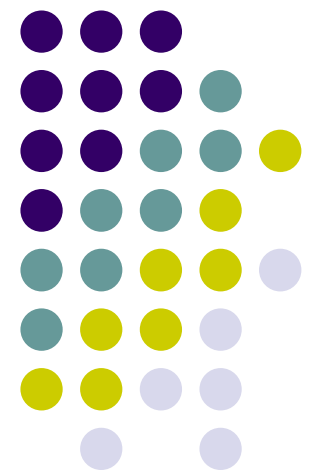


CS 403X Mobile and Ubiquitous Computing

Lecture 8: Fragments Camera

Emmanuel Agu



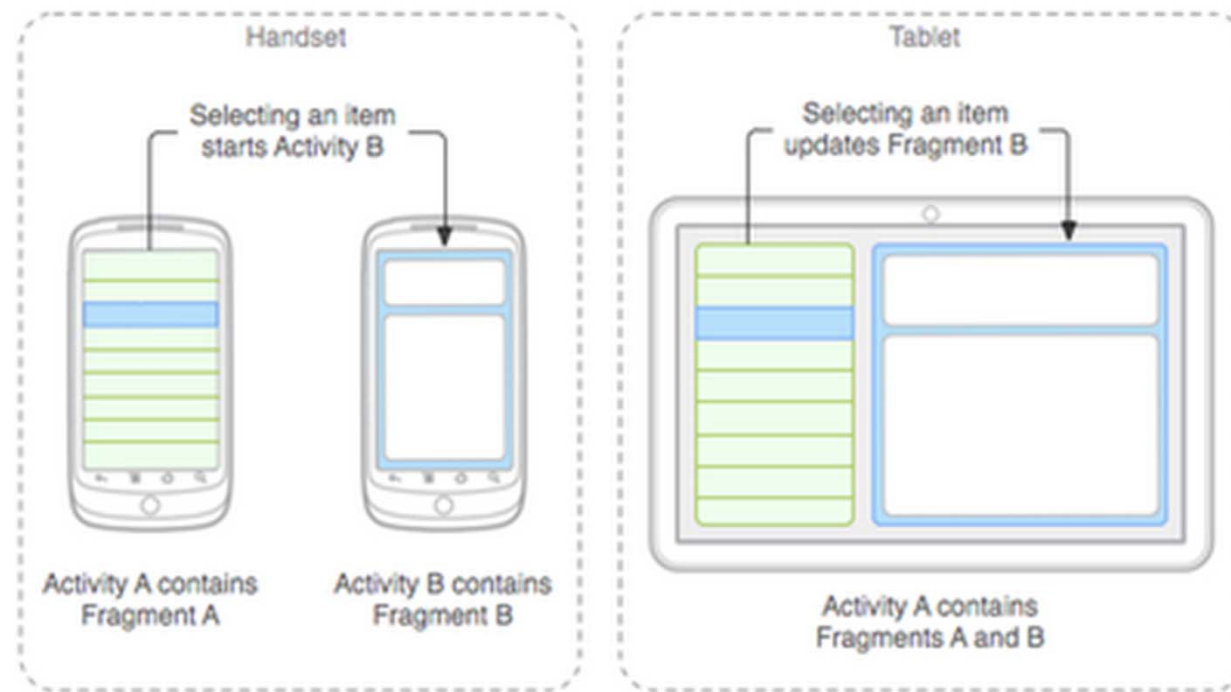


Fragments



Recall: Fragments

- Sub-components of an Activity (screen)
- An activity can contain multiple fragments, organized differently on different devices (e.g. phone vs tablet)
- Fragments need to be attached to Activities.

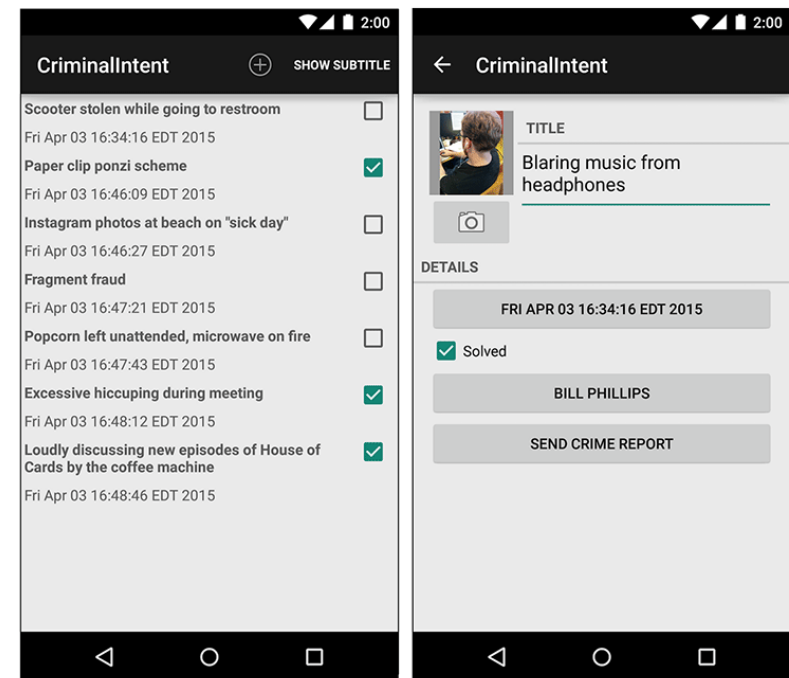
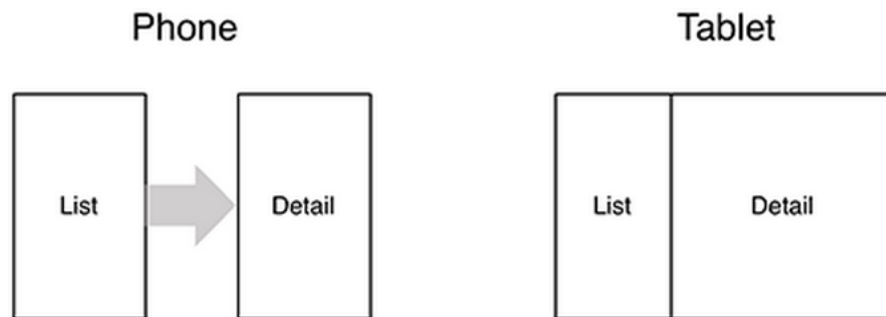


Fragments

Ref: Android Nerd Ranch (2nd ed), Ch 7, pg 121



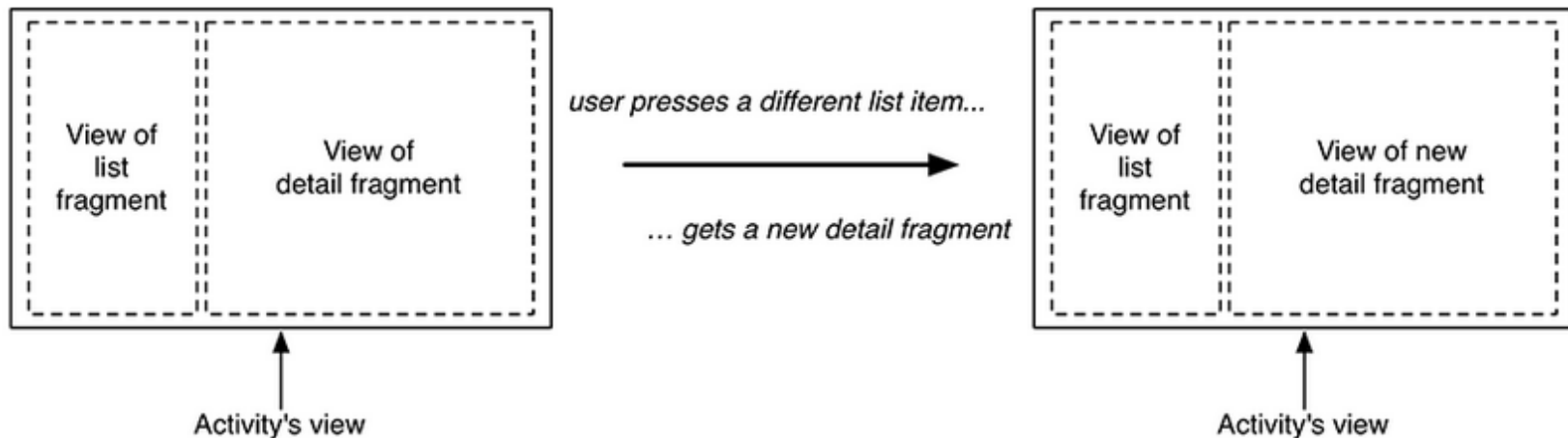
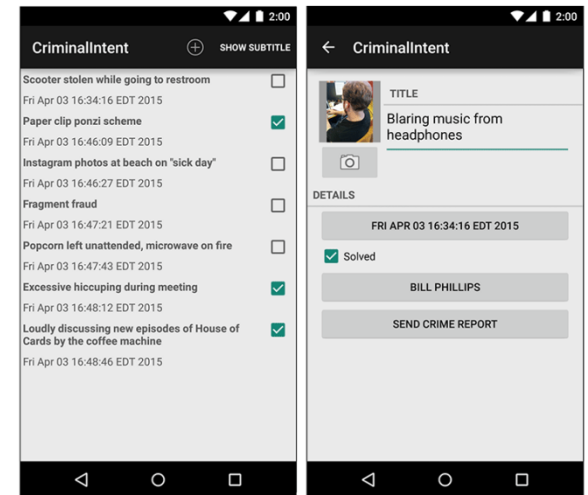
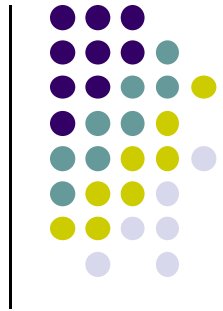
- To illustrate fragments, we create new app **CriminalIntent**
- Used to record “office crimes” e.g. leaving plates in sink, etc
- Record includes:
 - Title, date, photo
- List-detail app + Fragments



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

Fragments

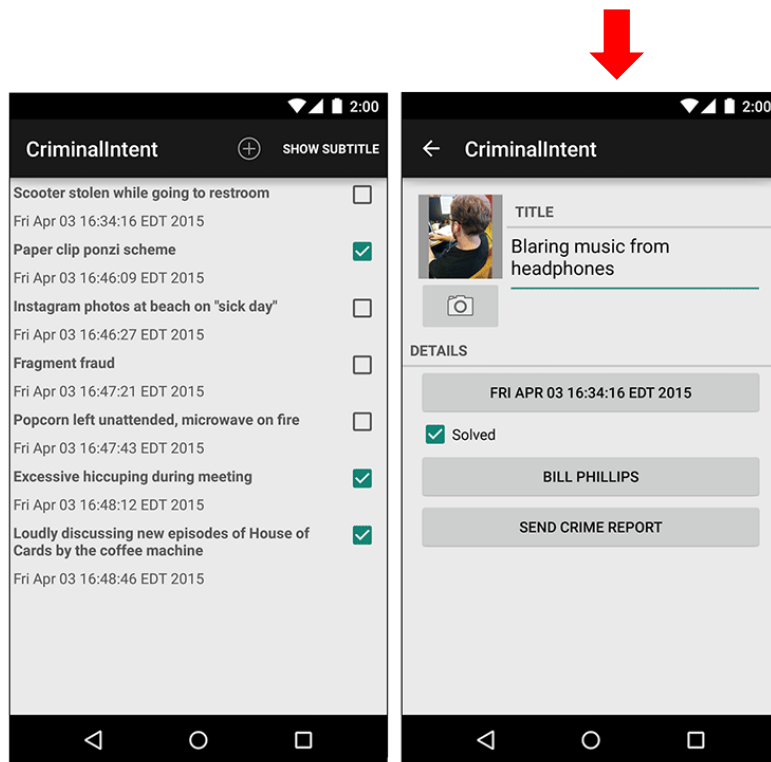
- Activities can contain multiple fragments
- Fragment's views are inflated from a layout file
- Can rearrange fragments as desired on an activity
 - i.e. different arrangement on phone vs tablet



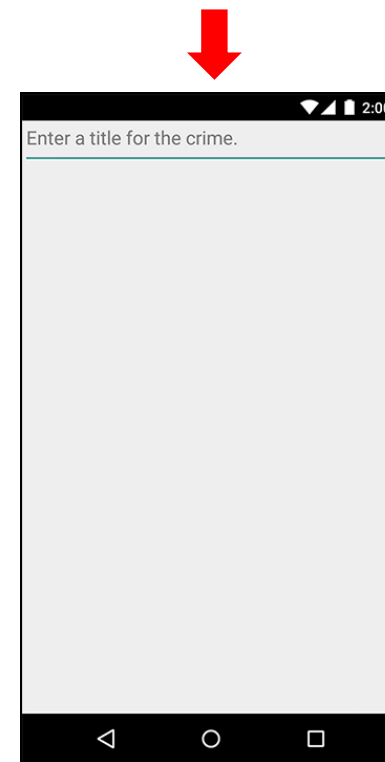
Starting Criminal Intent



- Initially, develop detail view of **CriminalIntent** using Fragments



Final Look of CriminalIntent



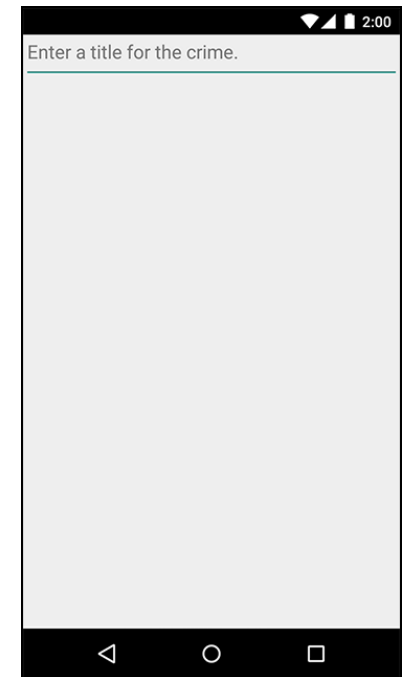
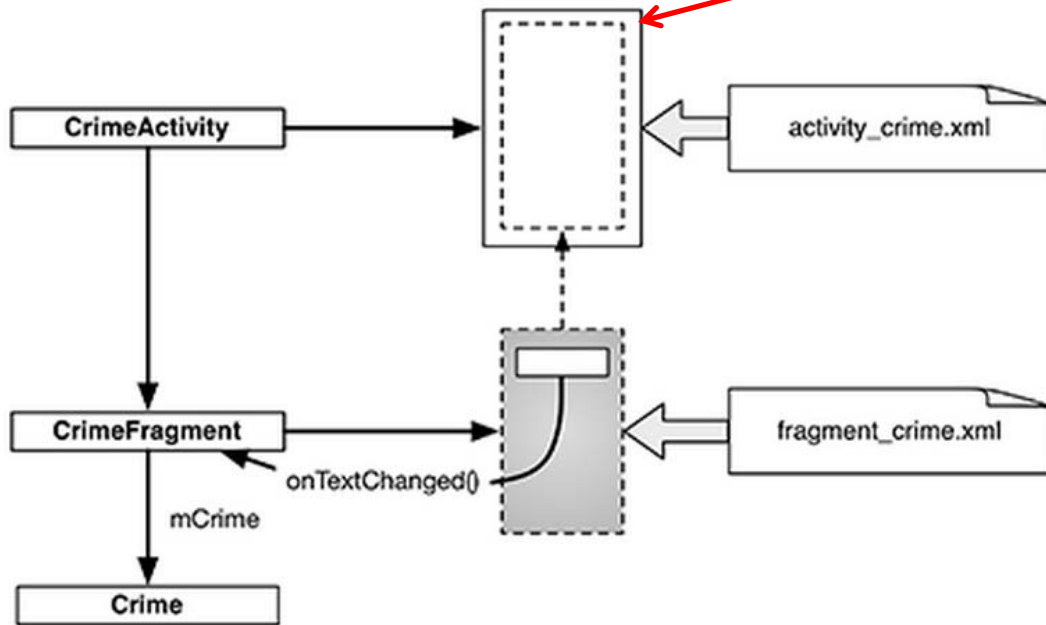
**Start small
Develop detail view using Fragments**

Starting Criminal Intent



- **CrimeFragment:** UI fragment to manage Detail screen
- **CrimeActivity:** Activity that contains **CrimeFragment**
- **Crime:** holds record of 1 office crime. Has
 - **Title** e.g. "Someone stole my yogurt!"
 - **ID:** uniquely identifies crime

Next: Create CrimeActivity



Create CrimeActivity in Android Studio



The screenshot shows the 'Create New Project' dialog in Android Studio, specifically the 'Customize the Activity' screen. The dialog has a green header with the Android logo and the text 'Customize the Activity'. Below the header, there is a preview of a 'Blank Activity' on the left. The main area contains the following fields:

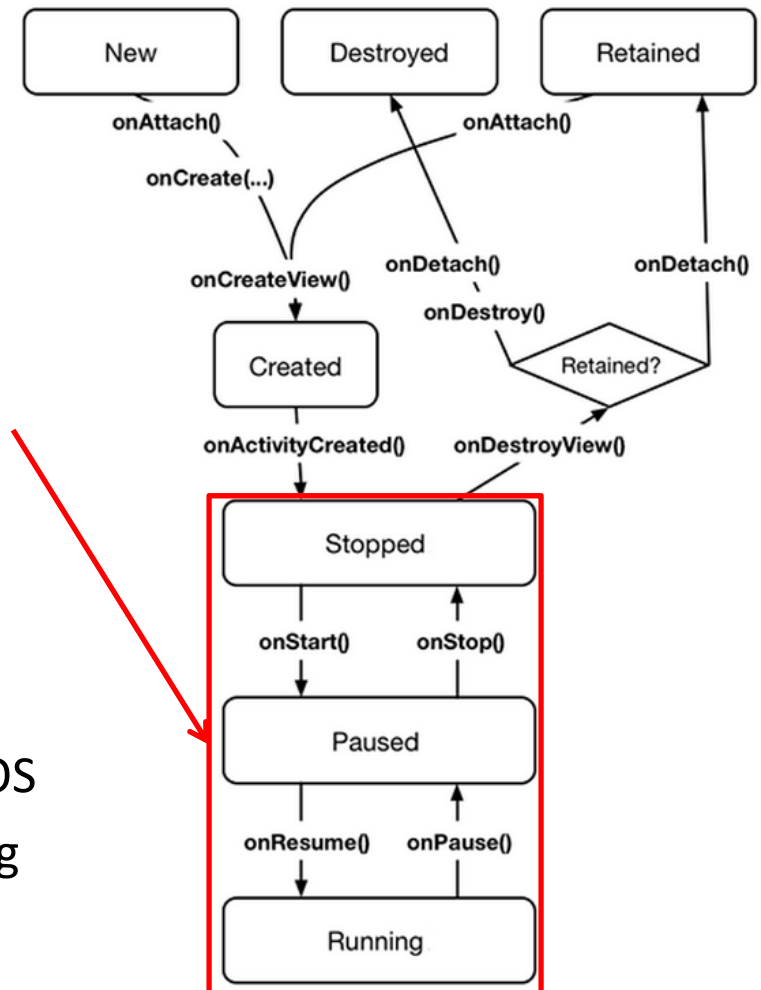
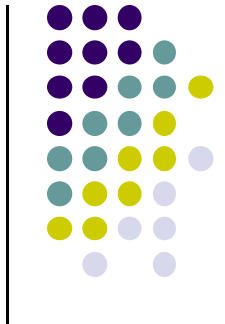
- Activity Name:
- Layout Name:
- Title:
- Menu Resource Name:

Below the fields, there is a section titled 'The name of the activity class to create' with a text input field. At the bottom, there are four buttons: 'Cancel', 'Previous', 'Next', and 'Finish'.

Creates CrimeActivity.java
Formatted using
activity_crime.xml

Hosting a UI Fragment

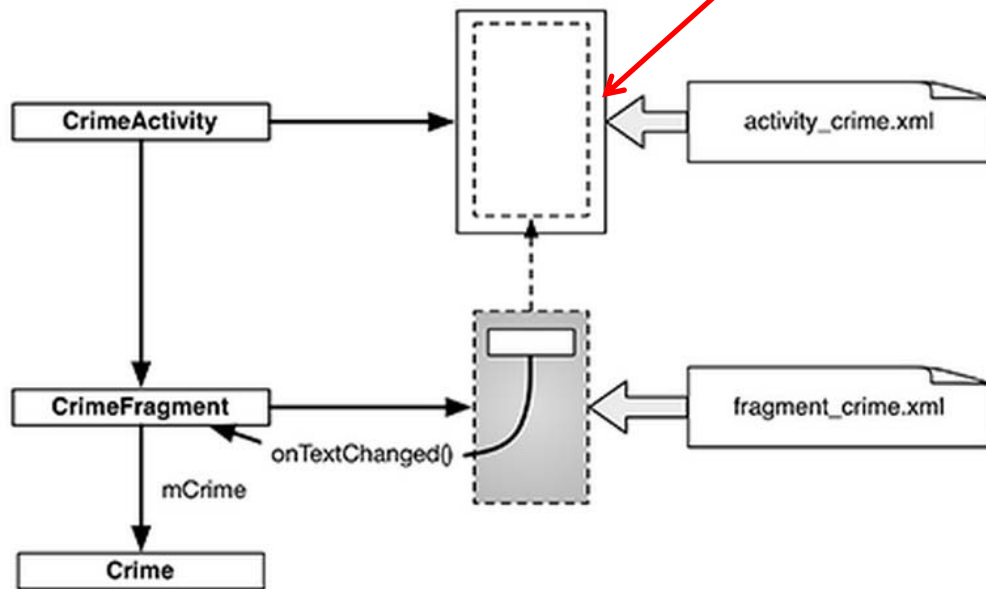
- To host a UI fragment, an activity must
 - Define a spot in its layout for the fragment
 - Manage the lifecycle of the fragment instance
- Fragment's lifecycle somewhat similar to activity lifecycle
 - Has states **running**, **paused** and **stopped**
 - Also has some similar activity lifecycle methods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





Hosting UI Fragment in an Activity

- 2 options. Can add fragment to either
 - **Activity's XML file (layout fragment),** or
 - **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



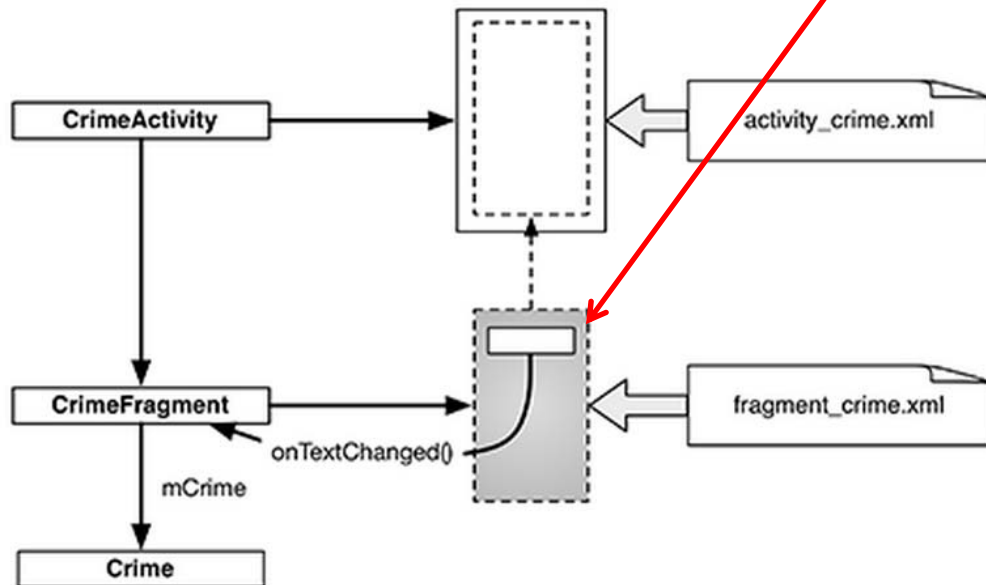
```
FrameLayout
xmlns:android="http://schemas.android.com/apk/res/android"
android:id="@+id/fragment_container"
android:layout_width="match_parent"
android:layout_height="match_parent"
```



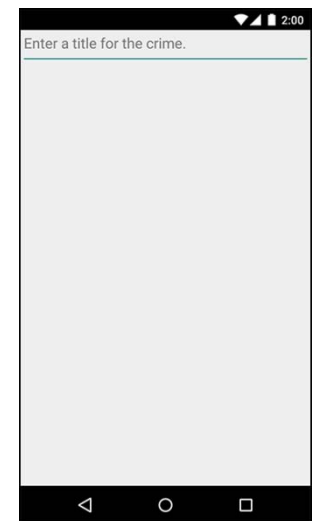
Creating a UI Fragment



- Creating Fragment is similar to creating activity
 1. Define widgets in a layout (XML) file
 2. Create java 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 CrimeFragment



- In **CrimeFragment** Override **CrimeFragment's onCreate()** function

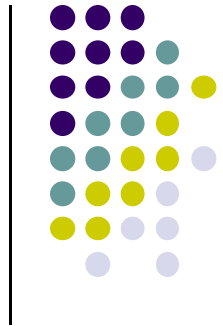
```
public class CrimeFragment extends Fragment {
    private Crime mCrime;

    @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 inflated in **Fragment.onCreateView()**, NOT **onCreate**

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
            }
        });

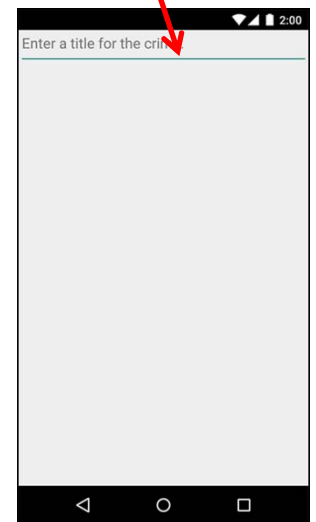
        return v;
    }
}
```

Find EditText widget

Add listener for text change event

Store user's input as Crime Title

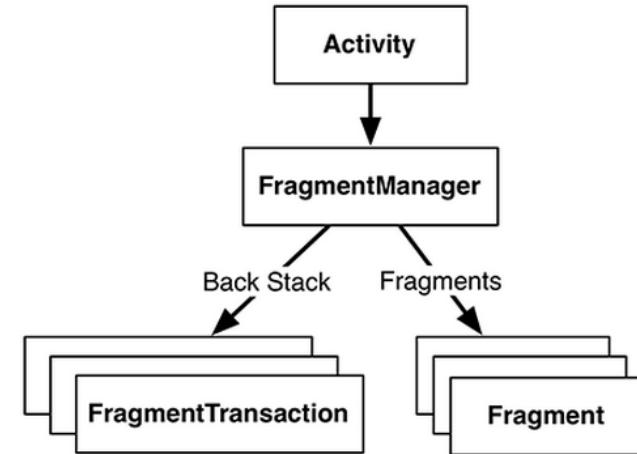
EditText widget



Adding UI Fragment to FragmentManager



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



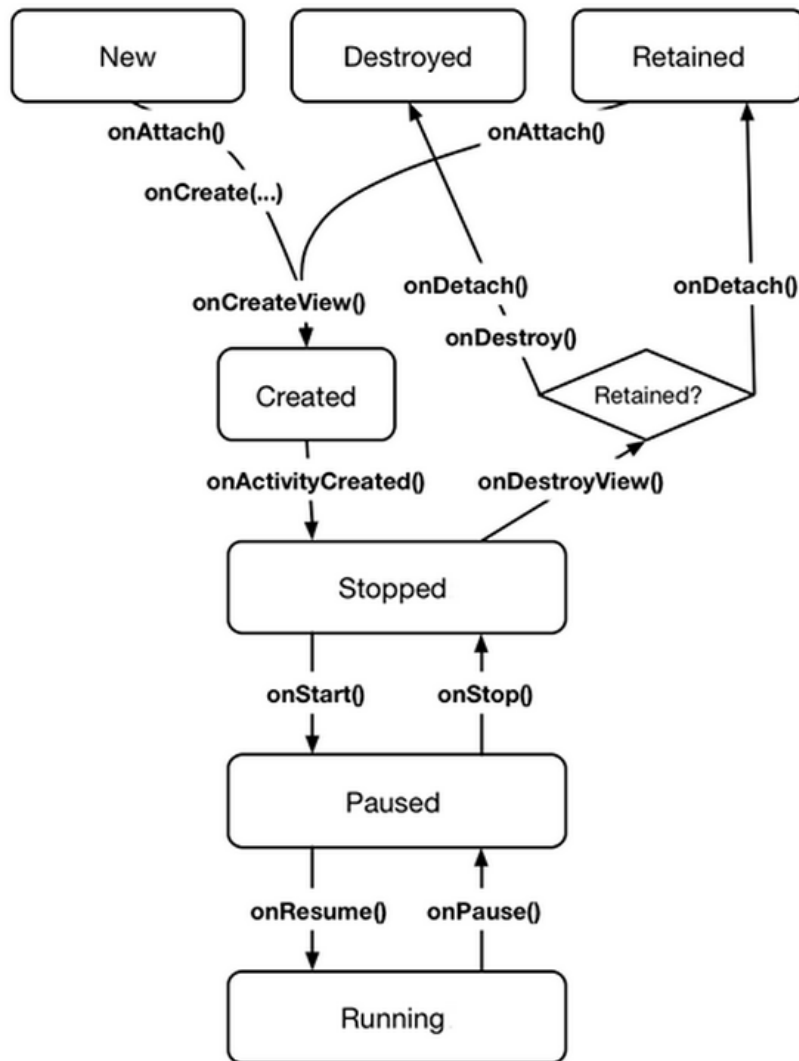
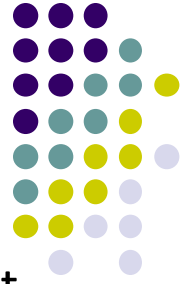
```
public class CrimeActivity extends FragmentActivity {  
  
    @Override  
    protected void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        setContentView(R.layout.activity_crime);  
  
        FragmentManager fm = getSupportFragmentManager();  
        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



Simply Taking Pictures

Android Camera



- How to take photos from your app using existing Android Camera app
- Steps:
 1. Request Camera Permission
 2. Take a Photo with the Camera App
 3. Get the Thumbnail
 4. Save the Full-size Photo



Request Camera Permission

- If your app takes pictures using Android Camera, on Google Play, make your app visible only to devices with a camera

```
<manifest ... >  
  <uses-feature android:name="android.hardware.camera"  
               android:required="true" />  
  ...  
</manifest>
```



Take a Photo with the Camera App

- To take picture, your app needs to send **Intent** to Android's Camera app, (i.e. action is capture an image)
- Check that at least 1 Activity that can handle request to take picture using **resolveActivity**
- Call **startActivityForResult()** with Camera intent

```
static final int REQUEST_IMAGE_CAPTURE = 1;
```

Build Intent describing taking a picture

```
private void dispatchTakePictureIntent() {  
    Intent takePictureIntent = new Intent(MediaStore.ACTION_IMAGE_CAPTURE);  
    if (takePictureIntent.resolveActivity(getPackageManager()) != null) {  
        startActivityForResult(takePictureIntent, REQUEST_IMAGE_CAPTURE);  
    }  
}
```

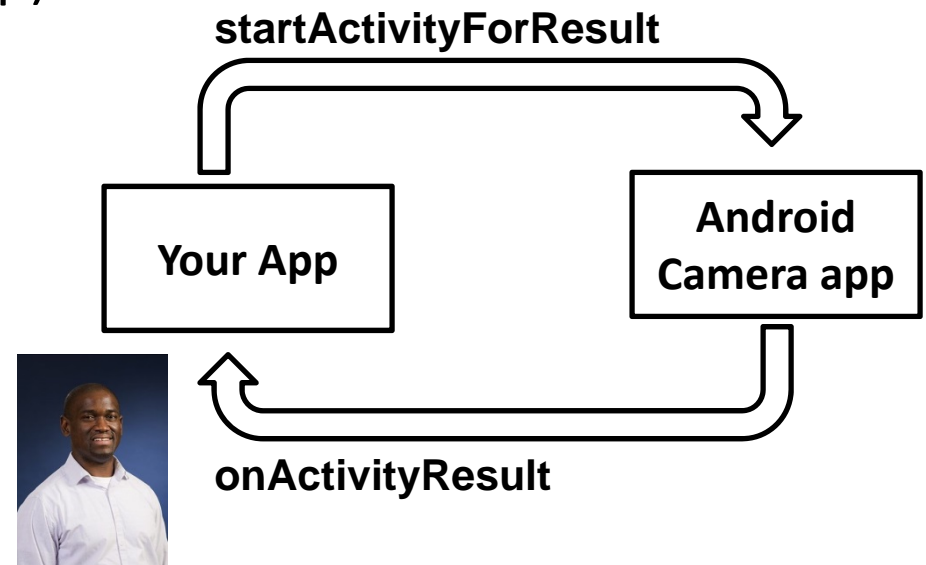
Send Intent requesting Android's Camera app take a picture

Check that there's at least 1 Activity that can handle request to take picture



Get the Thumbnail

- Android Camera app returns thumbnail of photo (small bitmap)
- Thumbnail returned in **Intent** delivered to **onActivityResult()**



```
protected void onActivityResult(int requestCode, int resultCode, Intent data) {
    if (requestCode == REQUEST_IMAGE_CAPTURE && resultCode == RESULT_OK) {
        Bundle extras = data.getExtras();
        Bitmap imageBitmap = (Bitmap) extras.get("data");
        mImageView.setImageBitmap(imageBitmap);
    }
}
```



Save Full-Sized Photo

- Android Camera app can save full-size photo to
 1. **Public external storage** (shared by all apps)
 - `getExternalStoragePublicDirectory()`
 - Need to get permission
 2. **Private storage** (Seen by only your app, deleted when your app uninstalls):
 - `getExternalFilesDir()`
- Either way, need phone owner's permission to write to external storage

```
<manifest ...>  
    <uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />  
    ...  
</manifest>
```



Taking Pictures: Bigger Example

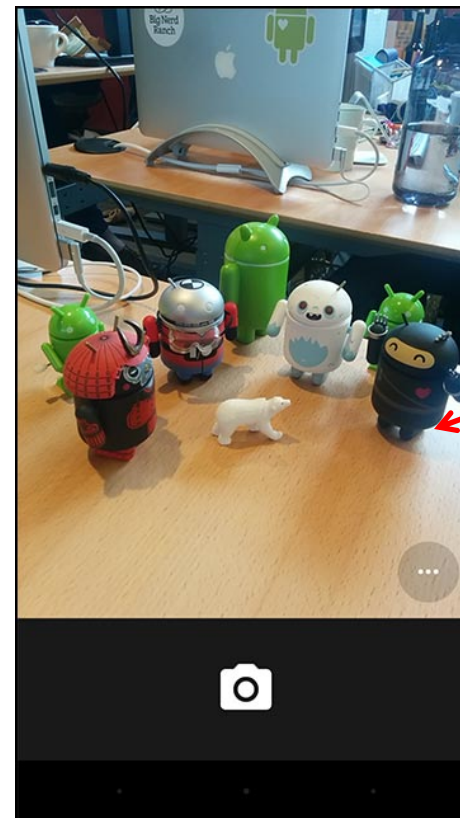
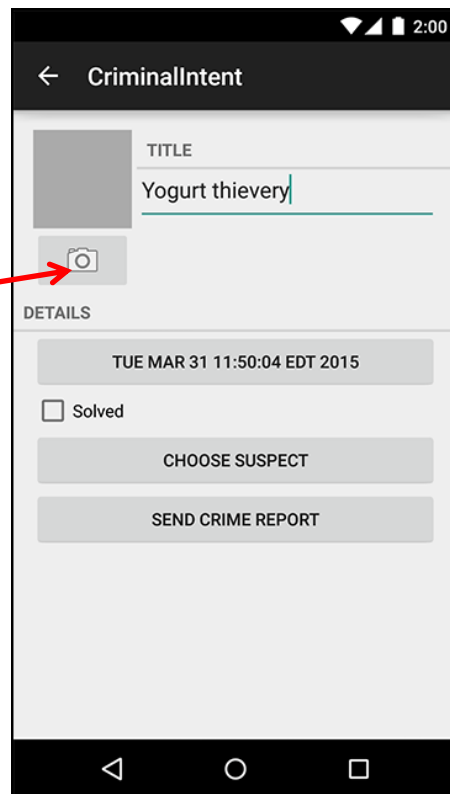


Taking Pictures with Intents

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

Click here
to take picture

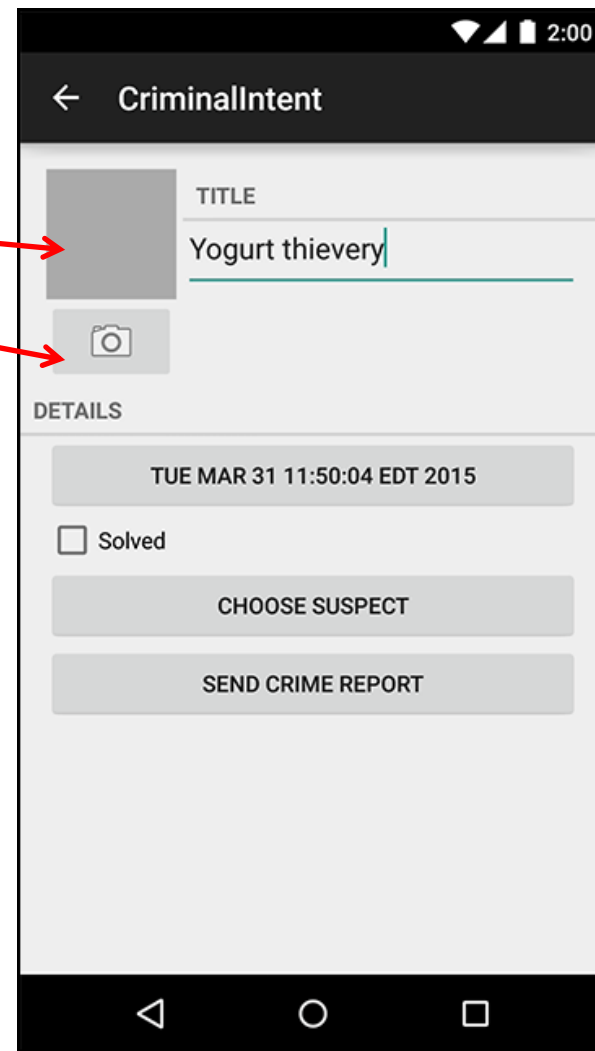


Launches
Camera app



Create Placeholder for Picture

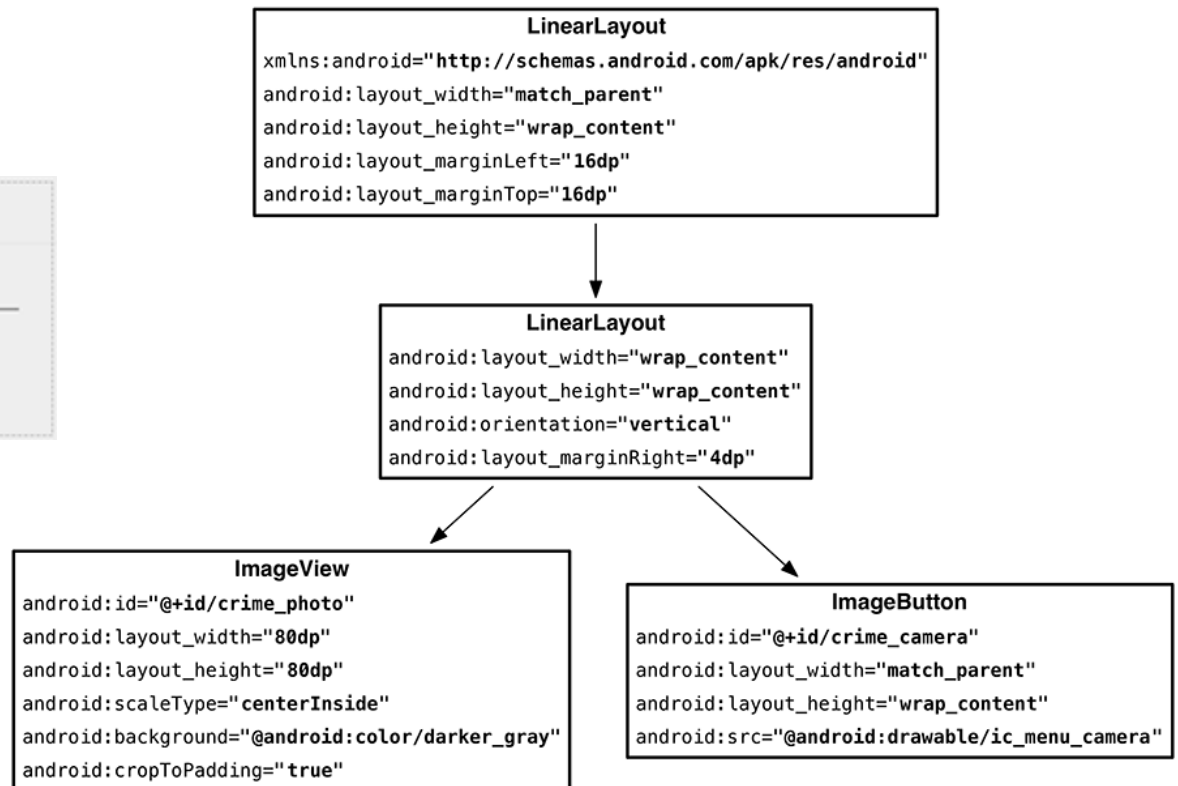
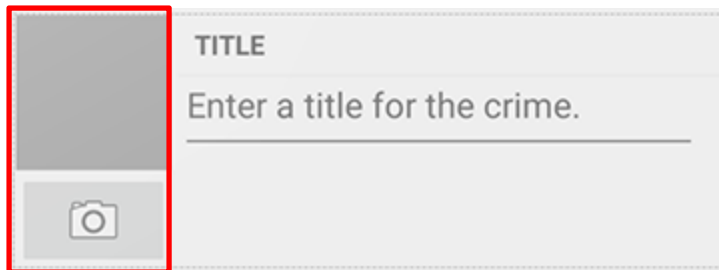
- Modify layout to include
 - ImageView for picture
 - Button to take picture





Create Camera and Title

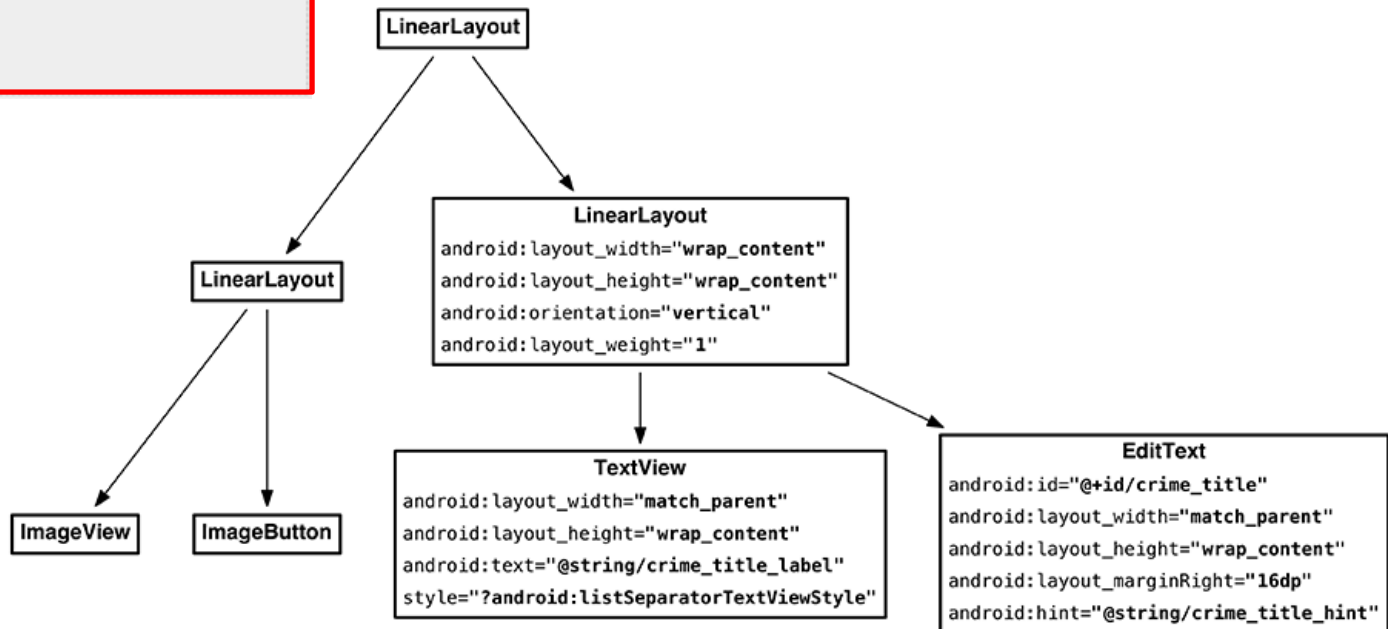
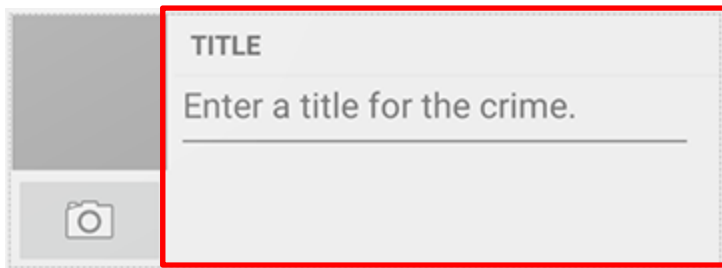
- Once created, we can include this in both landscape and portrait versions
- First, build out left side

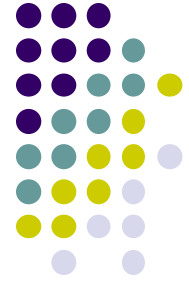




Create Camera and Title

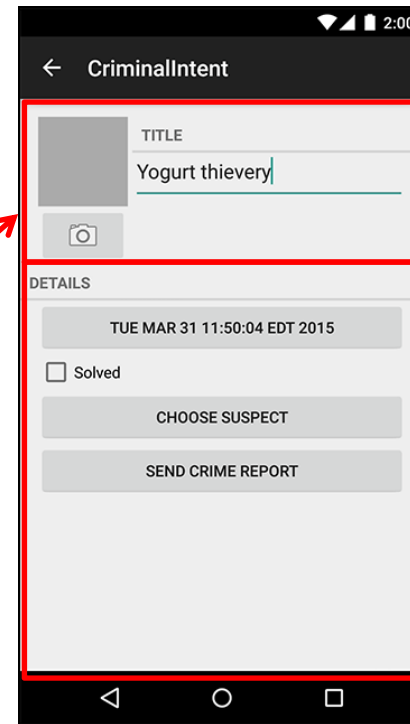
- Build out right side





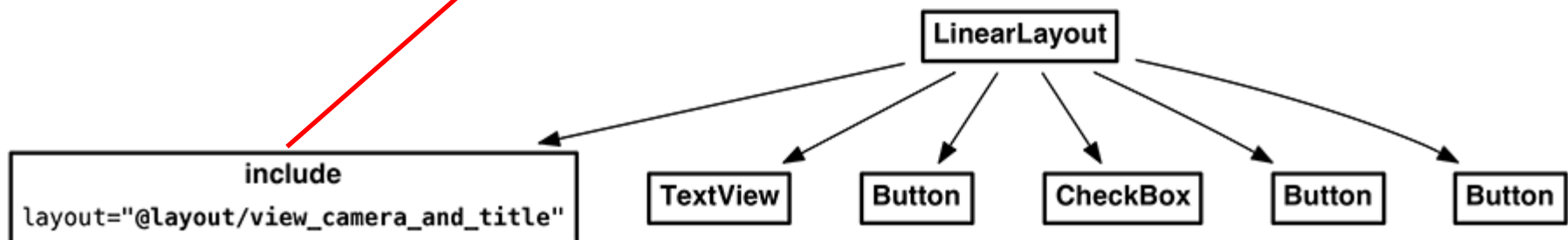
Include Camera and Title in Layout

- Include in previously created top part
- Create, add in bottom part



Camera and Title

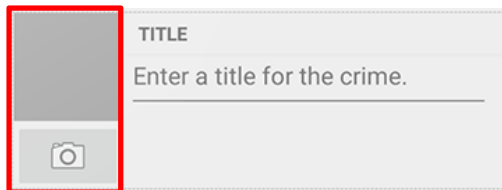
The rest of the layout





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;
}

...
```

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);
```

← Create new intent
for image capture

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

← Check with PackageManager
that Camera exists on this
phone

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

← Build Intent to capture
image, store at uri location

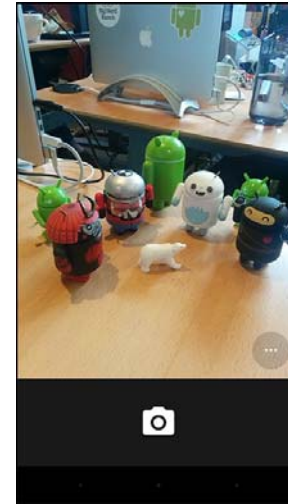
```
mPhotoButton.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        startActivityForResult(captureImage, REQUEST_PHOTO);
    }
});
```

← Take picture when
button is clicked

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

```
return v;
```

```
}
```





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

- Google Camera “Taking Photos Simply” Tutorials, <http://developer.android.com/training/camera/photobasics.html>
- 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