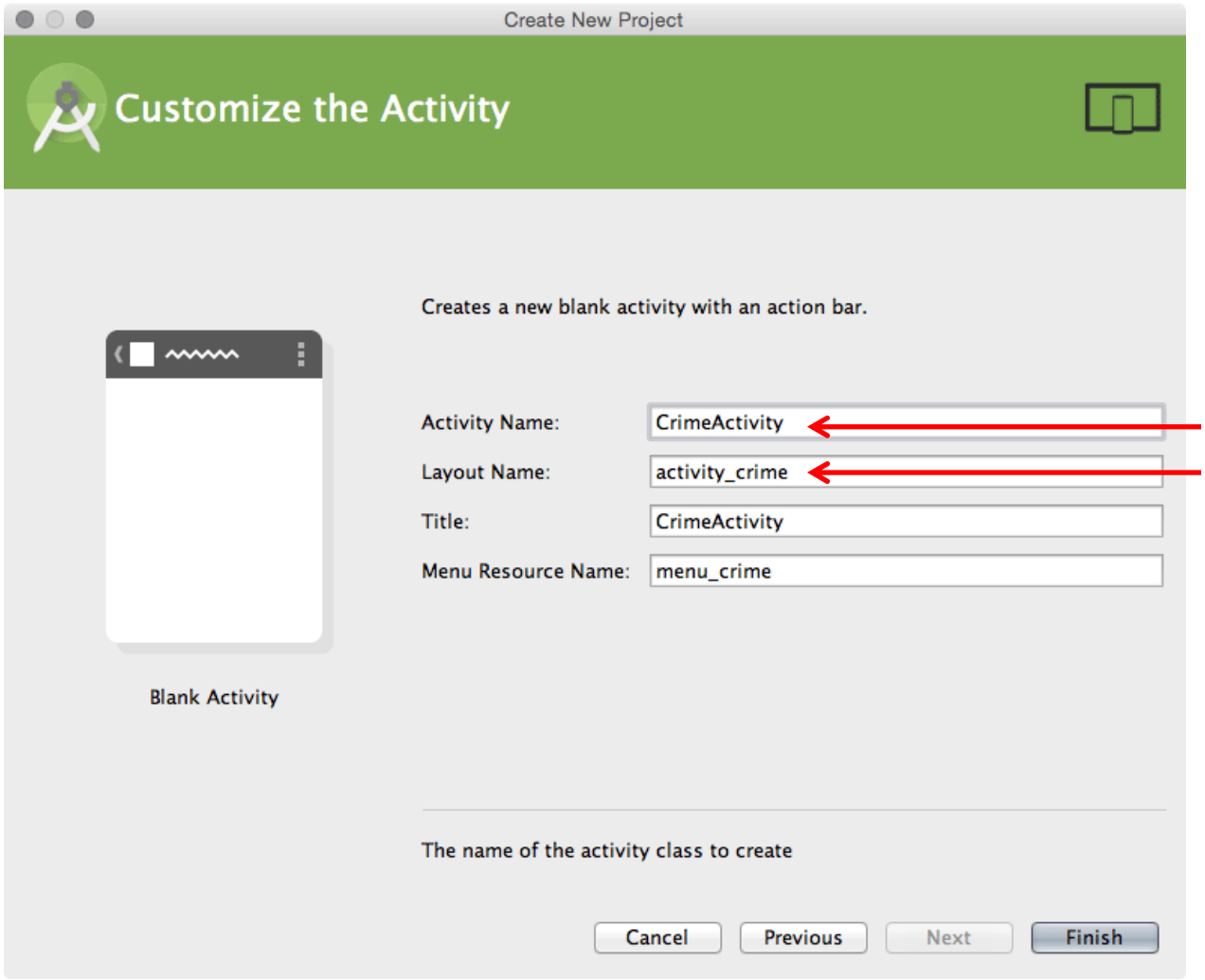
Starting Criminal Intent



- **Crime:** holds record of single office crime. Has
  - **Title** e.g. “Someone stole my yogurt!”
  - **ID:** uniquely identifies crime
- **CrimeFragment** has member variable **mCrime** to hold crimes
- **CrimeActivity** has a FrameLayout with position of **CrimeFragment** defined



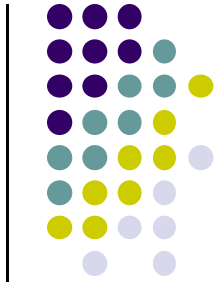
Create CrimeActivity in Android Studio



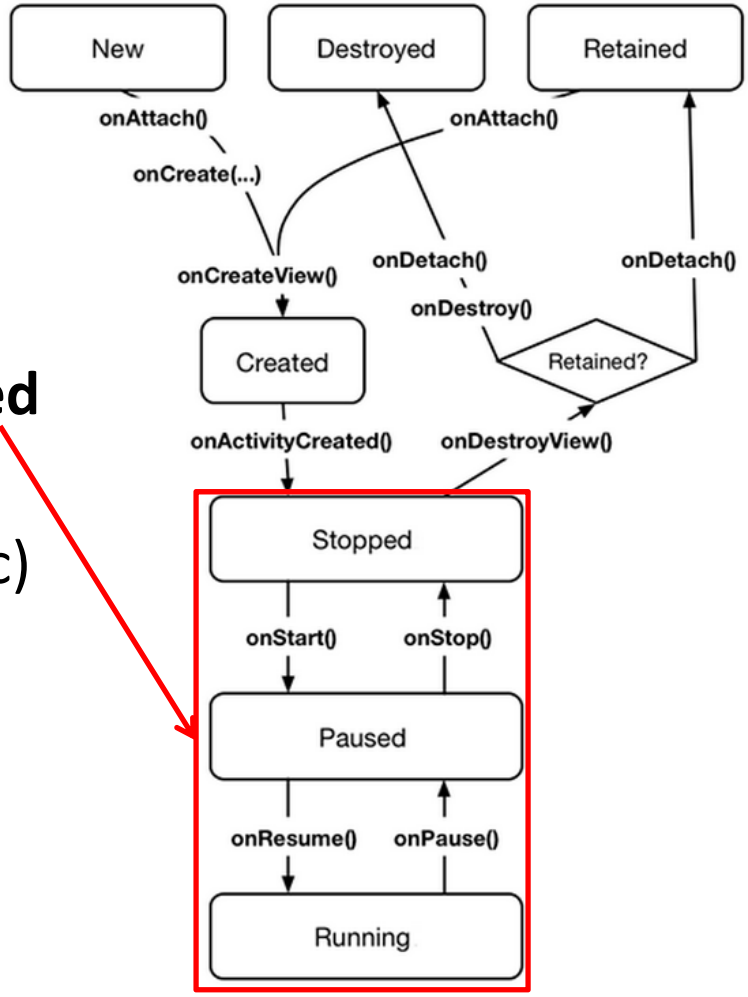
Creates CrimeActivity.java

Formatted using activity_crime.xml

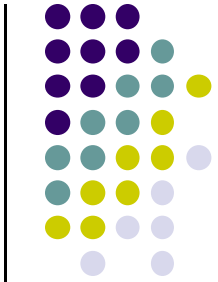
Hosting a UI Fragment



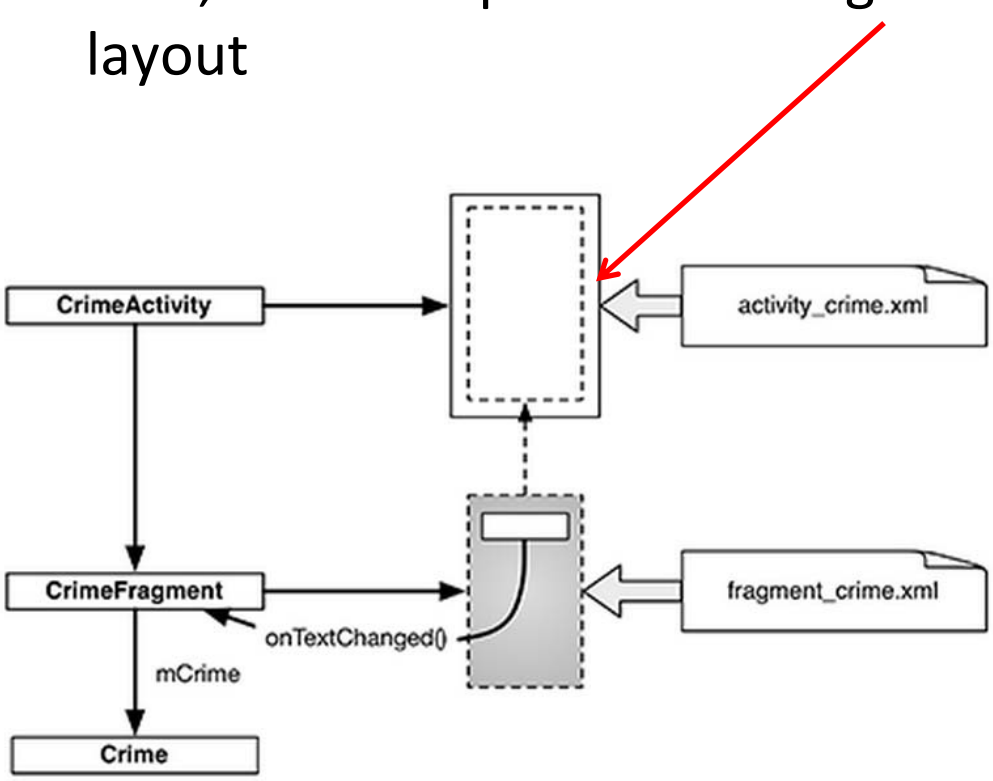
- To host a UI fragment, an activity must
  - Define a spot in its layout for the fragment's view
  - Manage the lifecycle of the fragment instance
- Fragment's lifecycle somewhat similar to activity lifecycle
- Has states **running**, **paused** and **stopped**
- Also has some similar activity lifecycle methods (e.g. **onPause()**, **onStop( )**, etc)
- **Key difference:**
  - Fragment's lifecycle's methods **called by hosting activity NOT Android OS!**



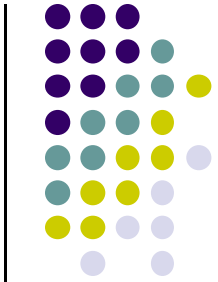
Hosting UI Fragment in an Activity



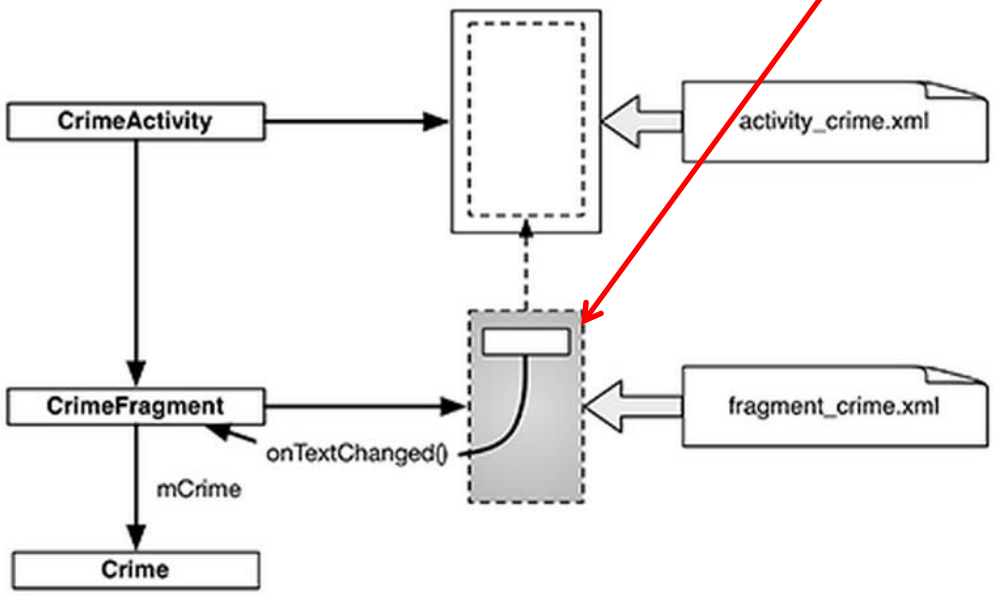
- 2 options. Can add fragment either
  - To **Activity's XML file (layout fragment),** or
  - In the **activity's .java file** (more complex but more flexible)
- We will add fragment to activity's .java file now
- First, create a spot for the fragment's view in **CrimeActivity's** layout



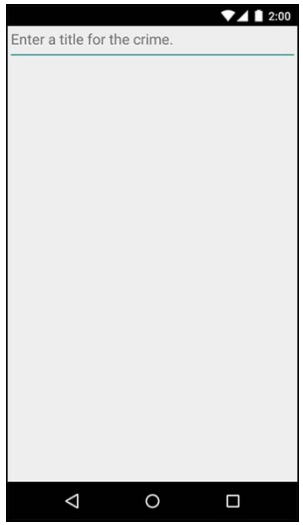
Creating a UI Fragment



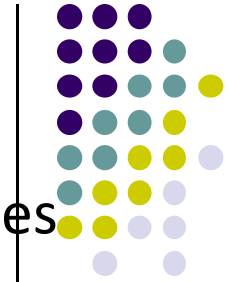
- Creating Fragment is similar to creating activity
  1. Define widgets in a layout file
  2. Create class and specify its view as layout above
  3. Wire up widget inflated from layout in code
- Defining layout file for **CrimeFragment (fragment_crime.xml)**




```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
  android:layout_width="match_parent"
  android:layout_height="match_parent"
  android:orientation="vertical"
  >
  <EditText android:id="@+id/crime_title"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:hint="@string/crime_title_hint"
    />
</LinearLayout>
```



Implementing Fragment Lifecycle Methods



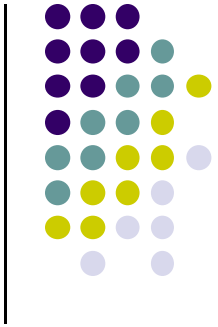
- **CrimeFragment** presents details of a specific crime + updates
- Override CrimeFragment's **onCreate( )** function

Declared public so that It can be called by any Activity hosting the fragment

```
public class CrimeFragment extends Fragment {
    private Crime mCrime;

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        mCrime = new Crime();
    }

    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container,
            Bundle savedInstanceState) {
        View v = inflater.inflate(R.layout.fragment_crime, container, false);
        return v;
    }
}
```

- **Note:** Fragment's view not inflated in **Fragment.onCreate( )**
- Fragment's view created and configured in another fragment lifecycle method (**onCreateView**)

Wiring up the EditText Widget



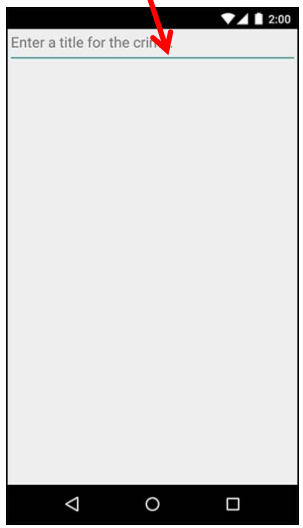
```
public class CrimeFragment extends Fragment {
    private Crime mCrime;
    private EditText mTitleField;

    ...

    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container,
            Bundle savedInstanceState) {
        View v = inflater.inflate(R.layout.fragment_crime, container, false);

        mTitleField = (EditText)v.findViewById(R.id.crime_title);
        mTitleField.addTextChangedListener(new TextWatcher() {
            @Override
            public void beforeTextChanged(
                CharSequence s, int start, int count, int after) {
                // This space intentionally left blank
            }

            @Override
            public void onTextChanged(
                CharSequence s, int start, int before, int count) {
                mCrime.setTitle(s.toString());
            }

            @Override
            public void afterTextChanged(Editable s) {
                // This one too
            }
        });

        return v;
    }
}
```

Find EditText widget

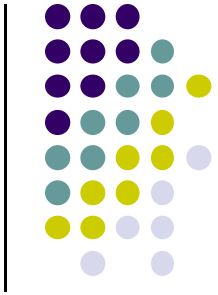
Add listener for text change event

User's input

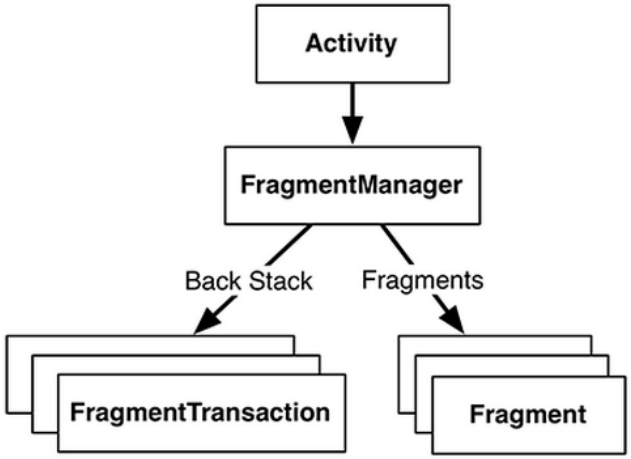
EditText widget



Adding UI Fragment to FragmentManager



- Finally, we add fragment just created to **FragmentManager**
- **FragmentManager**
  - Manages fragments
  - Adds their views to activity's view
  - Handles
    - List of fragment
    - Back stack of fragment transactions



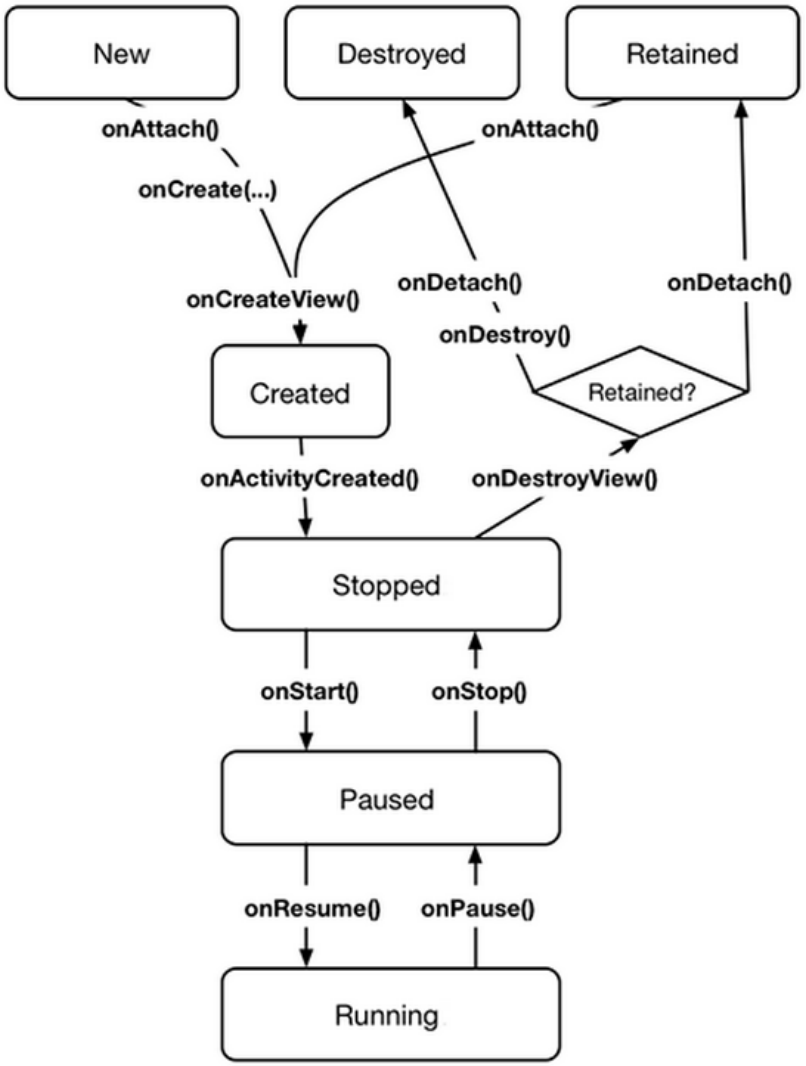

```
public class CrimeActivity extends FragmentActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_crime);

        FragmentManager fm = getSupportFragmentManager();
        Fragment fragment = fm.findFragmentById(R.id.fragment_container);

        if (fragment == null) {
            fragment = new CrimeFragment();
            fm.beginTransaction()
                .add(R.id.fragment_container, fragment)
                .commit();
        }
    }
}
```

Find Fragment using its ID

Interactions with FragmentManager are done using transactions

Add Fragment to activity's view

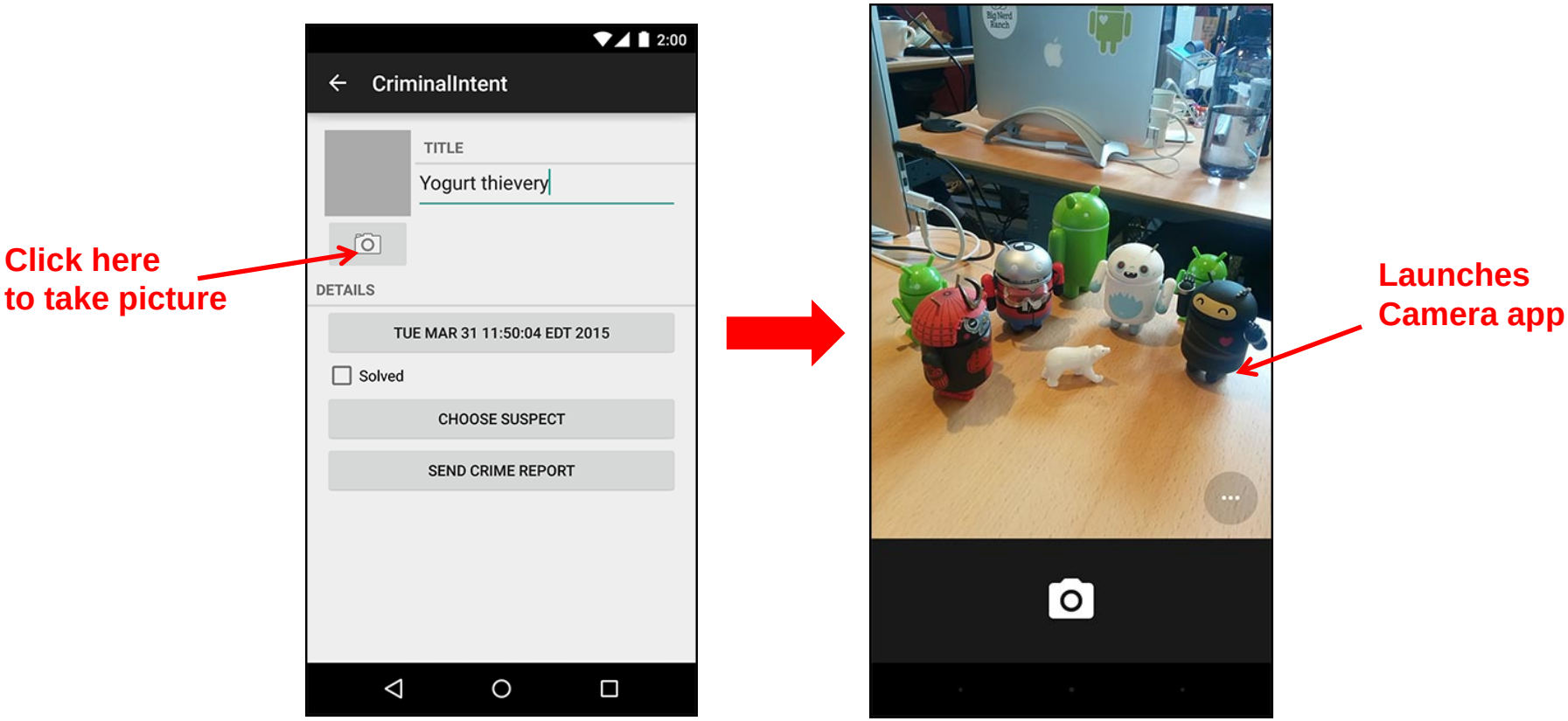
Examining Fragment's Lifecycle



- **FragmentManager** calls fragment lifecycle methods
- **onAttach( ), onCreate( )** and **onCreateView()** called when a fragment is added to **FragmentManager**
- **onActivityCreated( )** called after hosting activity's **onCreate( )** method is executed
- If fragment is added to already running Activity then **onAttach( ), onCreate( ), onCreateView()**, **onActivityCreated( )**, **onStart( )** and then **onResume( )** called

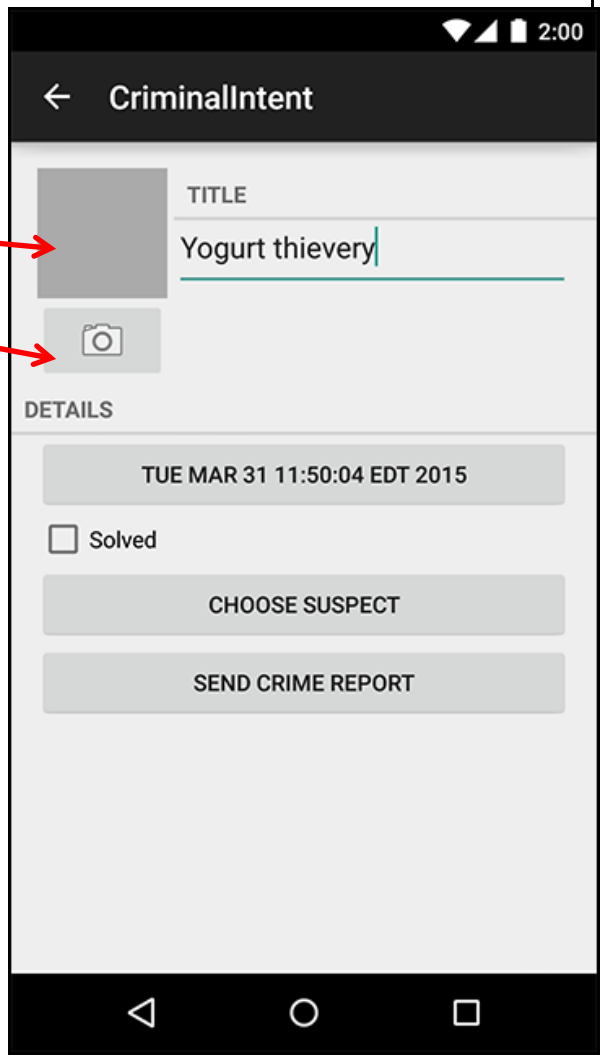
Taking Pictures with Intents
Ref: Ch 16 Android Nerd Ranch 2nd edition

- Would like to take picture of “Crime” to document it
- Use implicit intent to start Camera app from our CrimeIntent app
- **Recall:** Implicit intent used to call component in different activity

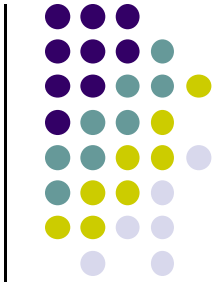


Create Placeholder for Picture

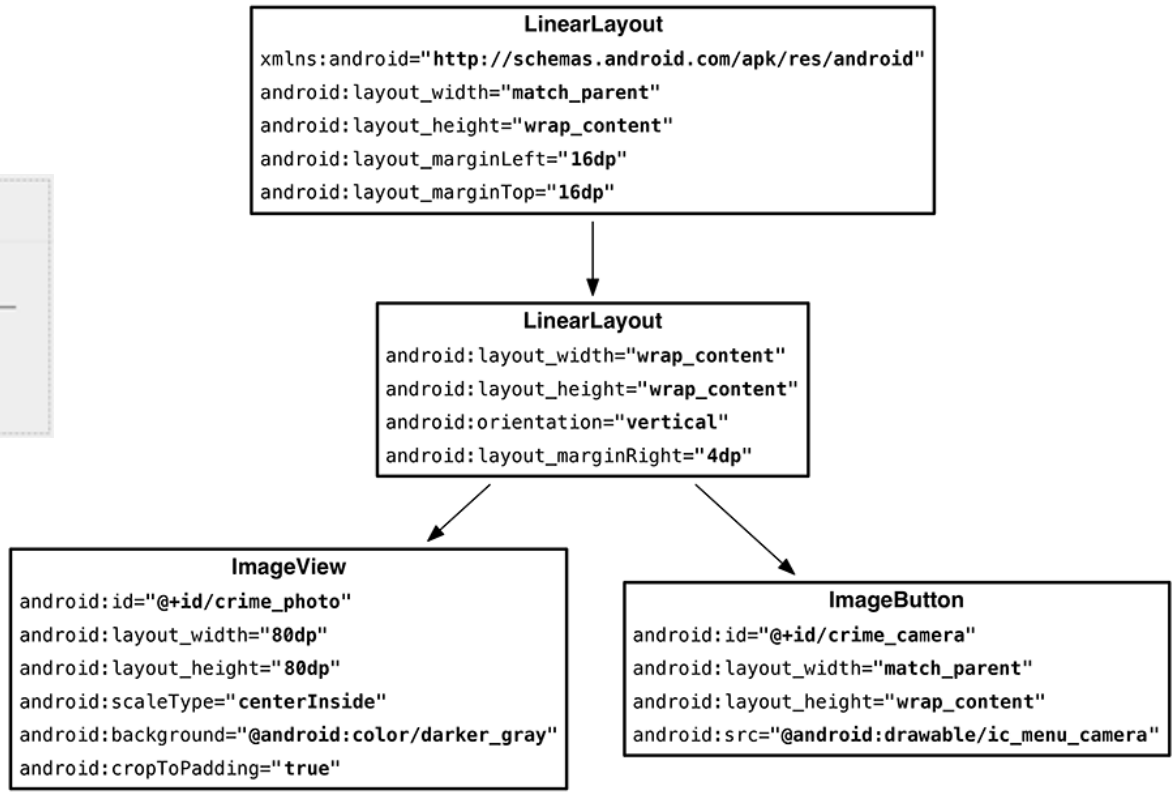
- Modify layout to include
  - ImageView for picture
  - Button to take picture



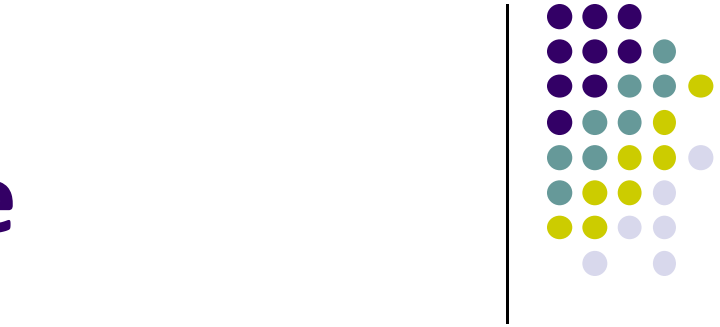
Create Camera and Title



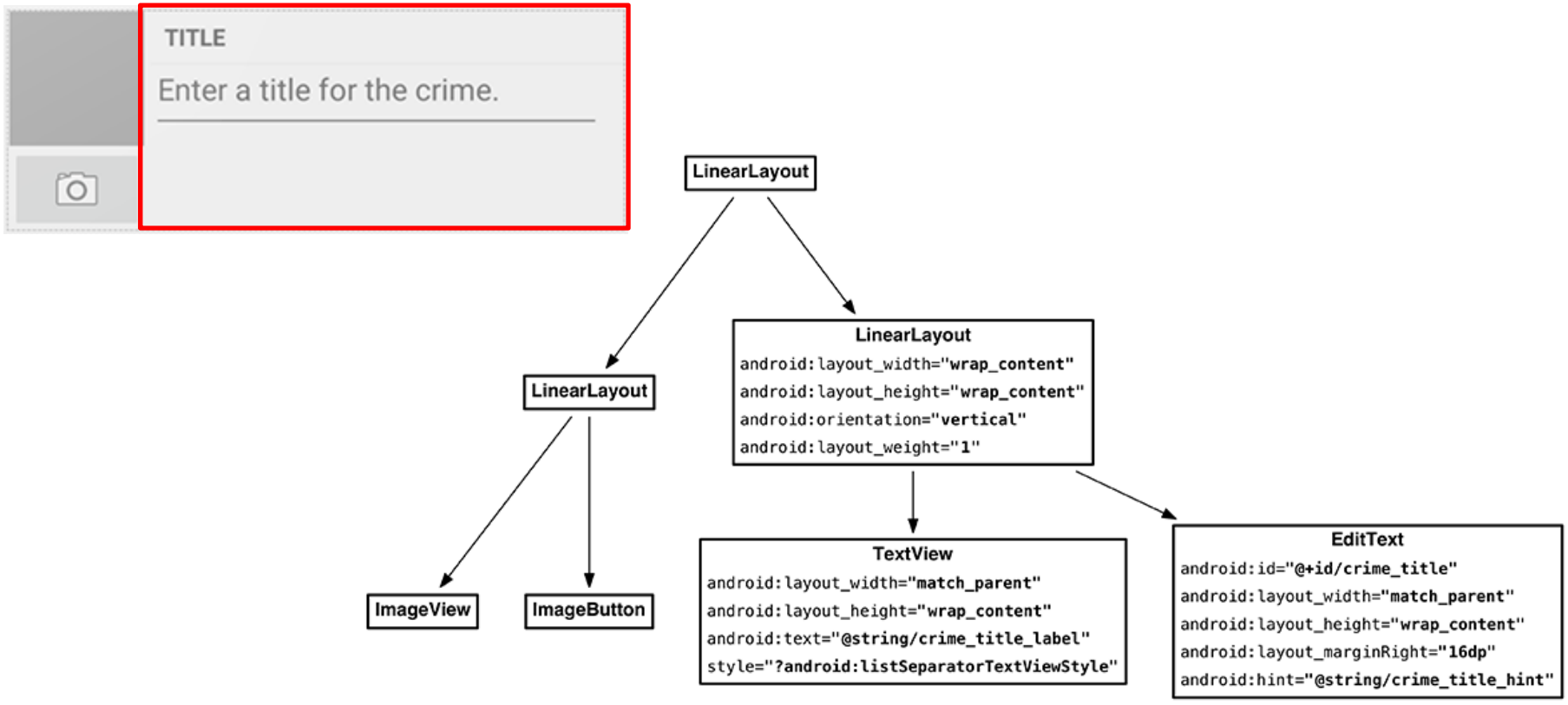
- Once created, we can include this in both landscape and portrait versions
- Store in: **res/layout/view_camera_and_title.xml**
- Build out left side



Create Camera and Title

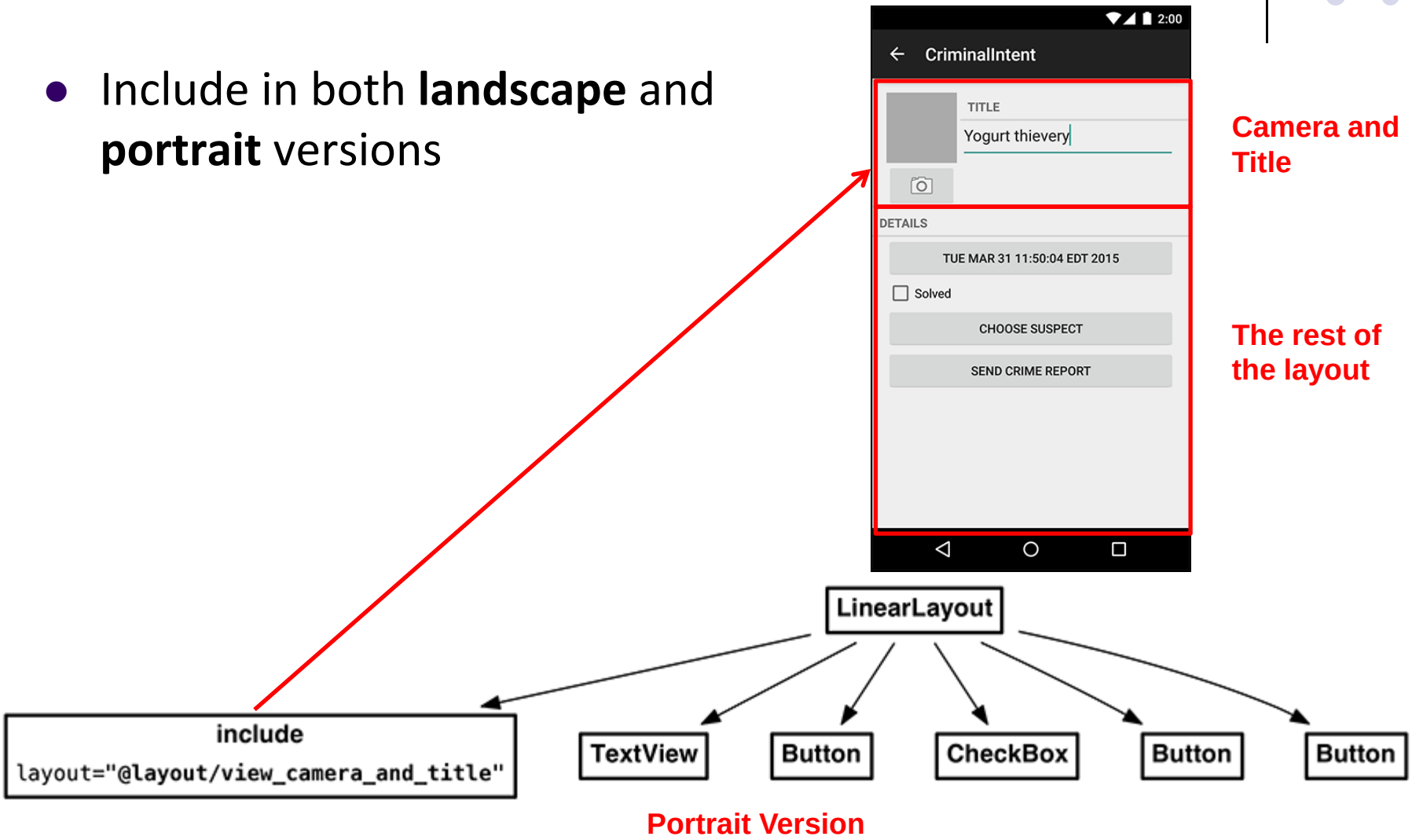


- Build out right side

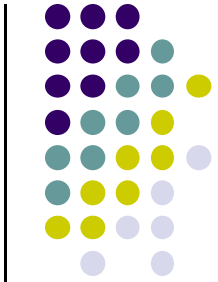


Include Camera and Title in Layout

- Include in both **landscape** and **portrait** versions



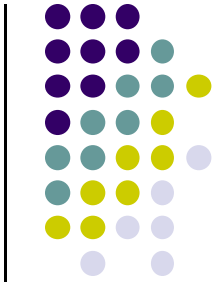
Get Handle of Camera Button and ImageView



- To respond to Camera Button click, in camera fragment, need handles to
  - Camera button
  - ImageView

```
...
private CheckBox mSolvedCheckbox;
private Button mSuspectButton;
private ImageButton mPhotoButton;
private ImageView mPhotoView;

...

@Override
public View onCreateView(LayoutInflater inflater, ViewGroup container,
                         Bundle savedInstanceState) {
    ...

    PackageManager packageManager = getActivity().getPackageManager();
    if (packageManager.resolveActivity(pickContact,
            PackageManager.MATCH_DEFAULT_ONLY) == null) {
        mSuspectButton.setEnabled(false);
    }

    mPhotoButton = (ImageButton) v.findViewById(R.id.crime_camera);
    mPhotoView = (ImageView) v.findViewById(R.id.crime_photo);

    return v;
}

...
```

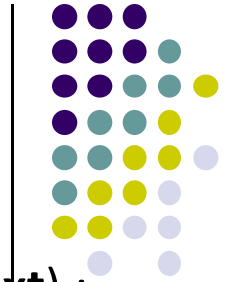
External Storage



- Would like to store pictures taken in phone's file system
- Two kinds of external storage
  - Primary (usually on device itself)
  - Everything else
- Primary location returned by

  Environment.getExternalStorageDirectory( )

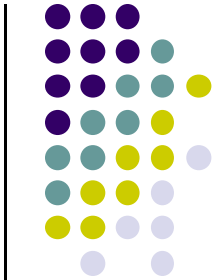
Storing New Images from Camera



- Basic methods for accessing phone's filesystem and directory (in **Context**) :

Method	Purpose
`File getExternalCacheDir()`	Returns a handle to a cache folder in primary external storage. Treat it like you do `getCacheDir()`, except a little more carefully. Android is even less likely to clean up this folder than the private storage one.
`File[] getExternalCacheDirs()`	Returns cache folders for multiple external storage locations.
`File getExternalFilesDir(String)`	Returns a handle to a folder on primary external storage in which to store regular files. If you pass in a type `String`, you can access a specific subfolder dedicated to a particular type of content. Type constants are defined in `Environment`, where they are prefixed with `DIRECTORY_`. For example, pictures go in `Environment.DIRECTORY_PICTURES`.
`File[] getExternalFilesDirs(String)`	Same as `getExternalFilesDir(String)`, but returns all possible file folders for the given type.
`File[] getExternalMediaDirs()`	Returns handles to all the external folders Android makes available for storing media – pictures, movies, and music. What makes this different from calling `getExternalFilesDir(Environment.DIRECTORY_PICTURES)` is that the media scanner automatically scans this folder. The media scanner makes files available to applications that play music, or browse movies and photos, so anything that you put in a folder returned by `getExternalMediaDirs()` will automatically appear in those apps.

Designing Picture Location



- Add method to build a filename for your picture

```
...

public void setSuspect(String suspect) {
    mSuspect = suspect;
}

public String getPhotoFilename() {
    return "IMG_" + getId().toString() + ".jpg";
}
}
```

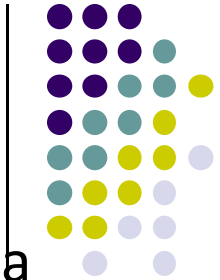
Get crime's ID and converts it to a string

- Find location for the photo

```
public class CrimeLab {
    ...

    public Crime getCrime(UUID id) {
        ...
    }

    public File getPhotoFile(Crime crime) {
        File externalFilesDir = mContext
                .getExternalFilesDir(Environment.DIRECTORY_PICTURES);

        if (externalFilesDir == null) {
            return null;
        }

        return new File(externalFilesDir, crime.getPhotoFilename());
    }

    ...
```

Note: Calling getPhotoFile will return location to store a picture

Using Camera Intent



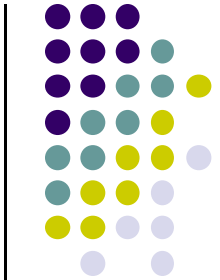
- From CrimeFragment.java, get a file location and store it in a variable **mPhotoFile**

```
...

private Crime mCrime;
private File mPhotoFile;
private EditText mTitleField;
...

@Override
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    UUID crimeId = (UUID) getArguments().getSerializable(ARG_CRIME_ID);
    mCrime = CrimeLab.get(getActivity()).getCrime(crimeId);
    mPhotoFile = CrimeLab.get(getActivity()).getPhotoFile(mCrime);
}

...
```

- Add external storage permissions to Android manifest

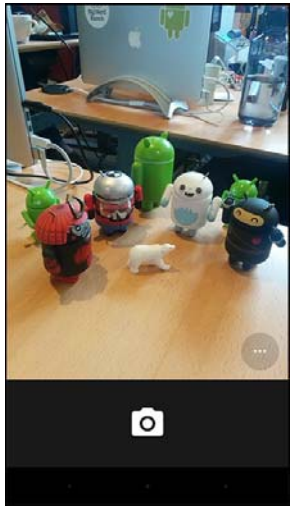
```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.bignerdranch.android.criminalintent" >

    <uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE"
                     android:maxSdkVersion="18"
        />
    ...
```

Firing the Intent



- **MediaStore** defines public interface for interacting with media (images, video and music)
  - Including image capture intent
- **ACTION_CAPTURE_IMAGE** is action that
  - Fires up camera application
  - Takes picture
- Picture taken is:
  - Small resolution thumbnail
  - Placed inside **Intent** object returned in **getActivityResult( )**
- If full resolution image is desired, location to save file on filesystem needs to be specified

Firing Camera Intent



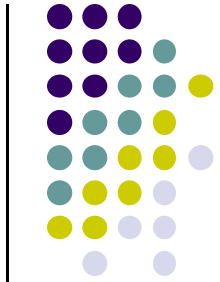
```
...

private static final int REQUEST_DATE = 0;
private static final int REQUEST_CONTACT = 1;
private static final int REQUEST_PHOTO= 2;

...

@Override
public View onCreateView(LayoutInflater inflater, ViewGroup container,
                         Bundle savedInstanceState) {
    ...

    mPhotoButton = (ImageButton) v.findViewById(R.id.crime_camera);
    final Intent captureImage = new Intent(MediaStore.ACTION_IMAGE_CAPTURE);

    boolean canTakePhoto = mPhotoFile != null &&
            captureImage.resolveActivity(packageManager) != null;
    mPhotoButton.setEnabled(canTakePhoto);

    if (canTakePhoto) {
        Uri uri = Uri.fromFile(mPhotoFile);
        captureImage.putExtra(MediaStore.EXTRA_OUTPUT, uri);
    }

    mPhotoButton.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            startActivityForResult(captureImage, REQUEST_PHOTO);
        }
    });

    mPhotoView = (ImageView) v.findViewById(R.id.crime_photo);

    return v;
}
```

Create new intent for image capture

Check with PackageManager that Camera exists on this phone

Take picture when button is clicked

Scaling and Displaying Bitmaps



- Want to load file as bitmap, show it to user
- Returned bitmap may be large, needs scaling
- Check raw size, scale if too large

```
public class PictureUtils {
    public static Bitmap getScaledBitmap(String path, int destWidth, int destHeight) {
        // Read in the dimensions of the image on disk
        BitmapFactory.Options options = new BitmapFactory.Options();
        options.inJustDecodeBounds = true;
        BitmapFactory.decodeFile(path, options);

        float srcWidth = options.outWidth;
        float srcHeight = options.outHeight;

        // Figure out how much to scale down by
        int inSampleSize = 1;
        if (srcHeight > destHeight || srcWidth > destWidth) {
            if (srcWidth > srcHeight) {
                inSampleSize = Math.round(srcHeight / destHeight);
            } else {
                inSampleSize = Math.round(srcWidth / destWidth);
            }
        }

        options = new BitmapFactory.Options();
        options.inSampleSize = inSampleSize;

        // Read in and create final bitmap
        return BitmapFactory.decodeFile(path, options);
    }
}
```

Used for bitmaps

Check if picture is too large,
Determine scaling factor

Set determined scaling factor

Get image file at location “path”,
Scale using determined size

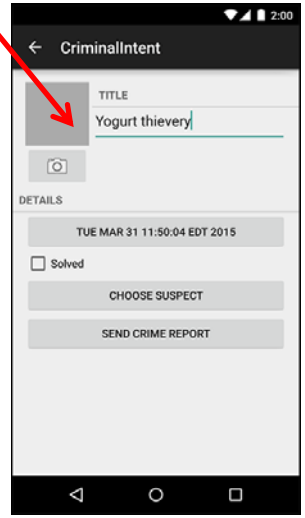
Scaling and Displaying Bitmaps

- Better to scale PhotoView based on size of hosting activity at runtime

```
public class PictureUtils {
    public static Bitmap getScaledBitmap(String path, Activity activity) {
        Point size = new Point();
        activity.getWindowManager().getDefaultDisplay()
                .getSize(size);

        return getScaledBitmap(path, size.x, size.y);
    }
```

Get activity size

PhotoView



- After scaling bitmap, load it into **PhotoView**

```
    ...

    private String getCrimeReport() {
        ...
    }

    private void updatePhotoView() {
        if (mPhotoFile == null || !mPhotoFile.exists()) {
            mPhotoView.setImageDrawable(null);
        } else {
            Bitmap bitmap = PictureUtils.getScaledBitmap(
                    mPhotoFile.getPath(), getActivity());
            mPhotoView.setImageBitmap(bitmap);
        }
    }
}
```

Load scaled bitmap into photoview

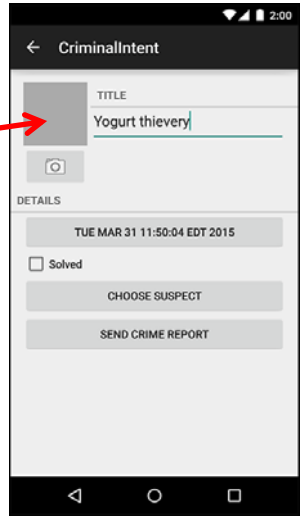
Scaling and Displaying Bitmaps

- Call this method from inside **onCreateView( )**

```
mPhotoButton.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        startActivityForResult(captureImage, REQUEST_PHOTO);
    }
});

mPhotoView = (ImageView) v.findViewById(R.id.crime_photo);
updatePhotoView();

return v;
}
```

Take a picture when button is clicked

Write image to photoView

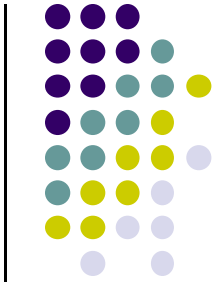


Declaring Features

- Declaring “uses-features” in Android manifest means only cameras with that feature will “see” this app for download on the app store
- E.g. declaring “uses-feature… android.hardware.camera”, only phones with cameras will see this for download

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.bignerdranch.android.criminalintent" >

    <uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE"
                     android:maxSdkVersion="18"
        />
    <uses-feature android:name="android.hardware.camera"
                  android:required="false"
        />

    ...
```

References



- Android Nerd Ranch (2nd edition)
- Android Nerd Ranch (1st edition)
- 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