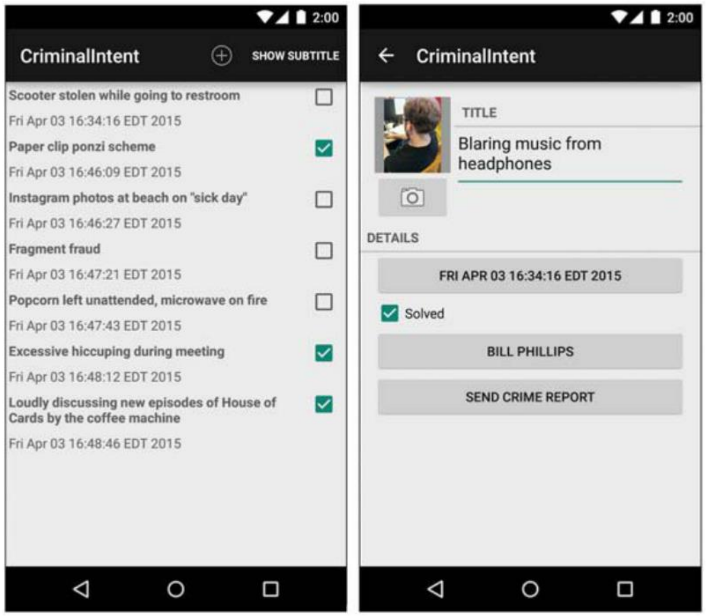
FRAGMENTS

Mengjun Xie

Why We Need Fragments

- UI flexibility: the ability to compose and recompose an activity's view at runtime depending on what the user or the device requires



2:00

CriminalIntent ⊕ SHOW SUBTITLE

Scooter stolen while going to restroom Fri Apr 03 16:34:16 EDT 2015	☐
Paper clip ponzi scheme Fri Apr 03 16:46:09 EDT 2015	✓
Instagram photos at beach on "sick day" Fri Apr 03 16:46:27 EDT 2015	☐
Fragment fraud Fri Apr 03 16:47:21 EDT 2015	☐
Popcorn left unattended, microwave on fire Fri Apr 03 16:47:43 EDT 2015	☐
Excessive hiccuping during meeting Fri Apr 03 16:48:12 EDT 2015	✓
Loudly discussing new episodes of House of Cards by the coffee machine Fri Apr 03 16:48:46 EDT 2015	✓

2:00

← CriminalIntent

TITLE

Blaring music from headphones

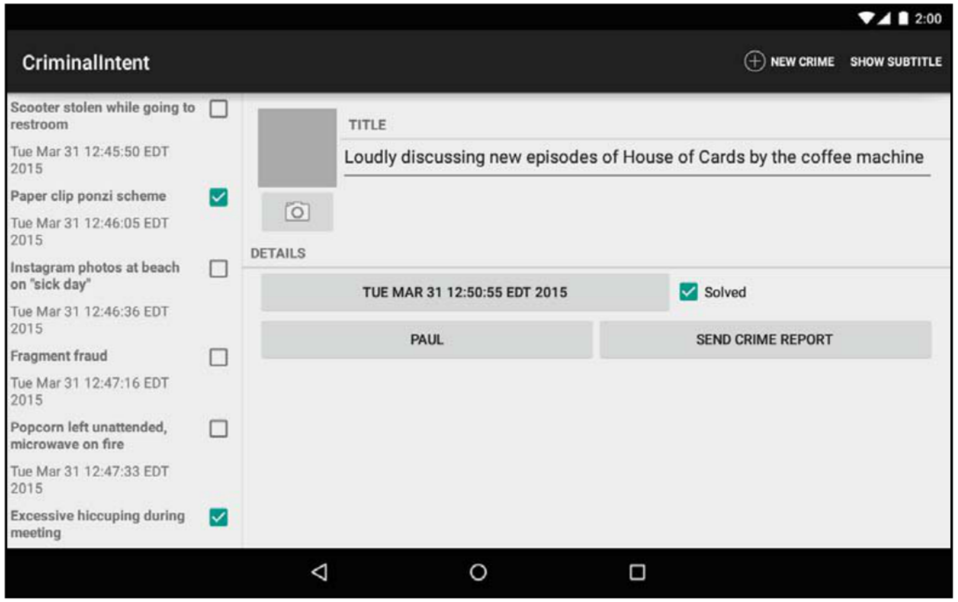
DETAILS

FRI APR 03 16:34:16 EDT 2015

✓ Solved

BILL PHILLIPS

SEND CRIME REPORT



2:00

CriminalIntent ⊕ NEW CRIME SHOW SUBTITLE

Scooter stolen while going to restroom Tue Mar 31 12:45:50 EDT 2015	☐
Paper clip ponzi scheme Tue Mar 31 12:46:05 EDT 2015	✓
Instagram photos at beach on "sick day" Tue Mar 31 12:46:36 EDT 2015	☐
Fragment fraud Tue Mar 31 12:47:16 EDT 2015	☐
Popcorn left unattended, microwave on fire Tue Mar 31 12:47:33 EDT 2015	☐
Excessive hiccuping during meeting	✓

TITLE

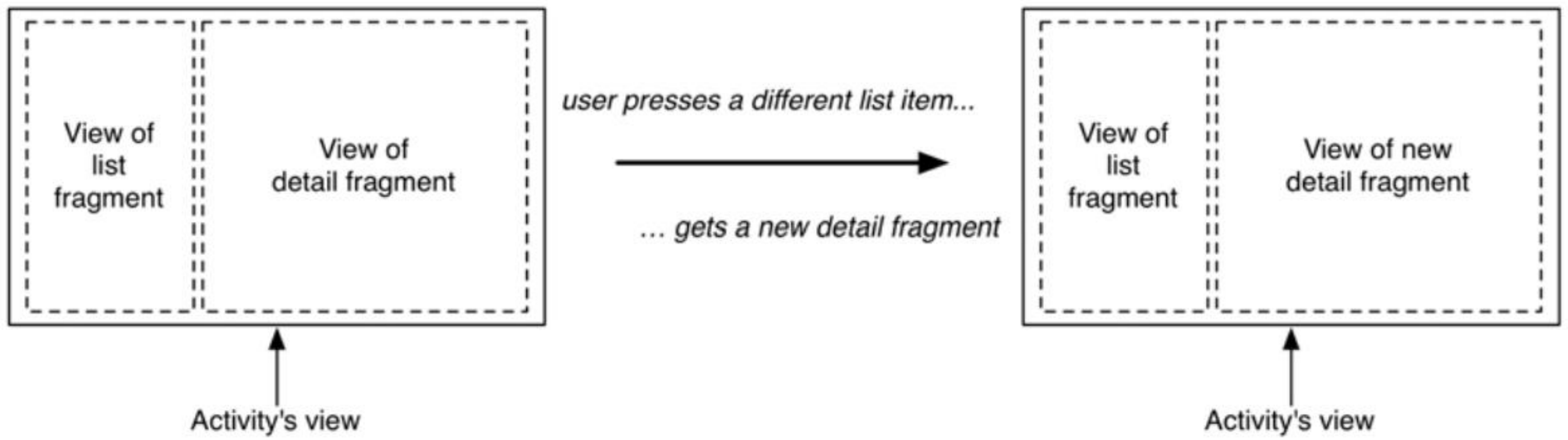
Loudly discussing new episodes of House of Cards by the coffee machine

DETAILS

TUE MAR 31 12:50:55 EDT 2015 ✓ Solved

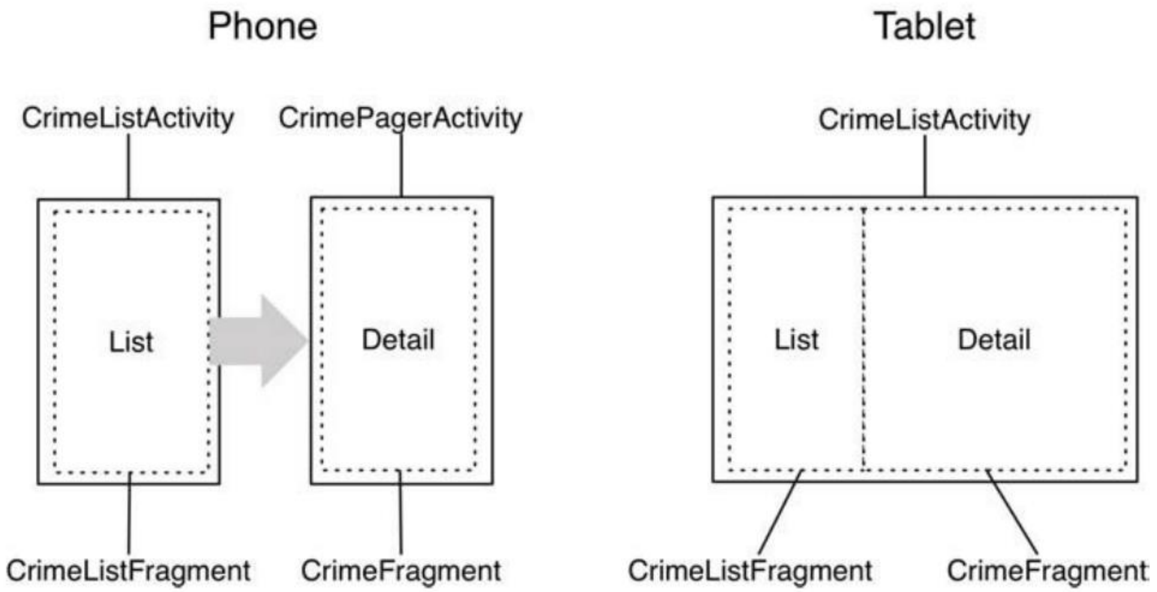
PAUL SEND CRIME REPORT

Example: List-detail App



- Using UI fragments separates the UI of your app into building blocks.
- Using UI fragments makes it easy to build list-detail or tab interfaces, tack on animated sidebars, and more.

Sample App: CriminalIntent



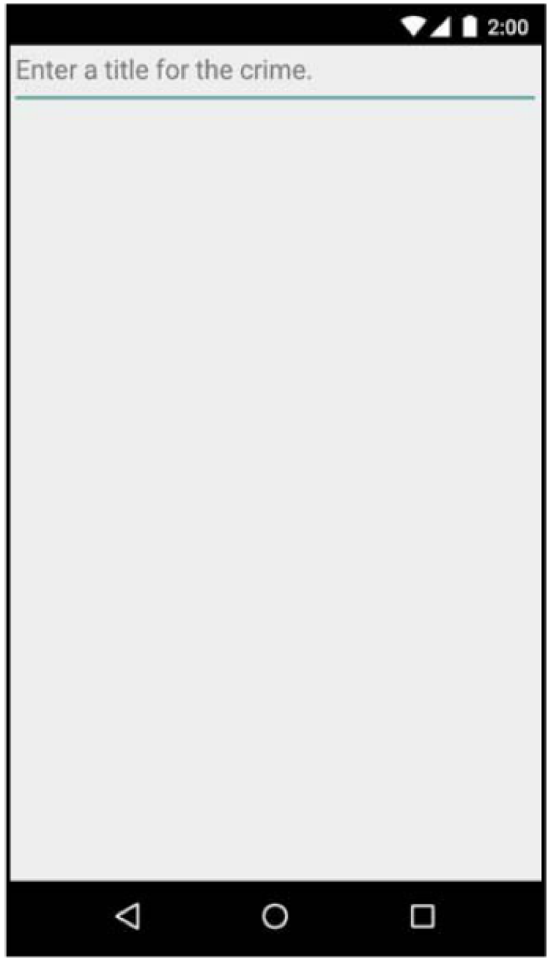
- Phone: 2 single-fragment activities
- Tablet: 1 activity with 2 fragments

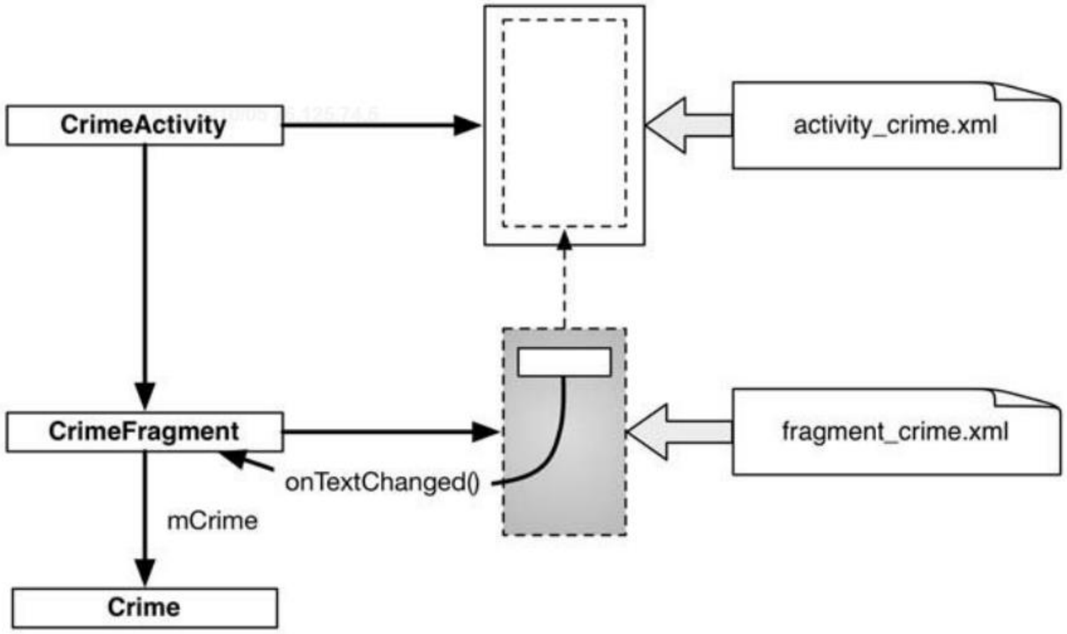
Overview of This Class

- Single-fragment Activities
- Lists with RecyclerView
- 2-fragment Activity with list-detail UI
- Toolbar

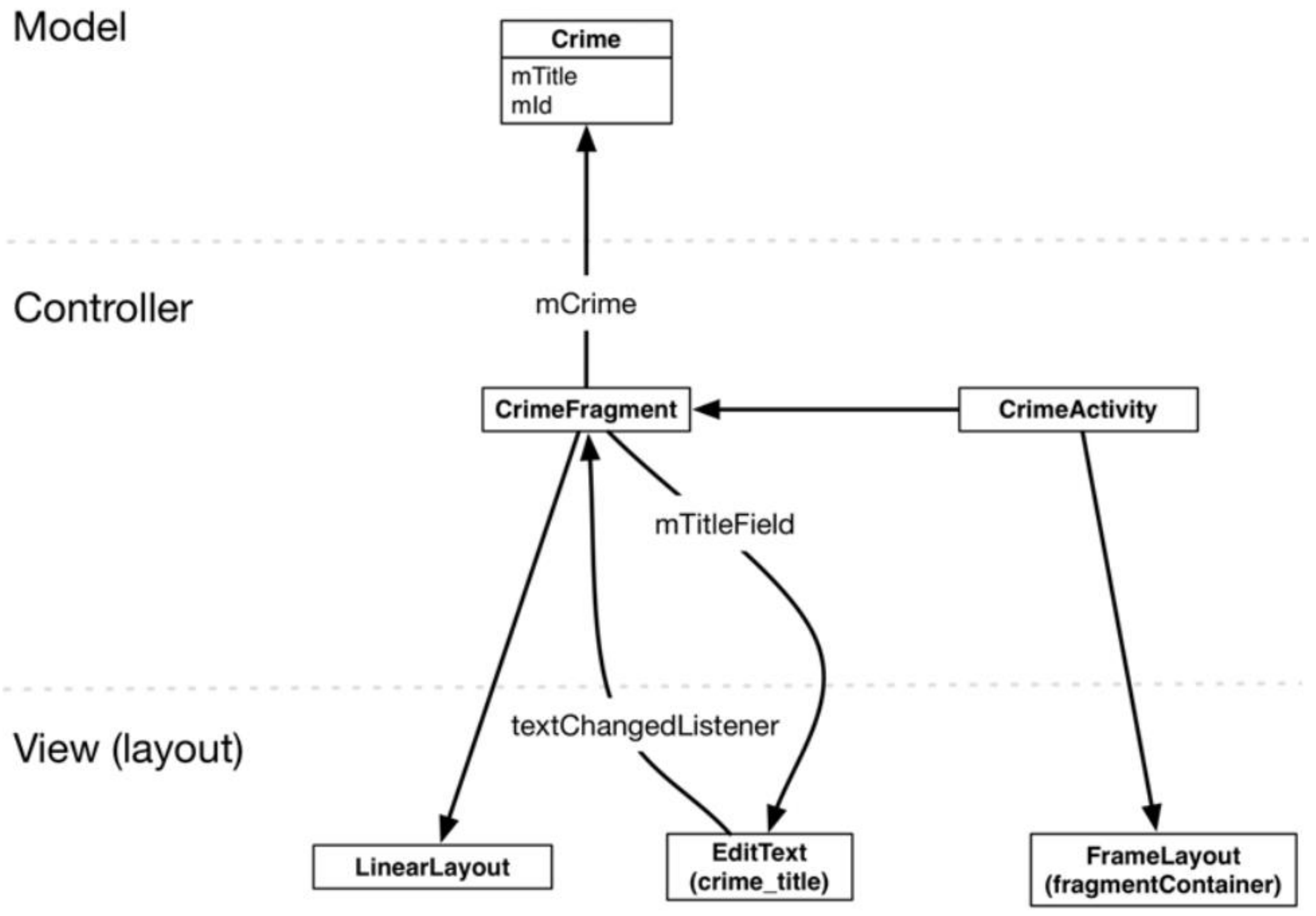
Single-fragment Activities

A Very Simple Start





Model-View-Controller (MVC)

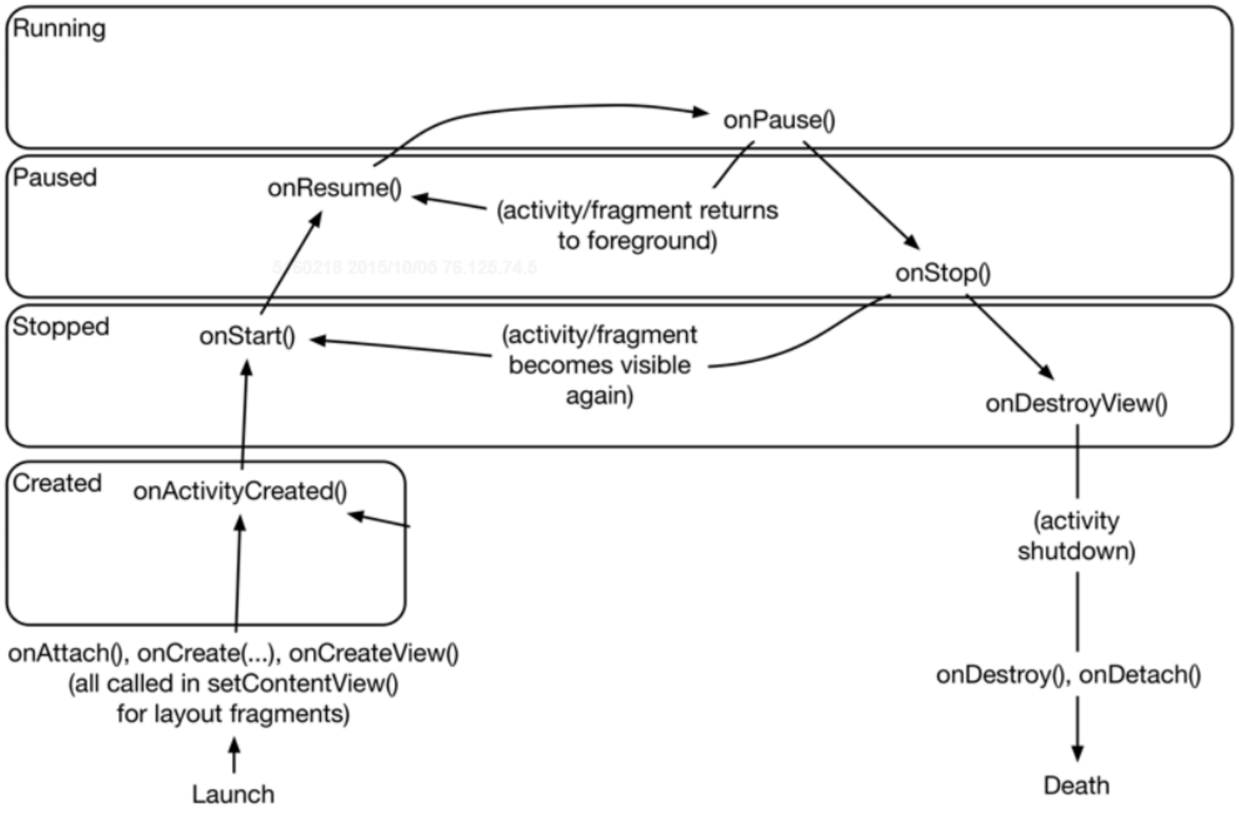


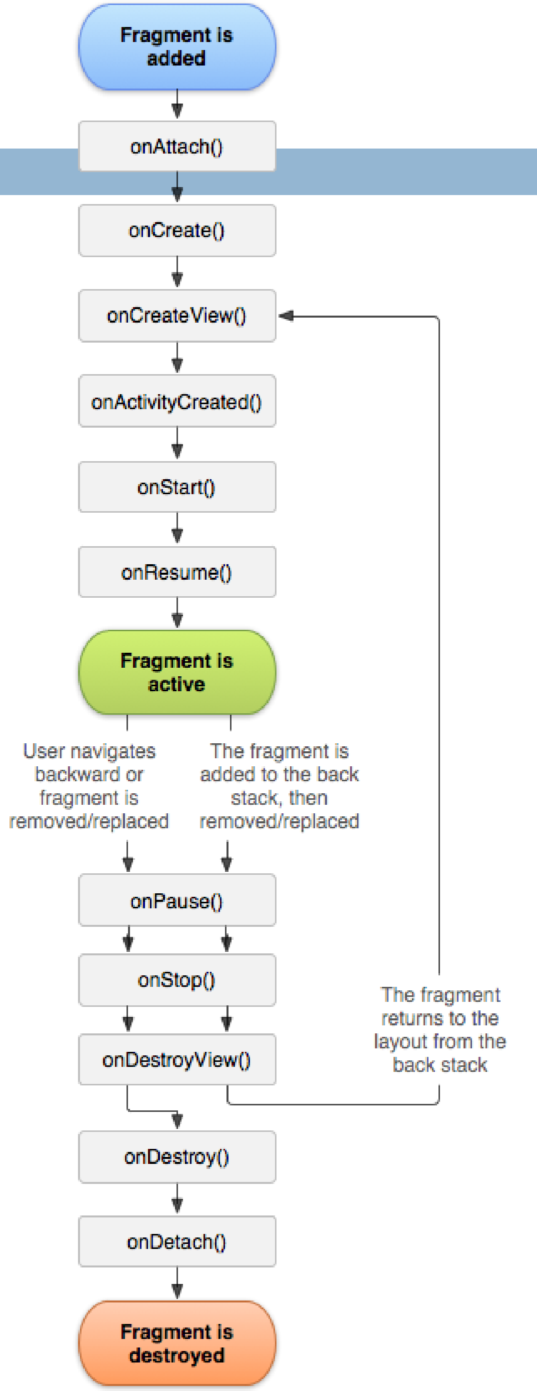
- A model object holds the app's data and "business logic," knowing nothing about UI.
- View objects know how to draw and how to respond to user input.
- Controller objects tie the view and model objects together. They contain "application logic."

Hosting a UI Fragment

- To host a UI fragment, an activity must:
  - Define a spot in its layout for the fragment’s view
    - Use **FrameLayout** in Activity’s layout xml as a container layout
    - Compose fragment UI in the same way as activity UI
    - Wire up the widgets inflated from the layout in code
  - Manage the lifecycle of the fragment instance
    - Fragments are the activity’s internal business.
    - Fragment lifecycle methods are called by the hosting activity, not the OS.

Fragment Lifecycle


Running
onPause()
Paused
onResume()
(activity/fragment returns to foreground)
onStop()
Stopped
onStart()
(activity/fragment becomes visible again)
onDestroyView()
Created
onActivityCreated()
(activity shutdown)
onAttach(), onCreate(...), onCreateView()
(all called in setContentView() for layout fragments)
onDestroy(), onDetach()
Launch
Death



CrimeFragment Class

```
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;
    }

}
```

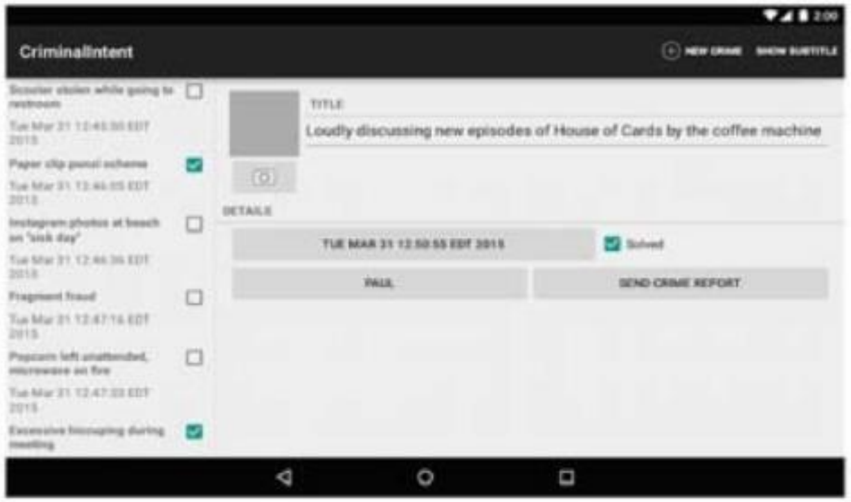
- Use **android.support.v4.app.Fragment** for better backward compatibility.
- Inflate fragment’s view and return the inflated View to the hosting activity in **onCreateView()**

CrimeActivity Class

```
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();
        }
    }
}
```

- Call **getSupportFragmentManager()** if using **FragmentActivity** in the support library

Less is More



CriminalIntent

NEW CRIME SHOW SUBTITLE

Scooter stolen while going to restroom ☐
Tue Mar 31 12:45:50 EDT 2015

Paper clip ponzi scheme ✓
Tue Mar 31 12:46:05 EDT 2015

Instagram photos at beach on "sick day" ☐
Tue Mar 31 12:46:36 EDT 2015

Fragment fraud ☐
Tue Mar 31 12:47:16 EDT 2015

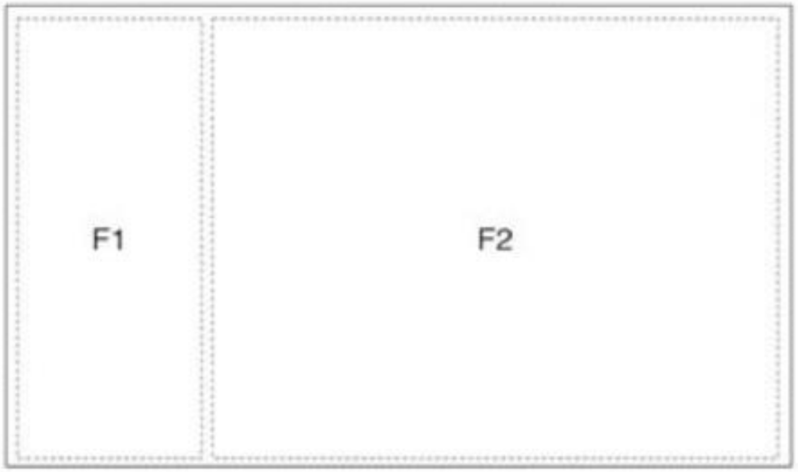
Popcorn left unattended, microwave on fire ☐
Tue Mar 31 12:47:33 EDT 2015

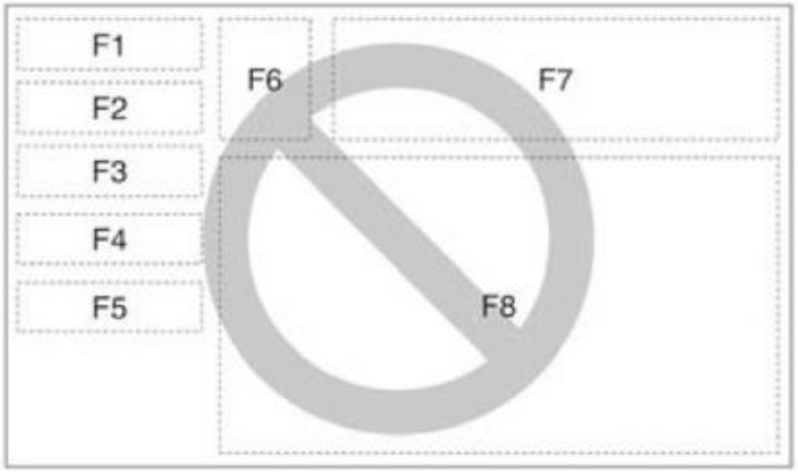
Excessive hiccuping during meeting ✓

TITLE
Loudly discussing new episodes of House of Cards by the coffee machine

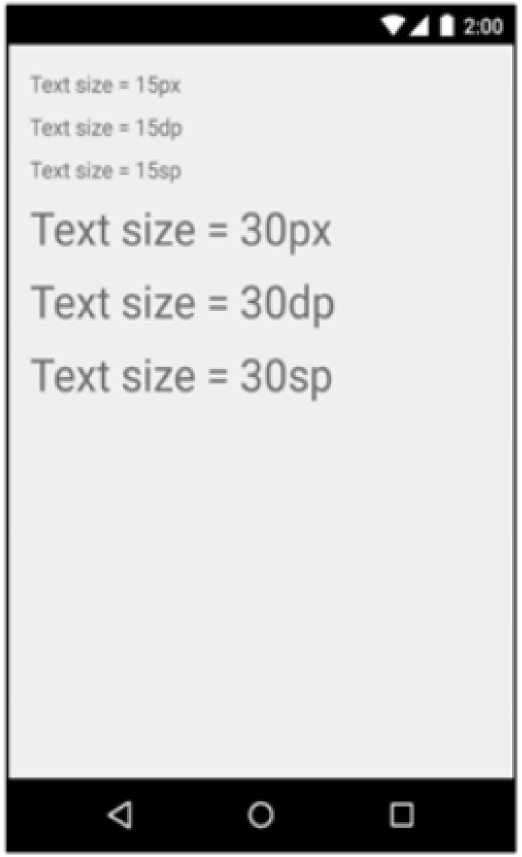
DETAILS
TUE MAR 31 12:50:55 EDT 2015 ✓ Solved

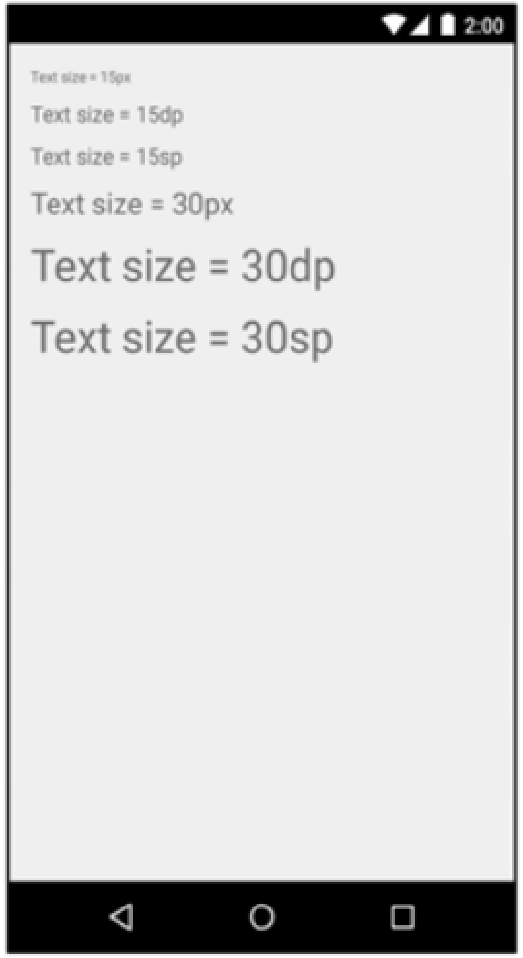
PAUL SEND CRIME REPORT

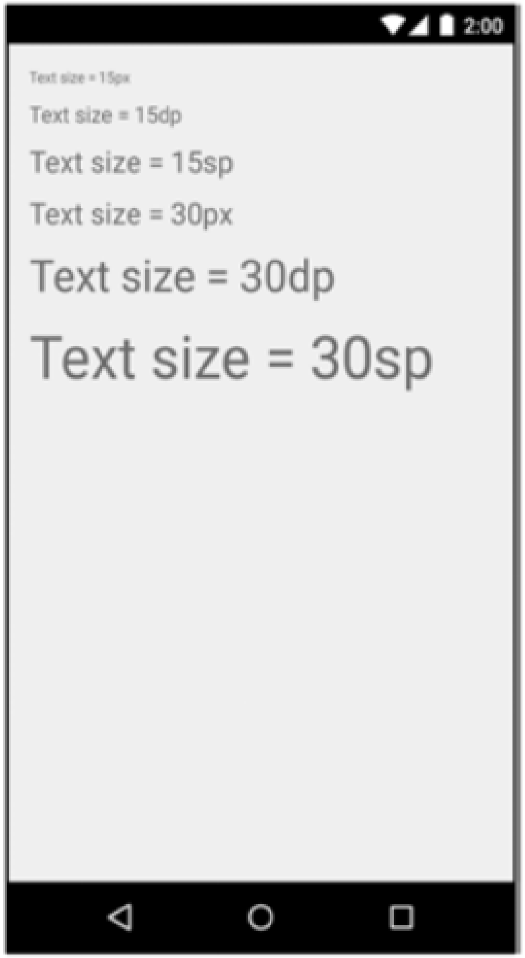




Dimension Units in Action


Text size = 15px
Text size = 15dp
Text size = 15sp
Text size = 30px
Text size = 30dp
Text size = 30sp


Text size = 15px
Text size = 15dp
Text size = 15sp
Text size = 30px
Text size = 30dp
Text size = 30sp


Text size = 15px
Text size = 15dp
Text size = 15sp
Text size = 30px
Text size = 30dp
Text size = 30sp

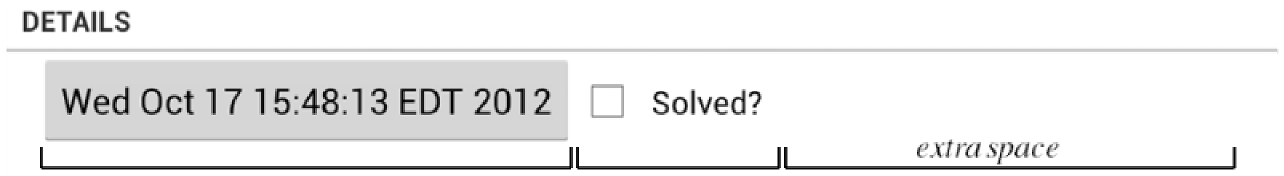
MDPI HDPI HDPI with large text

Dimension Units

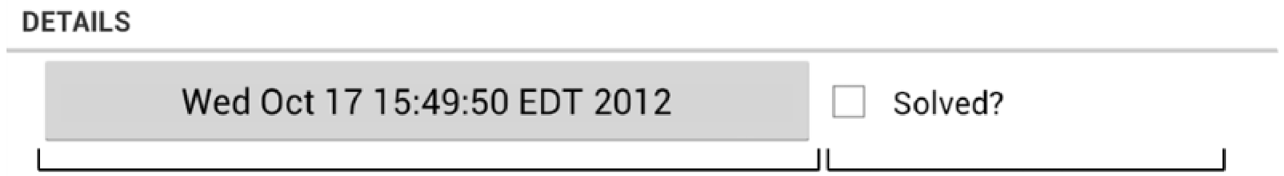
- **dp**: density-independent pixel. Typically use this for margins, padding, or anything else with a pixel value.
  - For displays with a higher density, dp expands to fill a larger number of screen pixels.
  - 1 dp is always 1/160 of an inch on screen.
- **sp**: scale-independent pixel. They are density-independent pixels that also take into account the user's font size preference.
  - Always use sp to set display text size.
- **pt, mm, in**: scaled units like dp for sizes in points (1/72 of an inch), millimeters, or inches.
  - However, not recommend using them: not all devices are correctly configured for these units to scale correctly.

android:layout_weight

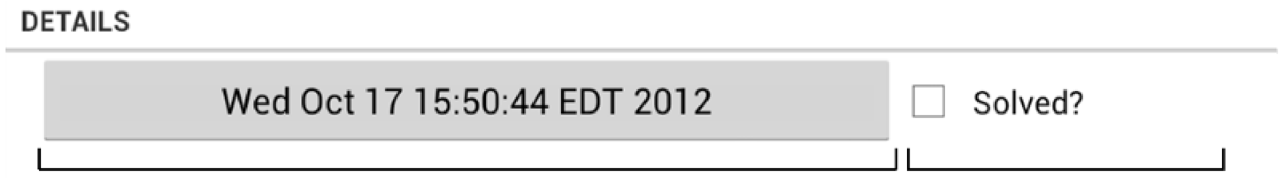
Layout_weight assigns an “importance” value to a view in terms of how much space it should occupy on the screen



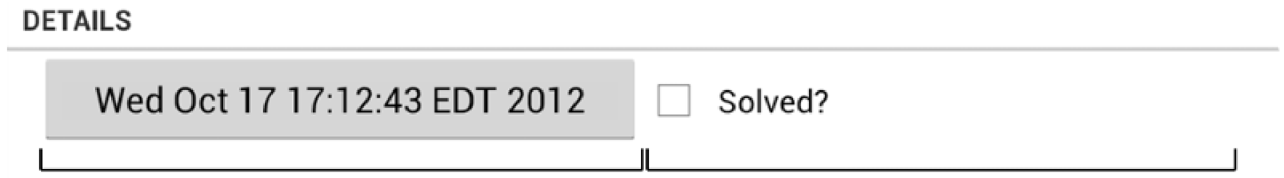
- No layout_weight



- 1:1

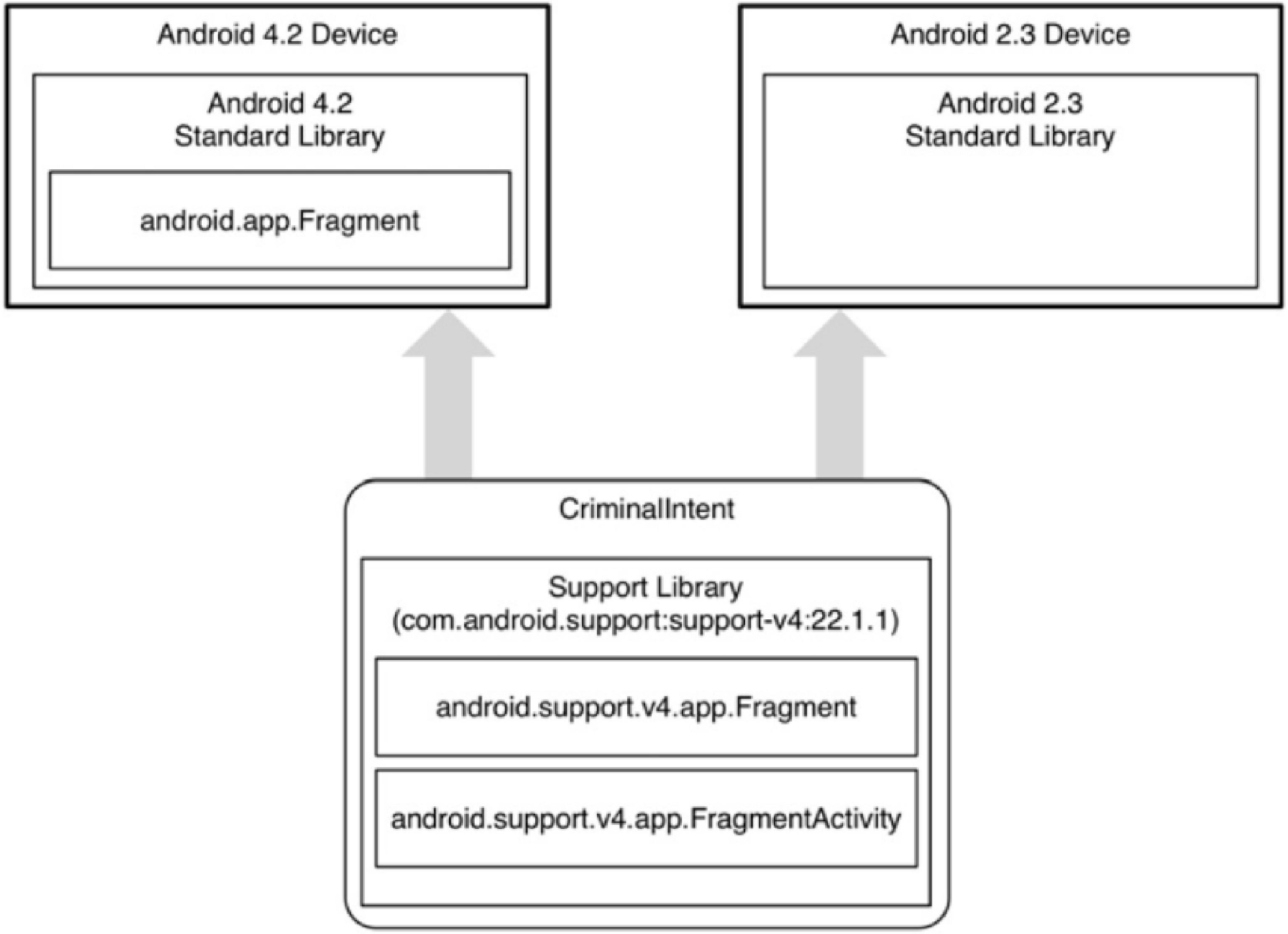


- 2:1



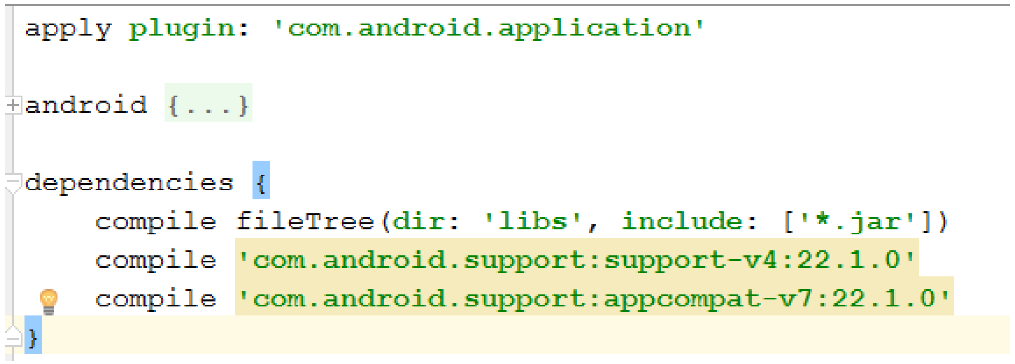
- 1:1 and layout_width= “0dp”

Two Fragment Classes


Android 4.2 Device
Android 4.2 Standard Library
android.app.Fragment
Android 2.3 Device
Android 2.3 Standard Library
CriminalIntent
Support Library (com.android.support:support-v4:22.1.1)
android.support.v4.app.Fragment
android.support.v4.app.FragmentActivity

Gradle Dependencies

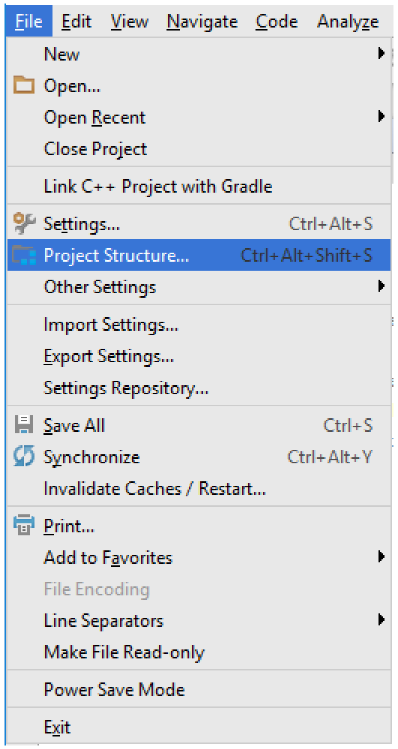
- Dependencies need to be added in app/build.gradle if Fragment class in the support library is used.

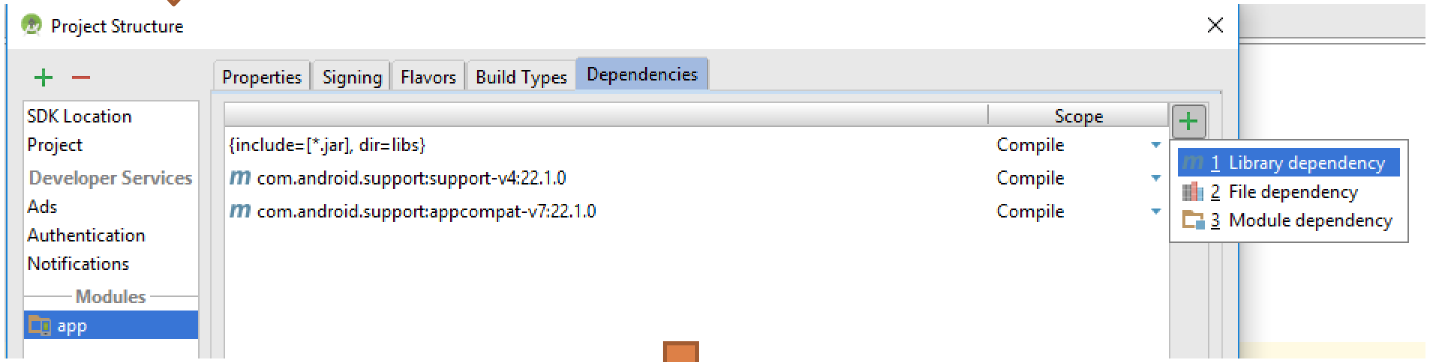


```
apply plugin: 'com.android.application'

android {...}

dependencies {
    compile fileTree(dir: 'libs', include: ['*.jar'])
    compile 'com.android.support:support-v4:22.1.0'
    compile 'com.android.support:appcompat-v7:22.1.0'
}
```

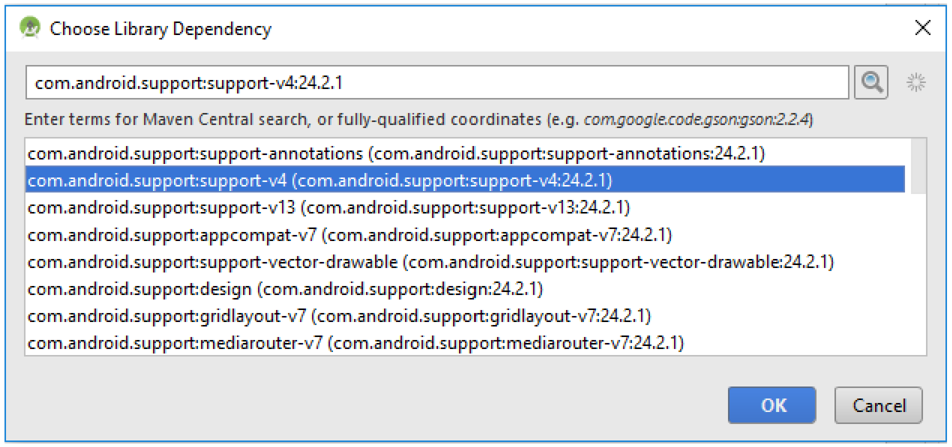
- The dependencies use Maven format: **groupId:artifactId:version.**
  - Maven is a dependency management tool.
  - <u>groupId</u>: unique ID for a set of libraries available in the Maven repository.
    - The library's base package name is often used as the groupId: com.android.support
  - <u>artifactId</u>: name of a specific library within the package: support-v4
    - com.android.support also contains other libraries such as support-v13 and appcompat-v7
    - Google uses the naming convention basename-vX for their support libraries, where -vX represents the minimum API level the library supports.
  - <u>version</u>: revision number of the library

Dependency Management in Studio





	Scope
{include=[*.jar], dir=libs}	Compile
com.android.support:support-v4:22.1.0	Compile
com.android.support:appcompat-v7:22.1.0	Compile



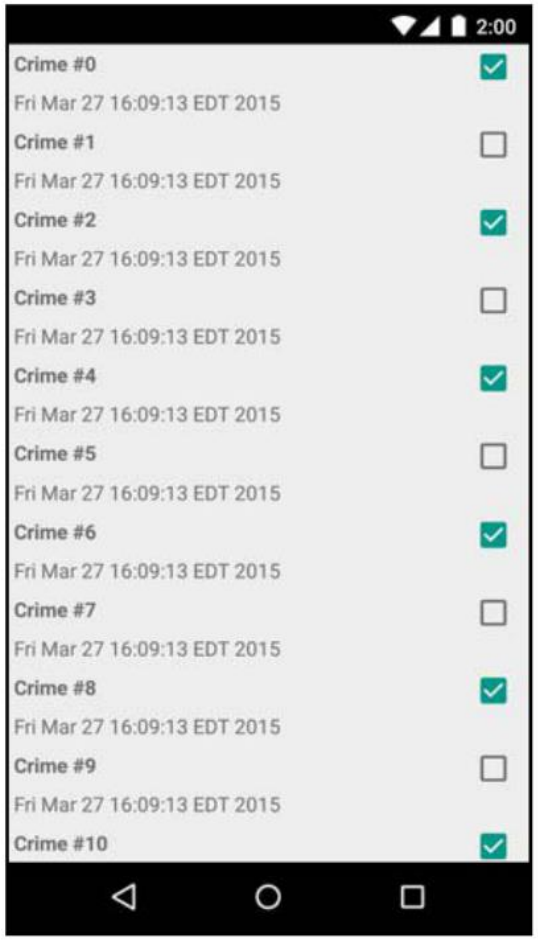
com.android.support:support-v4:24.2.1

Enter terms for Maven Central search, or fully-qualified coordinates (e.g. *com.google.code.gson:gson:2.2.4*)

```
com.android.support:support-annotations (com.android.support:support-annotations:24.2.1)
com.android.support:support-v4 (com.android.support:support-v4:24.2.1)
com.android.support:support-v13 (com.android.support:support-v13:24.2.1)
com.android.support:appcompat-v7 (com.android.support:appcompat-v7:24.2.1)
com.android.support:support-vector-drawable (com.android.support:support-vector-drawable:24.2.1)
com.android.support:design (com.android.support:design:24.2.1)
com.android.support:gridlayout-v7 (com.android.support:gridlayout-v7:24.2.1)
com.android.support:mediarouter-v7 (com.android.support:mediarouter-v7:24.2.1)
```

Displaying Lists with RecyclerView

Design with a List



2:00

Crime #0 ☑
Fri Mar 27 16:09:13 EDT 2015

Crime #1 ☐
Fri Mar 27 16:09:13 EDT 2015

Crime #2 ☑
Fri Mar 27 16:09:13 EDT 2015

Crime #3 ☐
Fri Mar 27 16:09:13 EDT 2015

Crime #4 ☑
Fri Mar 27 16:09:13 EDT 2015

Crime #5 ☐
Fri Mar 27 16:09:13 EDT 2015

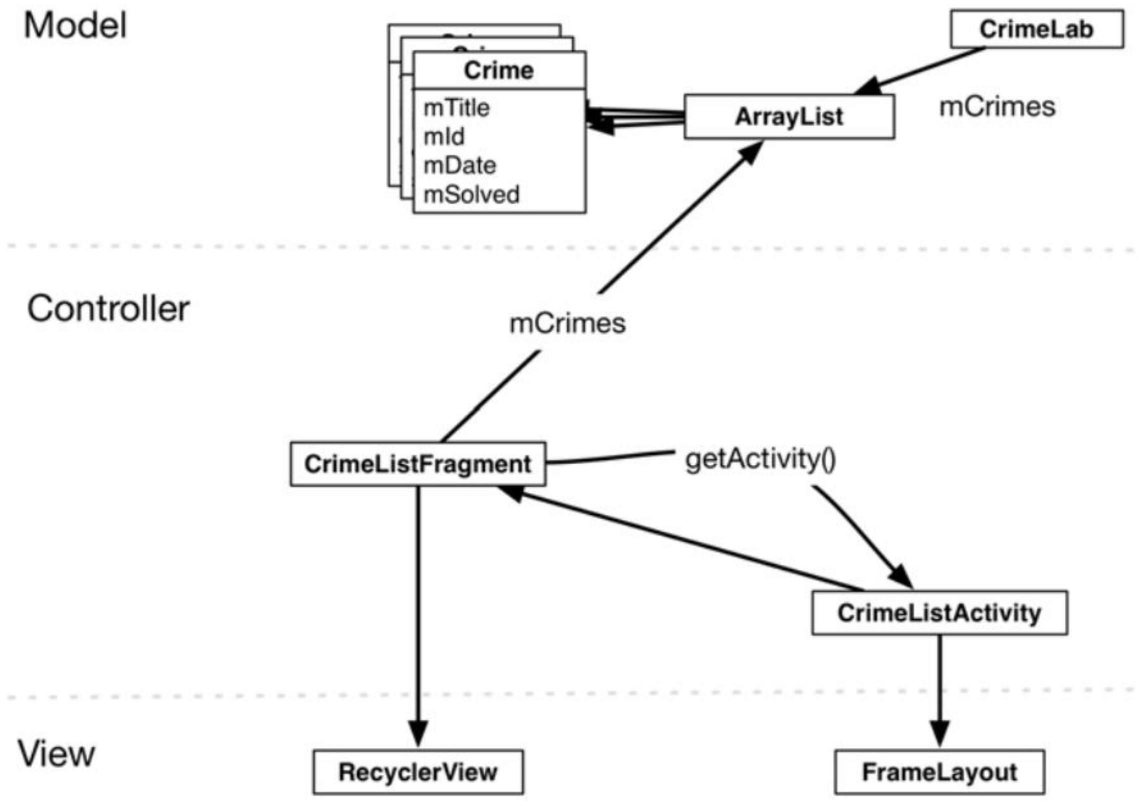
Crime #6 ☑
Fri Mar 27 16:09:13 EDT 2015

Crime #7 ☐
Fri Mar 27 16:09:13 EDT 2015

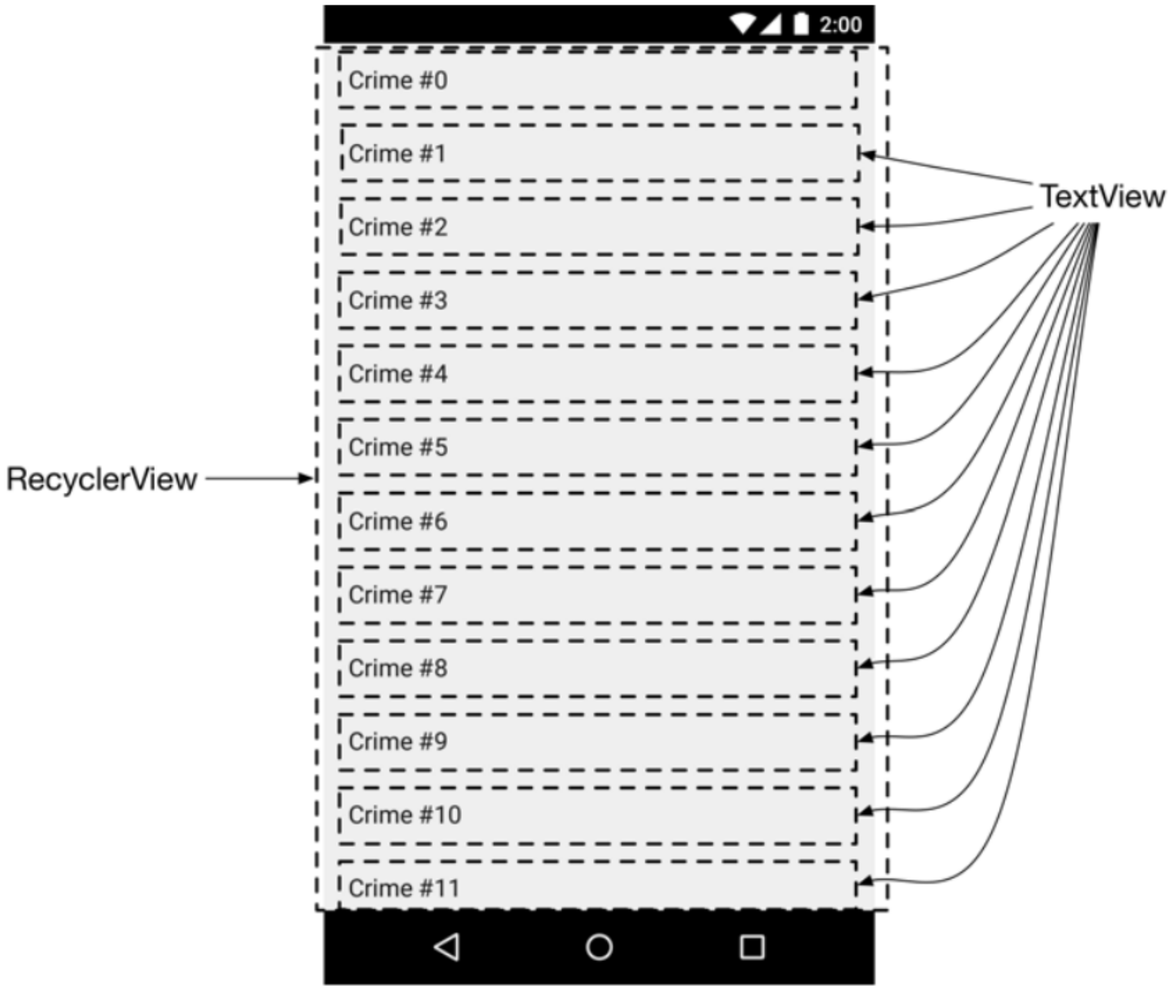
Crime #8 ☑
Fri Mar 27 16:09:13 EDT 2015

Crime #9 ☐
Fri Mar 27 16:09:13 EDT 2015

Crime #10 ☑

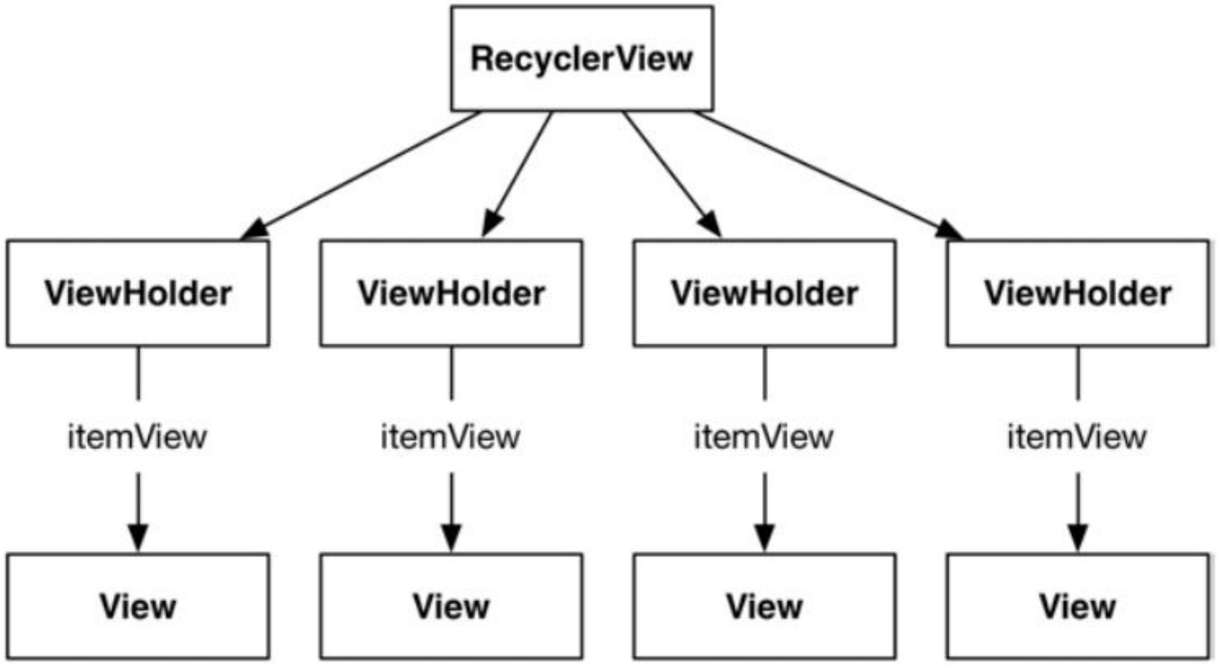


RecyclerView



- **RecyclerView** recycles TextViews and positions them on the screen.
  - To get the TextViews in the first place, it works with two classes: an **Adapter** subclass and a **ViewHolder** subclass

ViewHolder

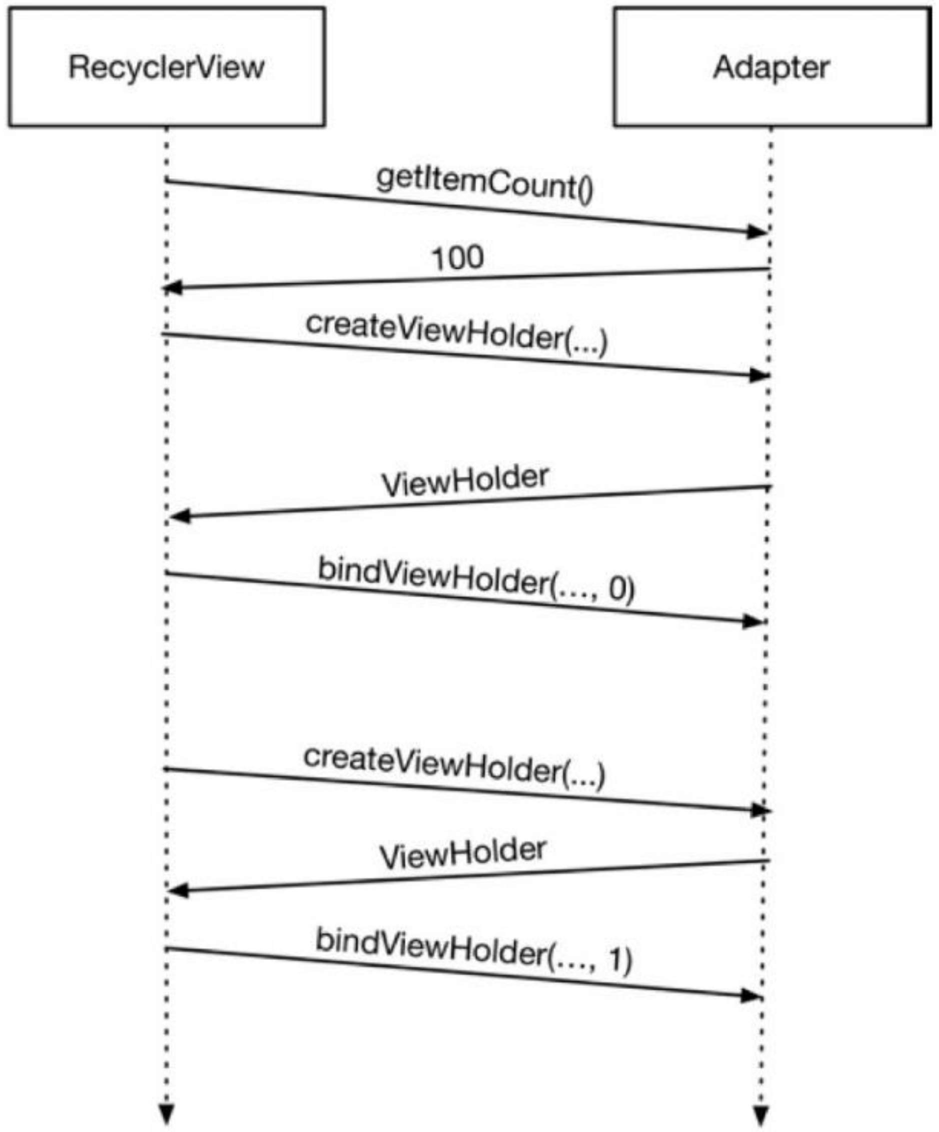


```
public class ListRow extends RecyclerView.ViewHolder {
    public ImageView mThumbnail;

    public ListRow(View view) {
        super(view);

        mThumbnail = (ImageView) view.findViewById(R.id.thumbnail);
    }
}
```

Adapter

- An adapter is a controller object that sits between RecyclerView and the data set the RecyclerView should display.
- The adapter is responsible for creating the necessary ViewHolders and binding ViewHolders to data from the model layer.
- When RecyclerView needs a view object, it will have a conversation with its adapter.

RecyclerView-Adapter Conversation



Using A RecyclerView

- In fragment_crime_list.xml

```
<android.support.v7.widget.RecyclerView
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/crime_recycler_view"
    android:layout_width="match_parent"
    android:layout_height="match_parent"/>
```

- In CrimeListFragment.java

```
public class CrimeListFragment extends Fragment {

    private RecyclerView mCrimeRecyclerView;

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

        mCrimeRecyclerView = (RecyclerView) view
                .findViewById(R.id.crime_recycler_view);
        mCrimeRecyclerView.setLayoutManager(new LinearLayoutManager(getActivity()));

        return view;
    }

}
```

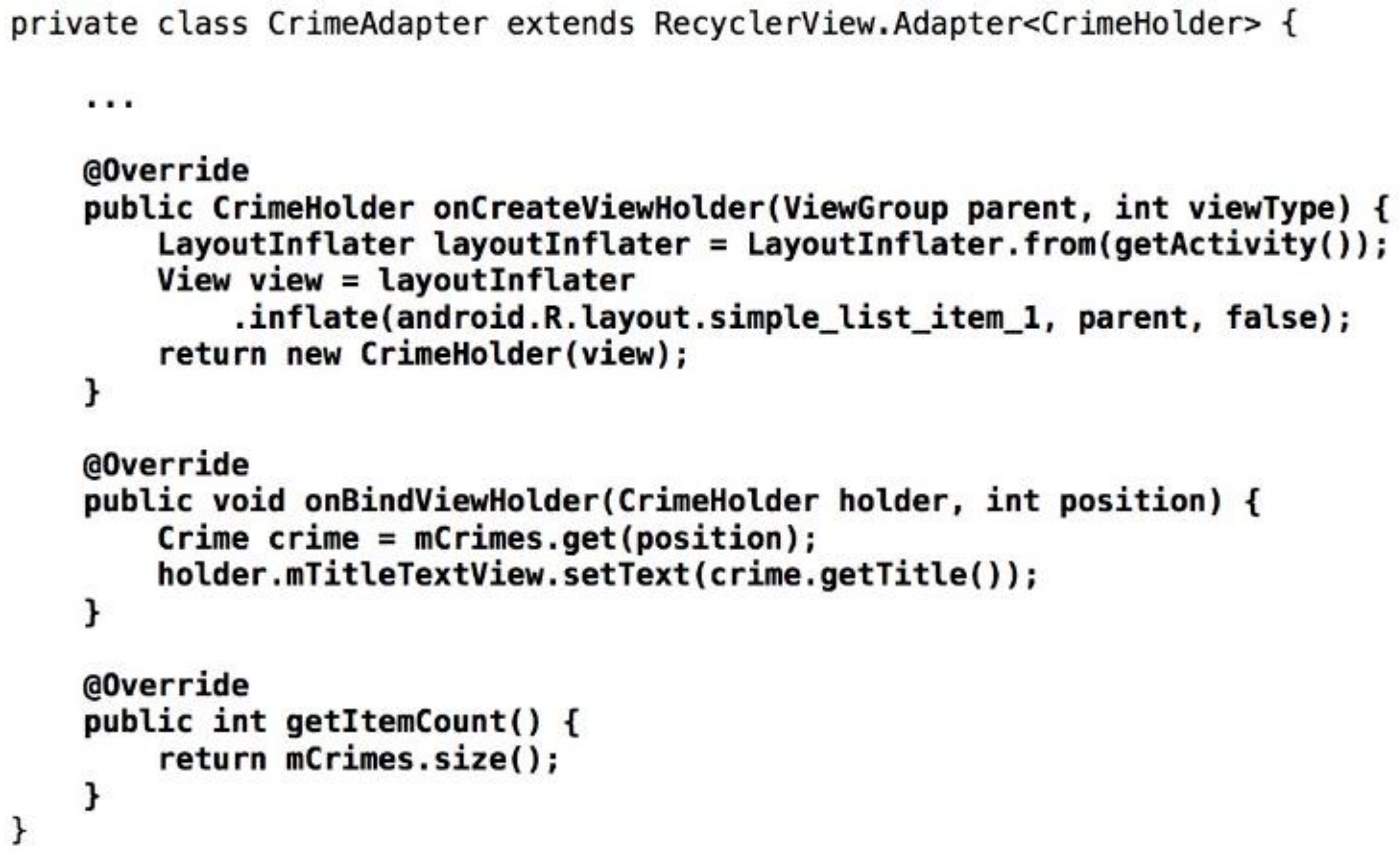
RecyclerView requires a **LayoutManager** to work

Implementing ViewHolder & Adapter

- In CrimeListFragment class

```
private class CrimeHolder extends RecyclerView.ViewHolder {

    public TextView mTitleTextView;

    public CrimeHolder(View itemView) {
        super(itemView);

        mTitleTextView = (TextView) itemView;
    }
}


private class CrimeAdapter extends RecyclerView.Adapter<CrimeHolder> {

    private List<Crime> mCrimes;

    public CrimeAdapter(List<Crime> crimes) {
        mCrimes = crimes;
    }
}
```

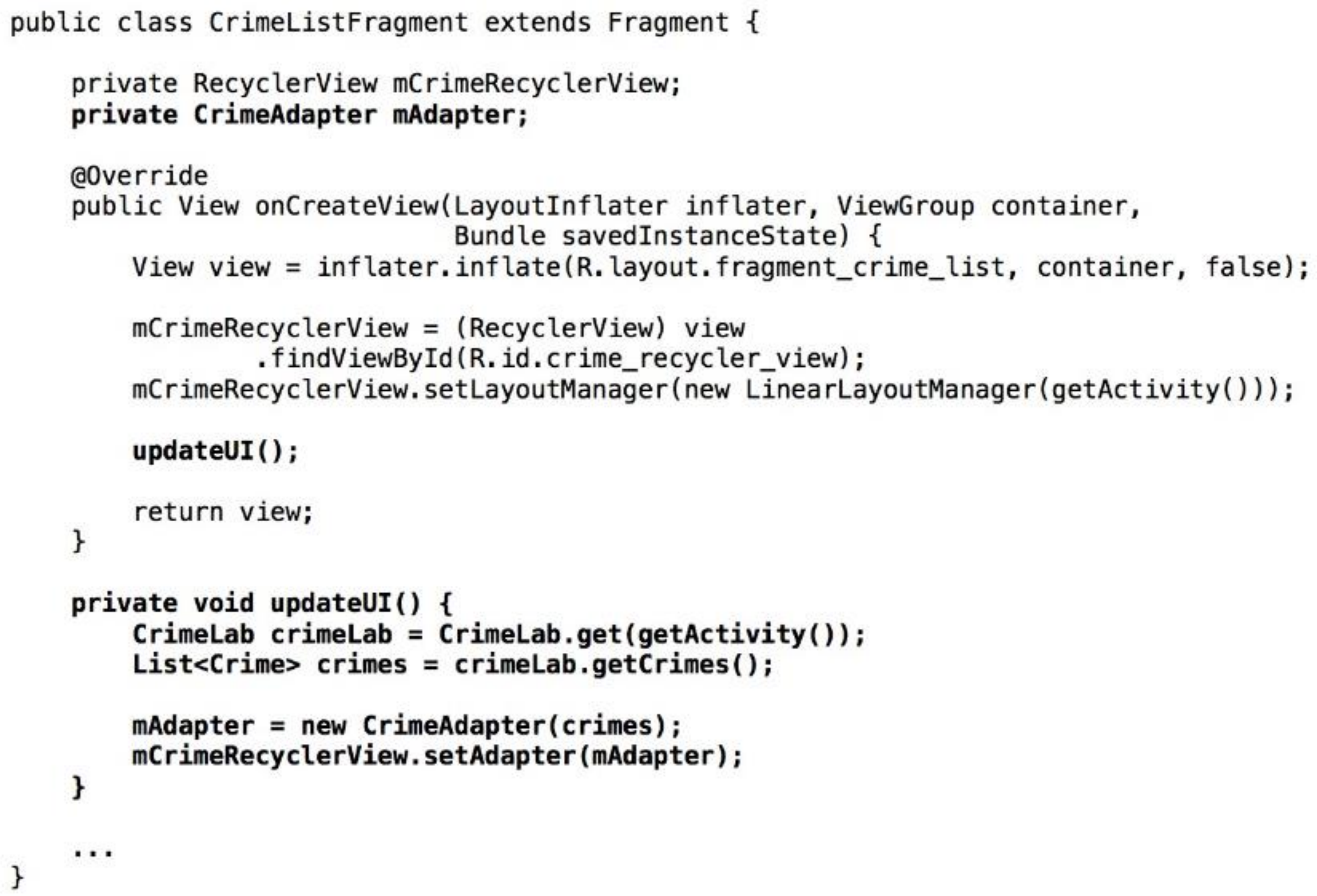
Implementing Adapter Callbacks



```
private class CrimeAdapter extends RecyclerView.Adapter<CrimeHolder> {

    ...

    @Override
    public CrimeHolder onCreateViewHolder(ViewGroup parent, int viewType) {
        LayoutInflater layoutInflater = LayoutInflater.from(getActivity());
        View view = layoutInflater
            .inflate(android.R.layout.simple_list_item_1, parent, false);
        return new CrimeHolder(view);
    }

    @Override
    public void onBindViewHolder(CrimeHolder holder, int position) {
        Crime crime = mCrimes.get(position);
        holder.mTitleTextView.setText(crime.getTitle());
    }

    @Override
    public int getItemCount() {
        return mCrimes.size();
    }
}
```

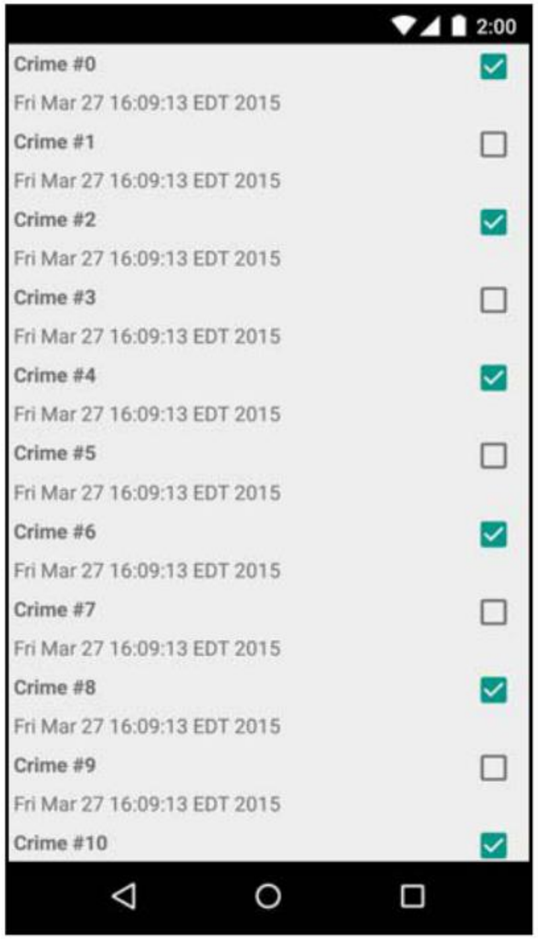
- onCreateViewHolder() is called by the RecyclerView when it needs a new View to display an item.
- onBindViewHolder() binds a ViewHolder’s View to your model object.

Setting Adapter



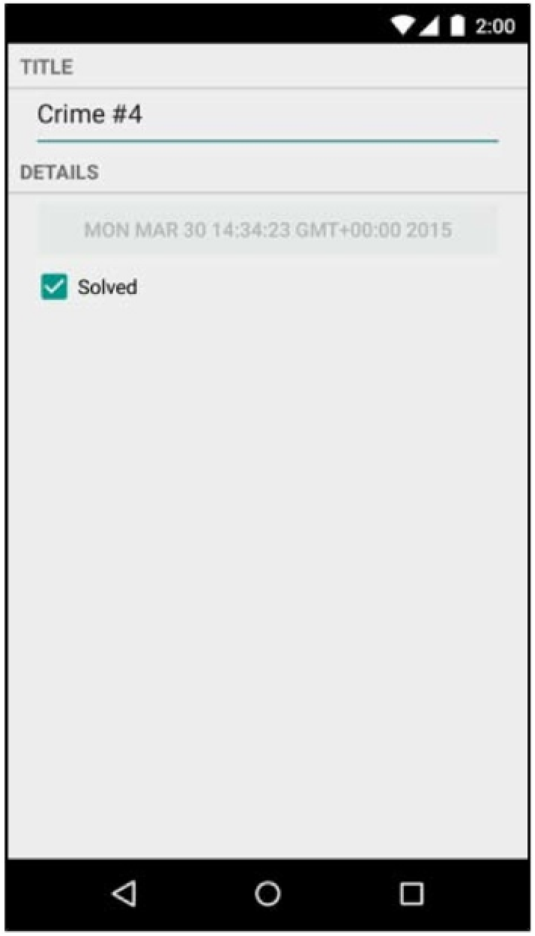
```
public class CrimeListFragment extends Fragment {

    private RecyclerView mCrimeRecyclerView;
    private CrimeAdapter mAdapter;

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

        mCrimeRecyclerView = (RecyclerView) view
                .findViewById(R.id.crime_recycler_view);
        mCrimeRecyclerView.setLayoutManager(new LinearLayoutManager(getActivity()));

        updateUI();

        return view;
    }

    private void updateUI() {
        CrimeLab crimeLab = CrimeLab.get(getActivity());
        List<Crime> crimes = crimeLab.getCrimes();

        mAdapter = new CrimeAdapter(crimes);
        mCrimeRecyclerView.setAdapter(mAdapter);
    }

    ...
}
```

Starting an Activity from a Fragment



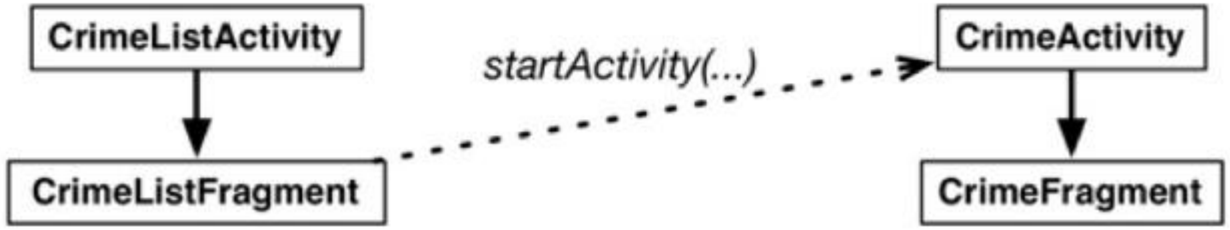
2:00

Crime #0 ✓
Fri Mar 27 16:09:13 EDT 2015
Crime #1 ☐
Fri Mar 27 16:09:13 EDT 2015
Crime #2 ✓
Fri Mar 27 16:09:13 EDT 2015
Crime #3 ☐
Fri Mar 27 16:09:13 EDT 2015
Crime #4 ✓
Fri Mar 27 16:09:13 EDT 2015
Crime #5 ☐
Fri Mar 27 16:09:13 EDT 2015
Crime #6 ✓
Fri Mar 27 16:09:13 EDT 2015
Crime #7 ☐
Fri Mar 27 16:09:13 EDT 2015
Crime #8 ✓
Fri Mar 27 16:09:13 EDT 2015
Crime #9 ☐
Fri Mar 27 16:09:13 EDT 2015
Crime #10 ✓

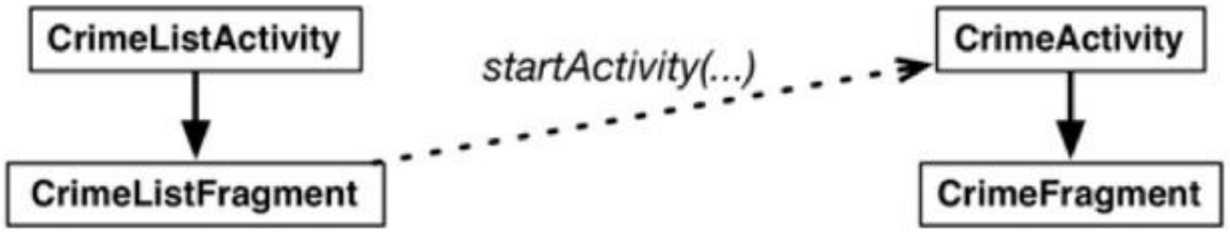


2:00

TITLE
Crime #4
DETAILS
MON MAR 30 14:34:23 GMT+00:00 2015
✓ Solved



Starting an Activity from a Fragment



```
private class CrimeHolder extends RecyclerView.ViewHolder
        implements View.OnClickListener {
    ...

    @Override
    public void onClick(View v) {
        ~~Toast.makeText(getActivity(),~~
                ~~mCrime.getTitle() + " clicked!", Toast.LENGTH_SHORT)~~
                ~~.show();~~

        Intent intent = new Intent(getActivity(), CrimeActivity.class);
        startActivity(intent);
    }
}
```

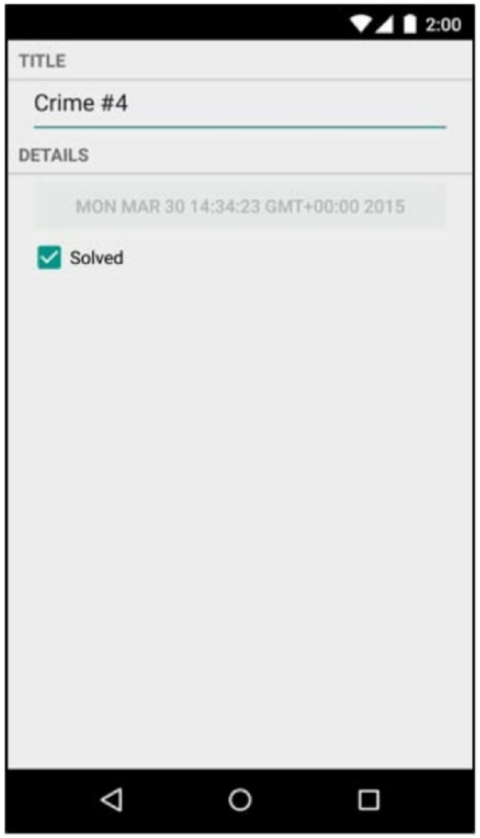
Passing Data to New Activity

```
public class CrimeActivity extends SingleFragmentActivity {

    public static final String EXTRA_CRIME_ID =
            "com.bignerdranch.android.criminalintent.crime_id";

    public static Intent newIntent(Context packageContext, UUID crimeId) {
        Intent intent = new Intent(packageContext, CrimeActivity.class);
        intent.putExtra(EXTRA_CRIME_ID, crimeId);
        return intent;
    }


private class CrimeHolder extends RecyclerView.ViewHolder
        implements View.OnClickListener {

    ...

    @Override
    public void onClick(View v) {
        ~~Intent intent = new Intent(getActivity(), CrimeActivity.class);~~
        Intent intent = CrimeActivity.newIntent(getActivity(), mCrime.getId());
        startActivity(intent);
    }
}
```

```
public class CrimeFragment extends Fragment {

    ...

    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        ~~mCrime = new Crime();~~
        UUID crimeId = (UUID) getActivity().getIntent()
                .getSerializableExtra(CrimeActivity.EXTRA_CRIME_ID);
        mCrime = CrimeLab.get(getActivity()).getCrime(crimeId);
    }
```

Passing Data to New Activity

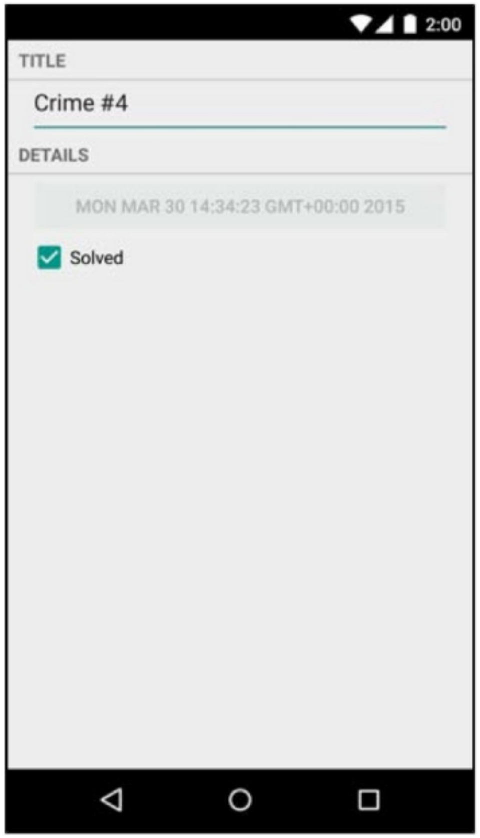
```
public class CrimeFragment extends Fragment {

    ...

    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        ~~mCrime = new Crime();~~
        UUID crimeId = (UUID) getActivity().getIntent()
                .getSerializableExtra(CrimeActivity.EXTRA_CRIME_ID);
        mCrime = CrimeLab.get(getActivity()).getCrime(crimeId);
    }

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

        mTitleField = (EditText)v.findViewById(R.id.crime_title);
        mTitleField.setText(mCrime.getTitle());
        mTitleField.addTextChangedListener(new TextWatcher() {
            ...
        });

        ...

        mSolvedCheckBox = (CheckBox)v.findViewById(R.id.crime_solved);
        mSolvedCheckBox.setChecked(mCrime.isSolved());
        mSolvedCheckBox.setOnCheckedChangeListener(new OnCheckedChangeListener() {
            ...
        });
```



- Fragment directly accessing its hosting activity's intent makes code simple, but breaks encapsulation—Fragment is not reusable.
- crime_id should not be in the hosting Activity's space.

Passing Data to New Activity

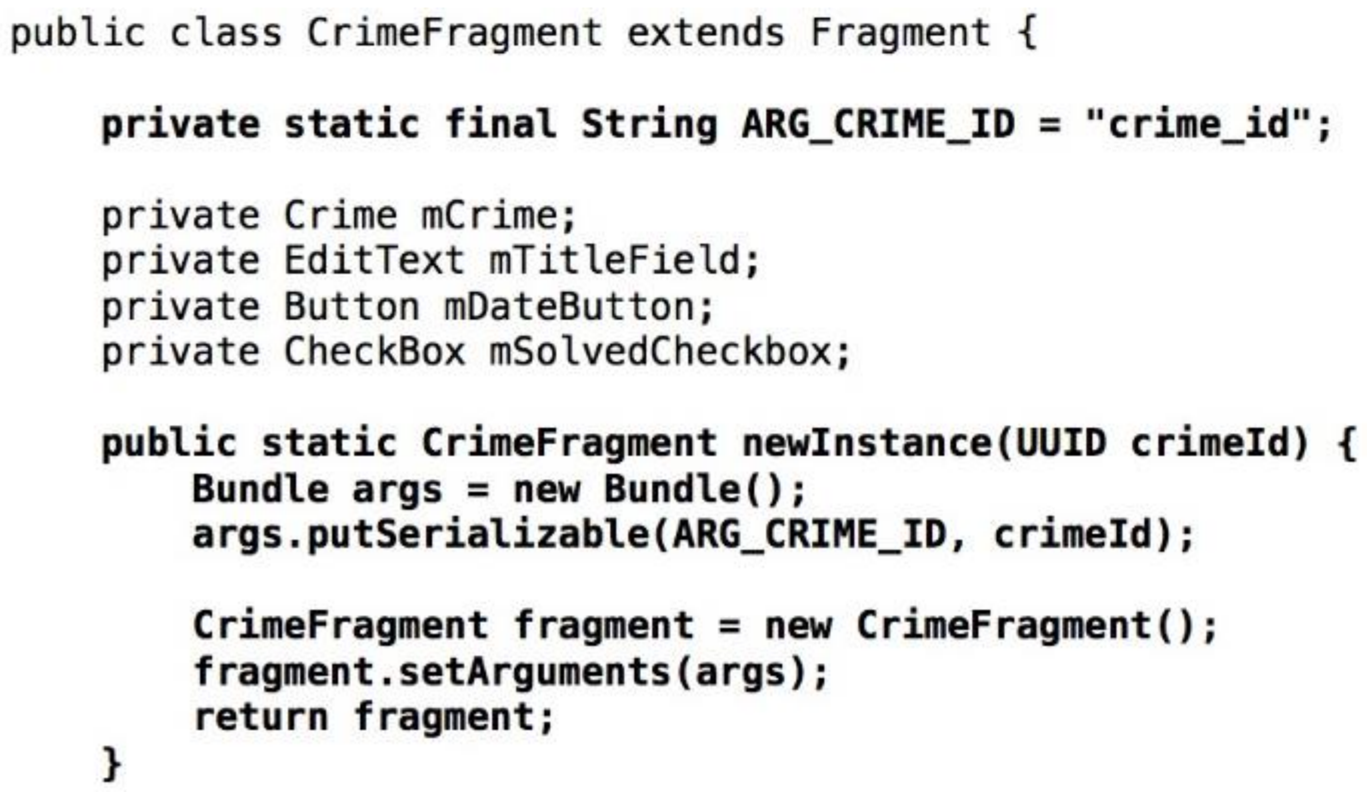
```
public class CrimeFragment extends Fragment {

    ...

    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        ~~mCrime = new Crime();~~
        UUID crimeId = (UUID) getActivity().getIntent()
                .getSerializableExtra(CrimeActivity.EXTRA_CRIME_ID);
        mCrime = CrimeLab.get(getActivity()).getCrime(crimeId);
    }

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

        mTitleField = (EditText)v.findViewById(R.id.crime_title);
        mTitleField.setText(mCrime.getTitle());
        mTitleField.addTextChangedListener(new TextWatcher() {
            ...
        });

        ...

        mSolvedCheckBox = (CheckBox)v.findViewById(R.id.crime_solved);
        mSolvedCheckBox.setChecked(mCrime.isSolved());
        mSolvedCheckBox.setOnCheckedChangeListener(new OnCheckedChangeListener() {
            ...
        });
```



- Fragment directly accessing its hosting activity's intent makes code simple, but breaks encapsulation—Fragment is not reusable.
- crime_id should not be in the hosting Activity's space.

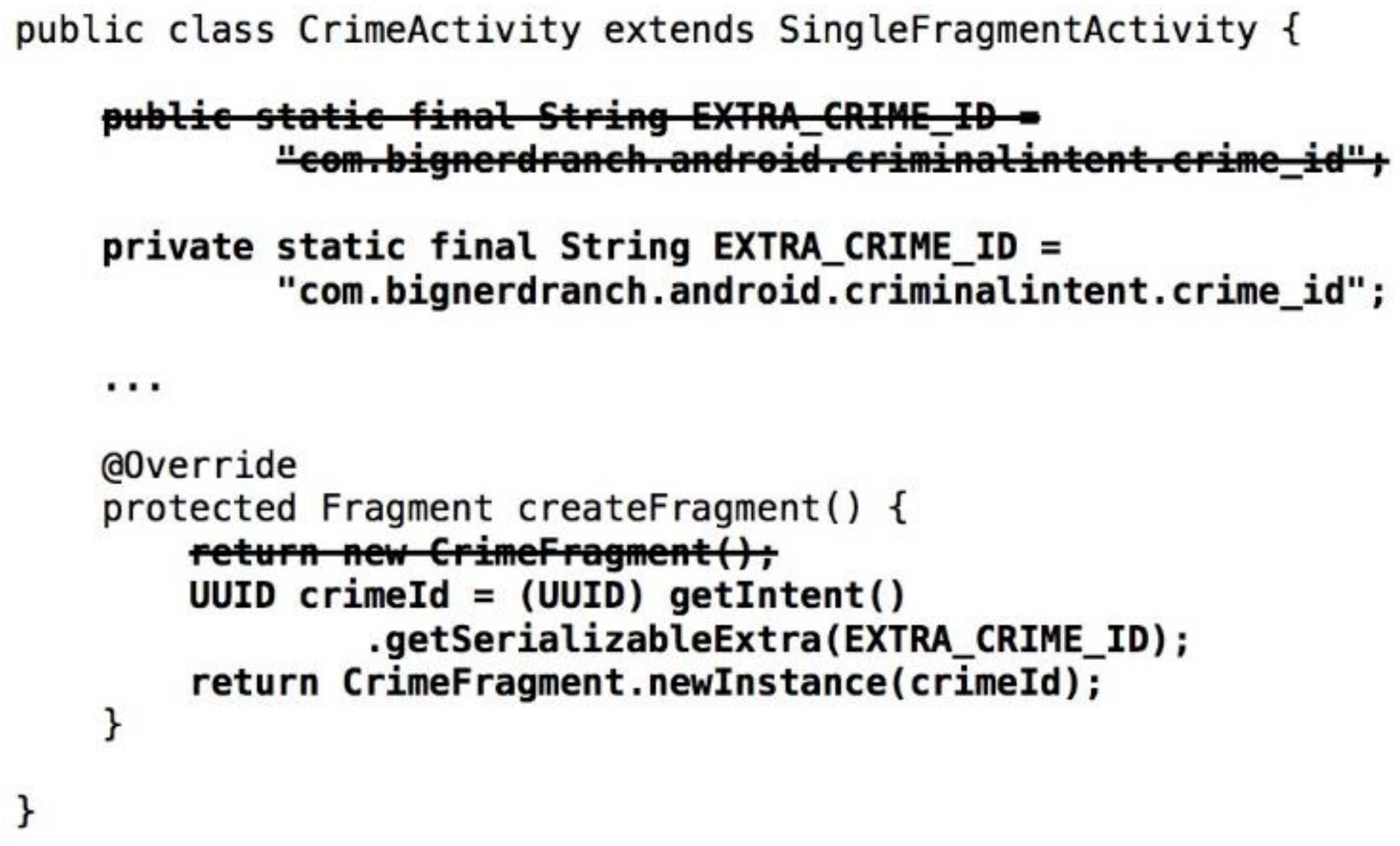
Fragment Arguments

- A fragment can have a **Bundle** object attached to it. A bundle contains key-value pairs and each pair is an argument.
- Attaching arguments to a fragment must be done before the fragment is added to an activity.
- The hosting activity can pass in required parameters the fragment needs to create its arguments through newInstance().



```
public class CrimeFragment extends Fragment {

    private static final String ARG_CRIME_ID = "crime_id";

    private Crime mCrime;
    private EditText mTitleField;
    private Button mDateButton;
    private CheckBox mSolvedCheckbox;

    public static CrimeFragment newInstance(UUID crimeId) {
        Bundle args = new Bundle();
        args.putSerializable(ARG_CRIME_ID, crimeId);

        CrimeFragment fragment = new CrimeFragment();
        fragment.setArguments(args);
        return fragment;
    }
```

Fragment Arguments

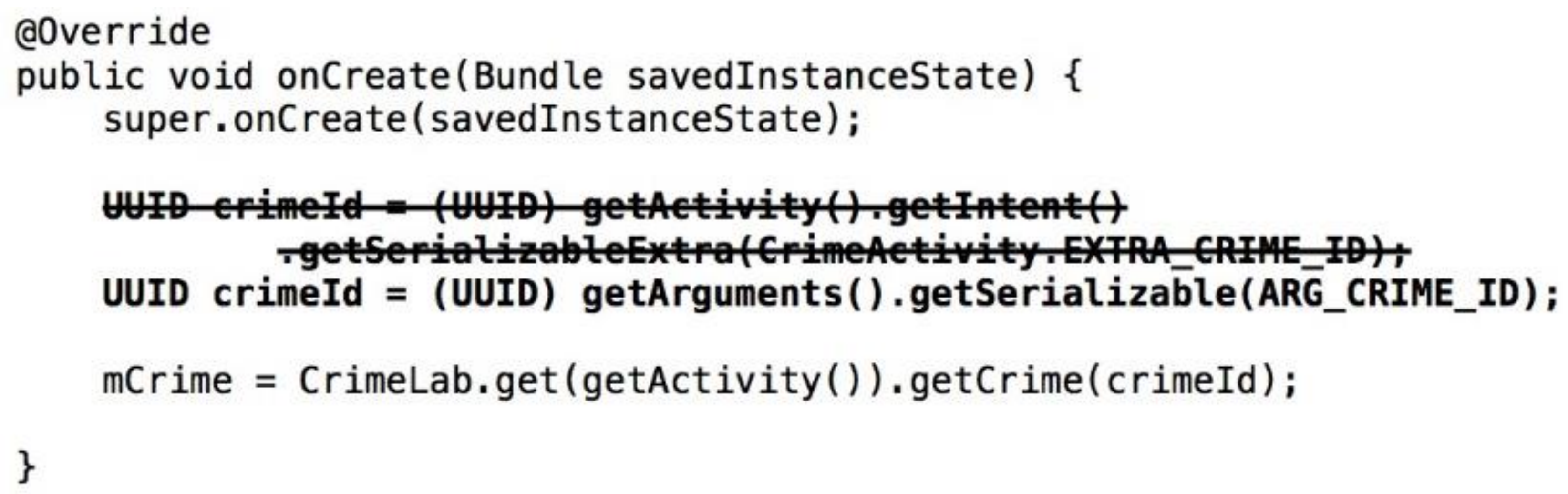


```
public class CrimeActivity extends SingleFragmentActivity {

    ~~public static final String EXTRA_CRIME_ID =~~
            ~~"com.bignerdranch.android.criminalintent.crime_id";~~

    private static final String EXTRA_CRIME_ID =
            "com.bignerdranch.android.criminalintent.crime_id";

    ...

    @Override
    protected Fragment createFragment() {
        ~~return new CrimeFragment();~~
        UUID crimeId = (UUID) getIntent()
                .getSerializableExtra(EXTRA_CRIME_ID);
        return CrimeFragment.newInstance(crimeId);
    }

}
```

- Hosting activities should know how to host their fragments, but fragments do not have to know specifics about their activities.

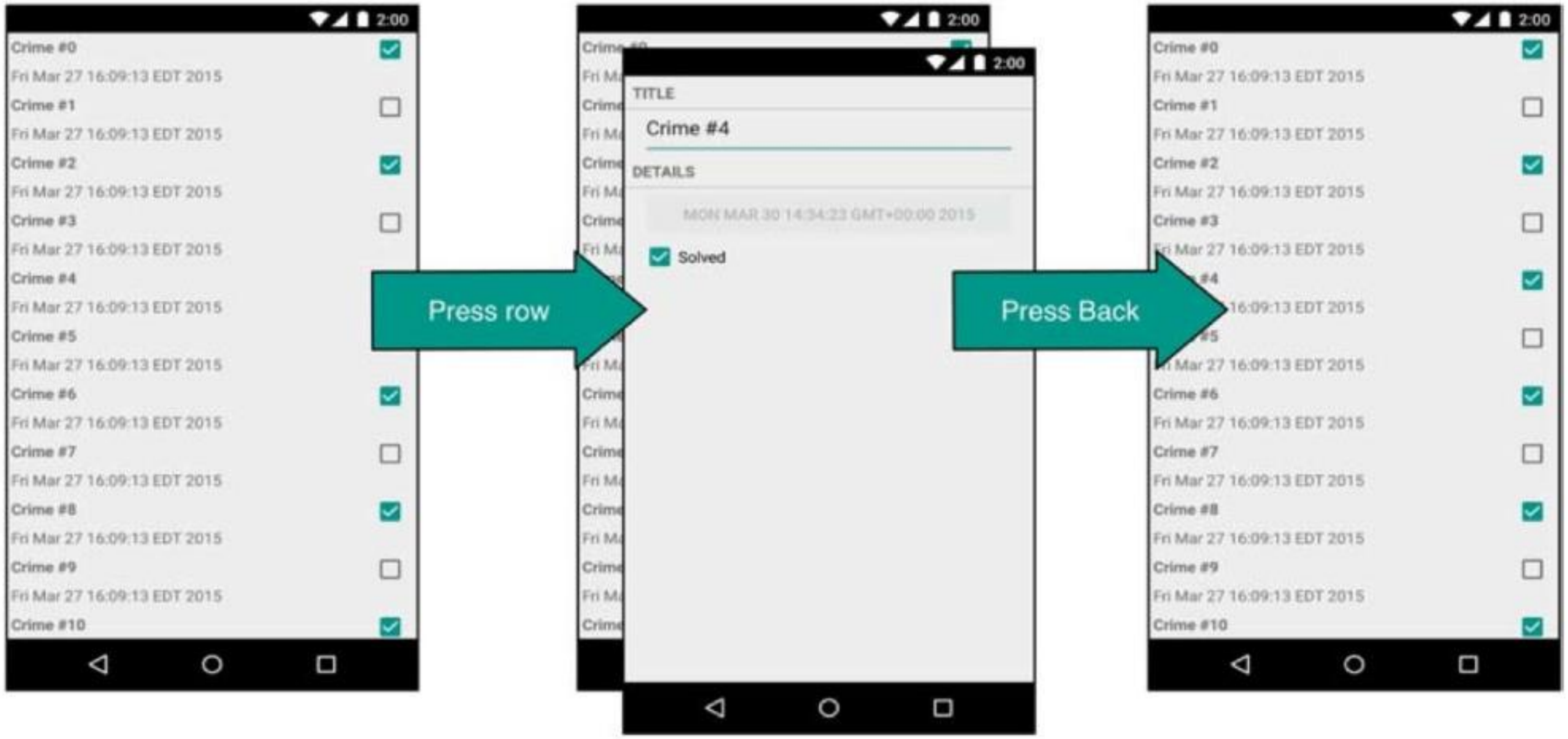
Retrieving Arguments

- Call getArguments() and then one of the type-specific “get” methods of Bundle in Fragment.



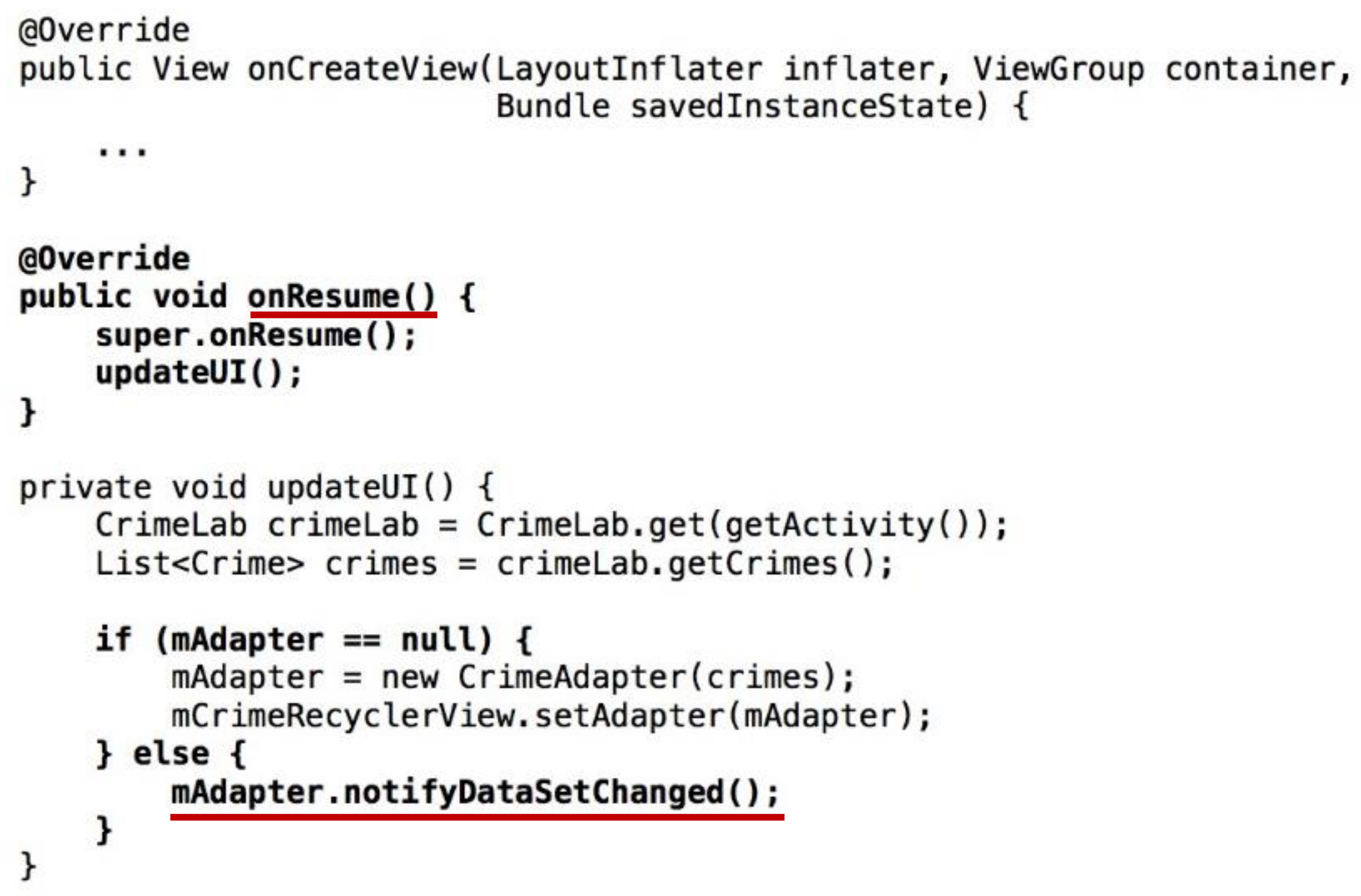
```
@Override
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);

    ~~UUID crimeId = (UUID) getActivity().getIntent()~~
            ~~.getSerializableExtra(CrimeActivity.EXTRA_CRIME_ID);~~
    UUID crimeId = (UUID) getArguments().getSerializable(ARG_CRIME_ID);

    mCrime = CrimeLab.get(getActivity()).getCrime(crimeId);

}
```

Data Updating & Reloading


Press row
Press Back
TITLE
Crime #4
DETAILS
MON MAR 30 14:34:23 GMT+00:00 2015
Solved
Crime #0
Fri Mar 27 16:09:13 EDT 2015

- Problem: press a list item, modify that Crime’s details, then return to the list. However, the RecyclerView is unchanged.
- The RecyclerView’s Adapter needs to be informed that the data has changed so that it can fetch the updated data and reload the list.

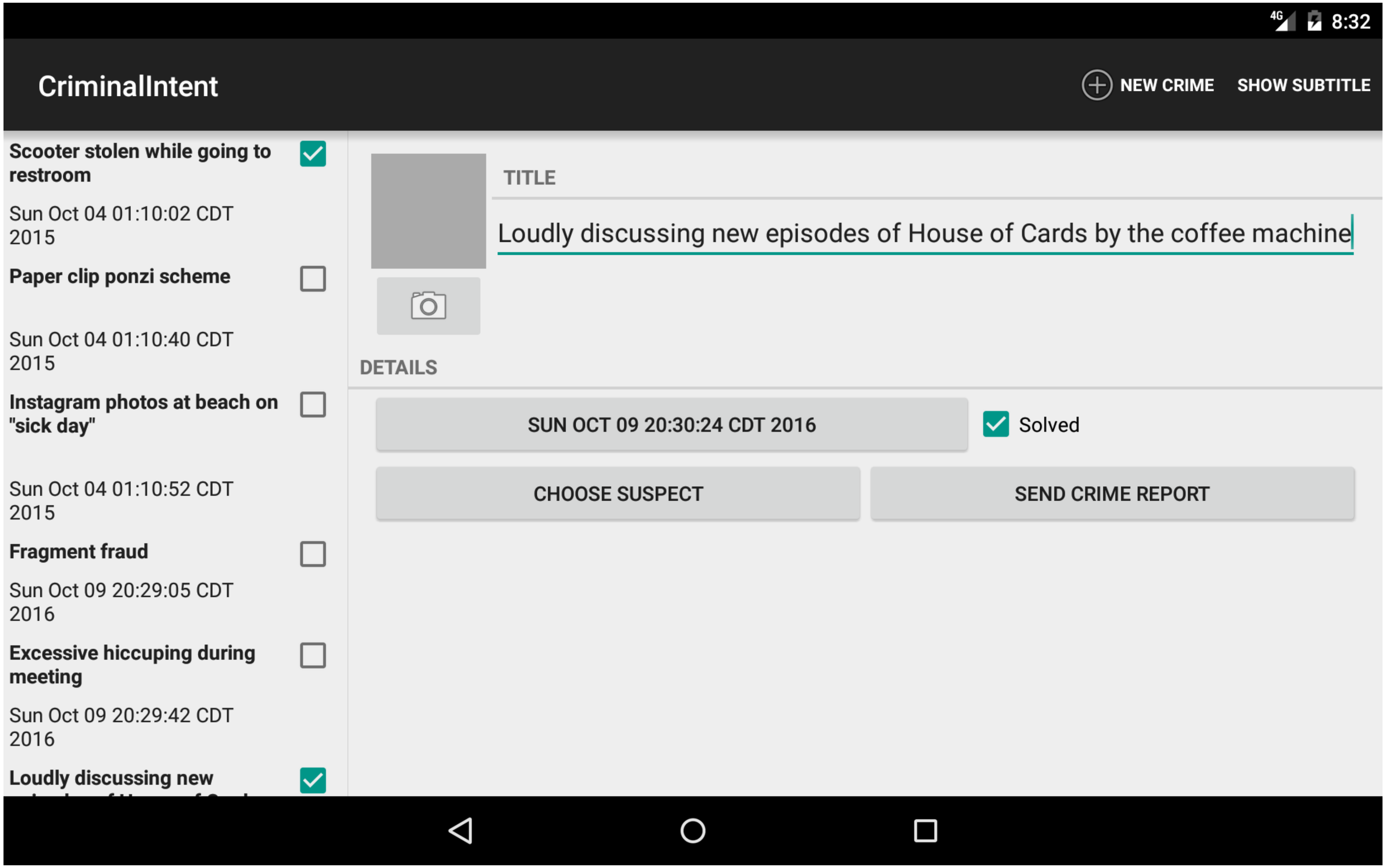
Data Updating & Reloading



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

@Override
public void onResume() {
    super.onResume();
    updateUI();
}

private void updateUI() {
    CrimeLab crimeLab = CrimeLab.get(getActivity());
    List<Crime> crimes = crimeLab.getCrimes();

    if (mAdapter == null) {
        mAdapter = new CrimeAdapter(crimes);
        mCrimeRecyclerView.setAdapter(mAdapter);
    } else {
        mAdapter.notifyDataSetChanged();
    }
}
```

Two-Pane Master-Detail UI

Screenshot from Tablet

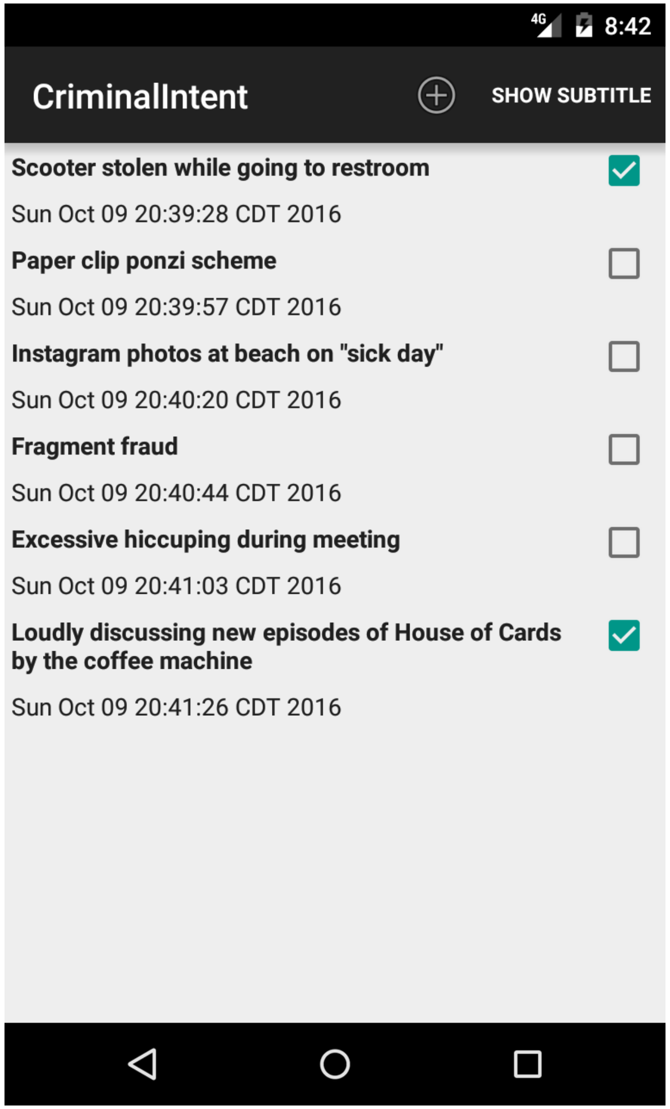


Scooter stolen while going to restroom Sun Oct 04 01:10:02 CDT 2015	✓
Paper clip ponzi scheme Sun Oct 04 01:10:40 CDT 2015	☐
Instagram photos at beach on "sick day" Sun Oct 04 01:10:52 CDT 2015	☐
Fragment fraud Sun Oct 09 20:29:05 CDT 2016	☐
Excessive hiccuping during meeting Sun Oct 09 20:29:42 CDT 2016	☐
Loudly discussing new	✓

Loudly discussing new episodes of House of Cards by the coffee machine

✓ Solved

Screenshot from Phone


8:42

CriminalIntent

SHOW SUBTITLE

Scooter stolen while going to restroom ✓
Sun Oct 09 20:39:28 CDT 2016

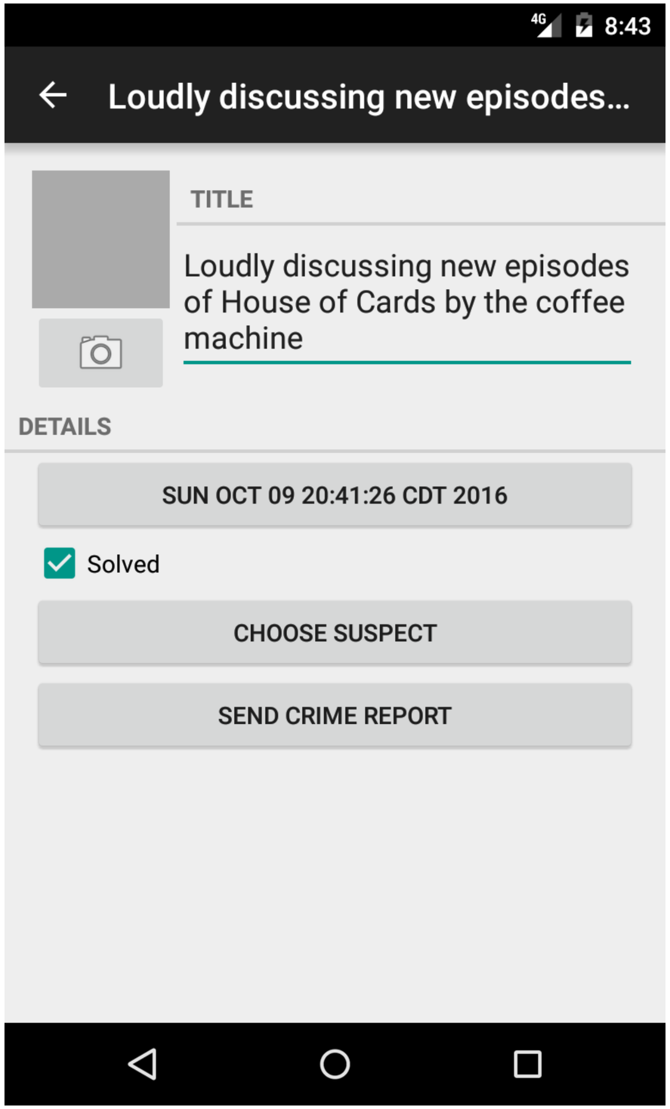
Paper clip ponzi scheme ☐
Sun Oct 09 20:39:57 CDT 2016

Instagram photos at beach on "sick day" ☐
Sun Oct 09 20:40:20 CDT 2016

Fragment fraud ☐
Sun Oct 09 20:40:44 CDT 2016

Excessive hiccuping during meeting ☐
Sun Oct 09 20:41:03 CDT 2016

Loudly discussing new episodes of House of Cards by the coffee machine ✓
Sun Oct 09 20:41:26 CDT 2016


8:43

Loudly discussing new episodes...

TITLE

Loudly discussing new episodes of House of Cards by the coffee machine

DETAILS

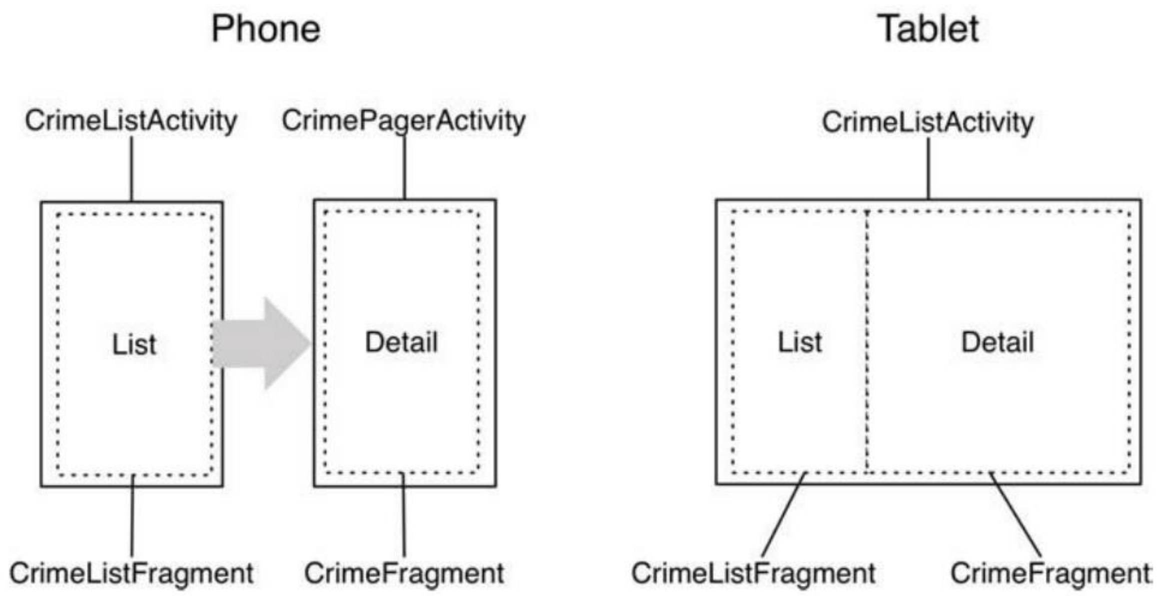
SUN OCT 09 20:41:26 CDT 2016

✓ Solved

CHOOSE SUSPECT

SEND CRIME REPORT

Master-Detail UI for Phone & Tablet



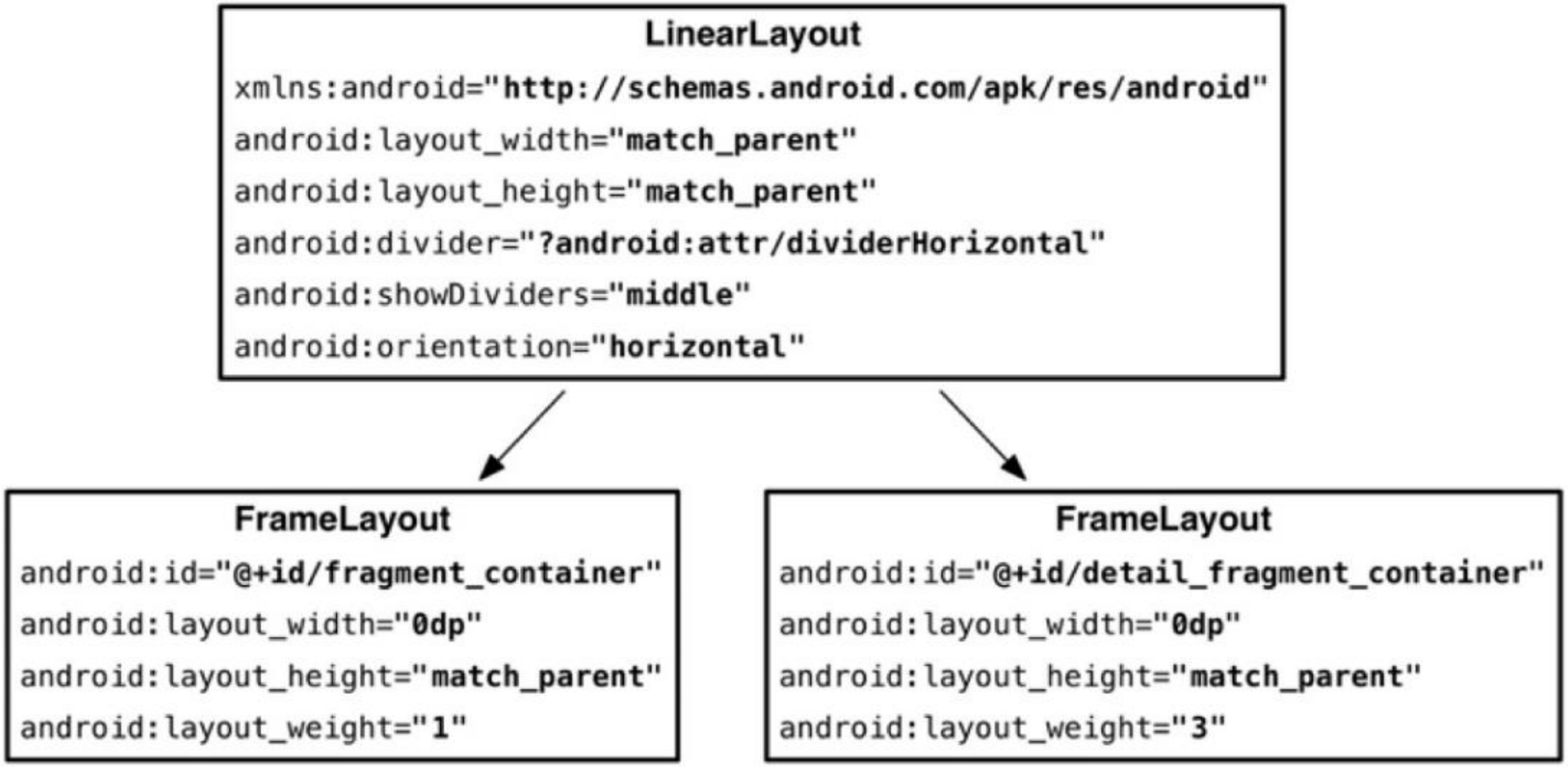
- A new layout consisting of two fragment containers is needed.
- **CrimeListActivity** now inflates a single-container layout on phones and a two-container layout on tablets.

Modifying SingleFragmentActivity

- In SingleFragmentActivity.java, add a protected method that returns the ID of the layout that the activity will inflate.

```
@LayoutRes
protected int getLayoutResId() {
    return R.layout.activity_fragment;
}

@Override
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    ~~setContentView(R.layout.activity_fragment);~~
    setContentView(getLayoutResId());
```

A Layout with Two Fragment Containers



```
LinearLayout
xmlns:android="http://schemas.android.com/apk/res/android"
android:layout_width="match_parent"
android:layout_height="match_parent"
android:divider="?android:attr/dividerHorizontal"
android:showDividers="middle"
android:orientation="horizontal"
```

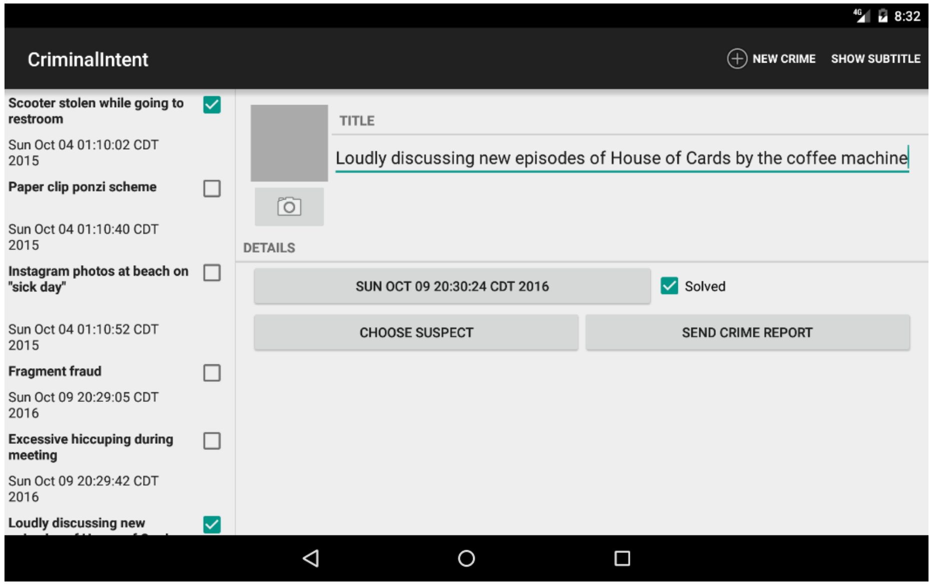
```
FrameLayout
android:id="@+id/fragment_container"
android:layout_width="0dp"
android:layout_height="match_parent"
android:layout_weight="1"
```

```
FrameLayout
android:id="@+id/detail_fragment_container"
android:layout_width="0dp"
android:layout_height="match_parent"
android:layout_weight="3"
```

- Create a new layout: layout/activity_twopane.xml
- In CrimeListActivity.class: return R.layout.activity_twopane in getLayoutResId()

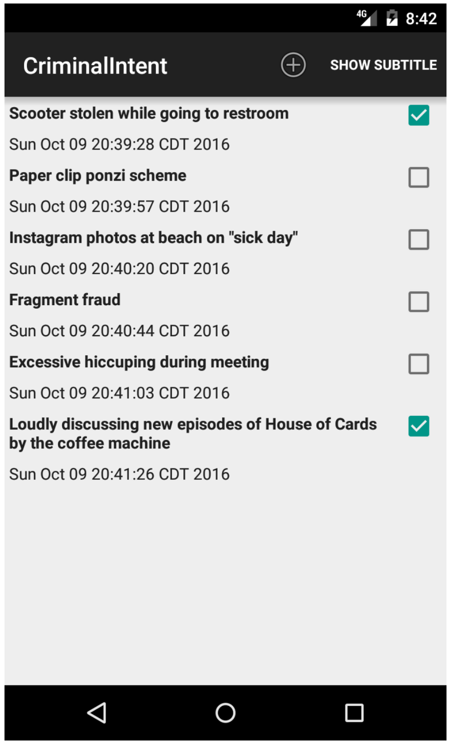
Using Alias Resource

- CrimeListActivity should render different layouts for tablets and phones.
  - Tablet: activity_twopane.xml
  - Phone: activity_fragment.xml
- Create an alias resource that points to activity_fragment.xml on phones and activity_twopane.xml on tablets.
  - An alias resource is a resource pointing to another resource
  - Alias resources are usually defined in **res/values/refs.xml**



Scooter stolen while going to restroom ✓
Sun Oct 04 01:10:02 CDT 2015
Paper clip ponzi scheme ☐
Sun Oct 04 01:10:40 CDT 2015
Instagram photos at beach on "sick day" ☐
Sun Oct 04 01:10:52 CDT 2015
Fragment fraud ☐
Sun Oct 09 20:29:05 CDT 2016
Excessive hiccuping during meeting ☐
Sun Oct 09 20:29:42 CDT 2016
Loudly discussing new ✓

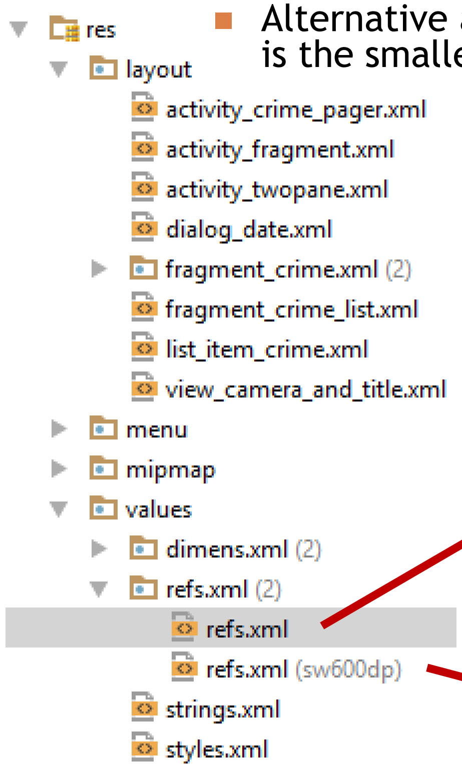
Loudly discussing new episodes of House of Cards by the coffee machine
SUN OCT 09 20:30:24 CDT 2016
✓ Solved

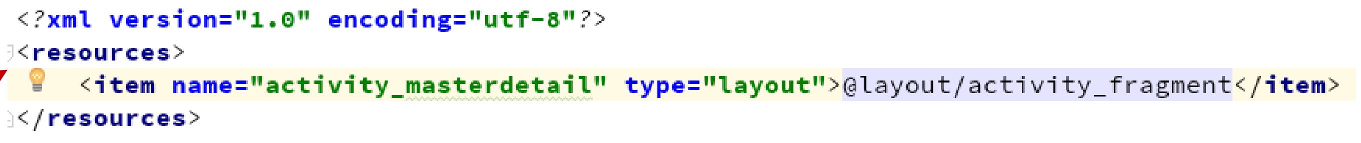


Scooter stolen while going to restroom ✓
Sun Oct 09 20:39:28 CDT 2016
Paper clip ponzi scheme ☐
Sun Oct 09 20:39:57 CDT 2016
Instagram photos at beach on "sick day" ☐
Sun Oct 09 20:40:20 CDT 2016
Fragment fraud ☐
Sun Oct 09 20:40:44 CDT 2016
Excessive hiccuping during meeting ☐
Sun Oct 09 20:41:03 CDT 2016
Loudly discussing new episodes of House of Cards by the coffee machine ✓
Sun Oct 09 20:41:26 CDT 2016

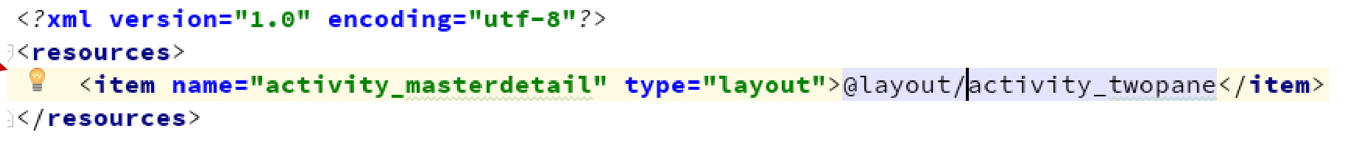
Using Alias Resource

- CrimeListActivity should render different layouts for tablets and phones.
  - Tablet: activity_twopane.xml
  - Phone: activity_fragment.xml
- Create an alias resource that points to activity_fragment.xml on phones and activity_twopane.xml on tablets.
  - An alias resource is a resource pointing to another resource
  - Alias resources for default layout are usually defined in **res/values/refs.xml**
    - Alternative alias for larger screen devices are in **res/values-swXXXdp/refs.xml** where XXX is the smallest screen width.





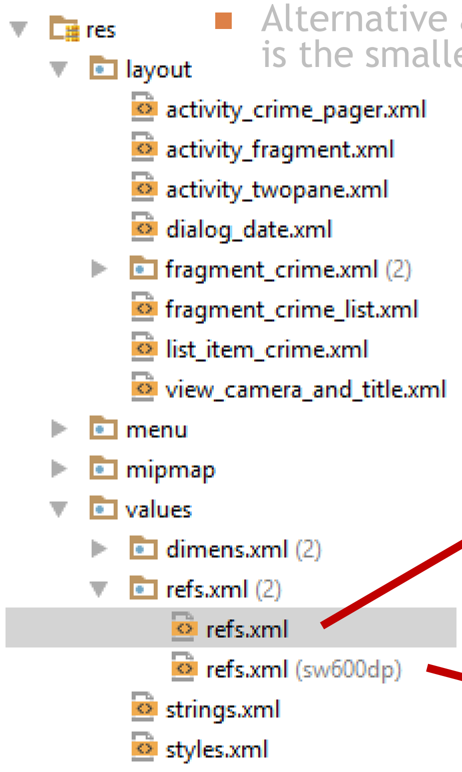
```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <item name="activity_masterdetail" type="layout">@layout/activity_fragment</item>
</resources>
```



```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <item name="activity_masterdetail" type="layout">@layout/activity_twopane</item>
</resources>
```

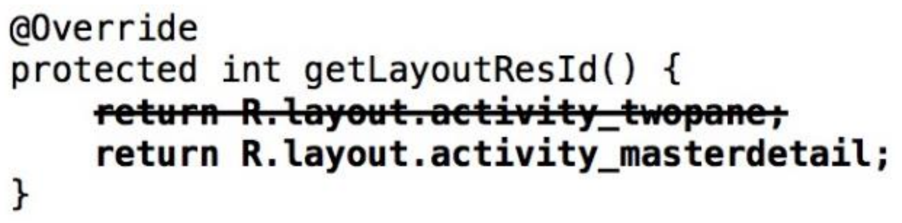
Using Alias Resource

- CrimeListActivity should render different layouts for tablets and phones.
  - Tablet: activity_twopane.xml
  - Phone: activity_fragment.xml
- Create an alias resource that points to activity_fragment.xml on phones and activity_twopane.xml on tablets.
  - An alias resource is a resource pointing to another resource
  - Alias resources for default layout are usually defined in **res/values/refs.xml**
    - Alternative alias for larger screen devices are in **res/values-swXXXdp/refs.xml** where XXX is the smallest screen width.

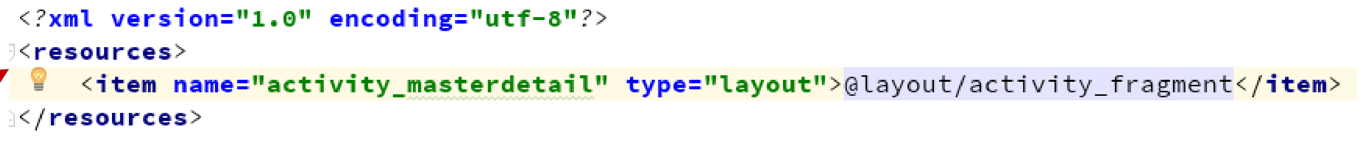


```
res
  layout
    activity_crime_pager.xml
    activity_fragment.xml
    activity_twopane.xml
    dialog_date.xml
    fragment_crime.xml (2)
    fragment_crime_list.xml
    list_item_crime.xml
    view_camera_and_title.xml
  menu
  mipmap
  values
    dimens.xml (2)
    refs.xml (2)
      refs.xml
      refs.xml (sw600dp)
    strings.xml
    styles.xml
```

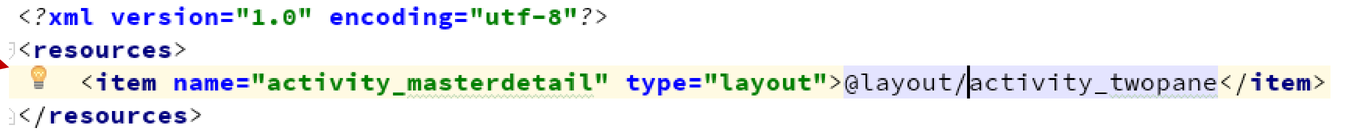
Switch layout in CrimeListActivity class



```
@Override
protected int getLayoutResId() {
    ~~return R.layout.activity_twopane;~~
    return R.layout.activity_masterdetail;
}
```

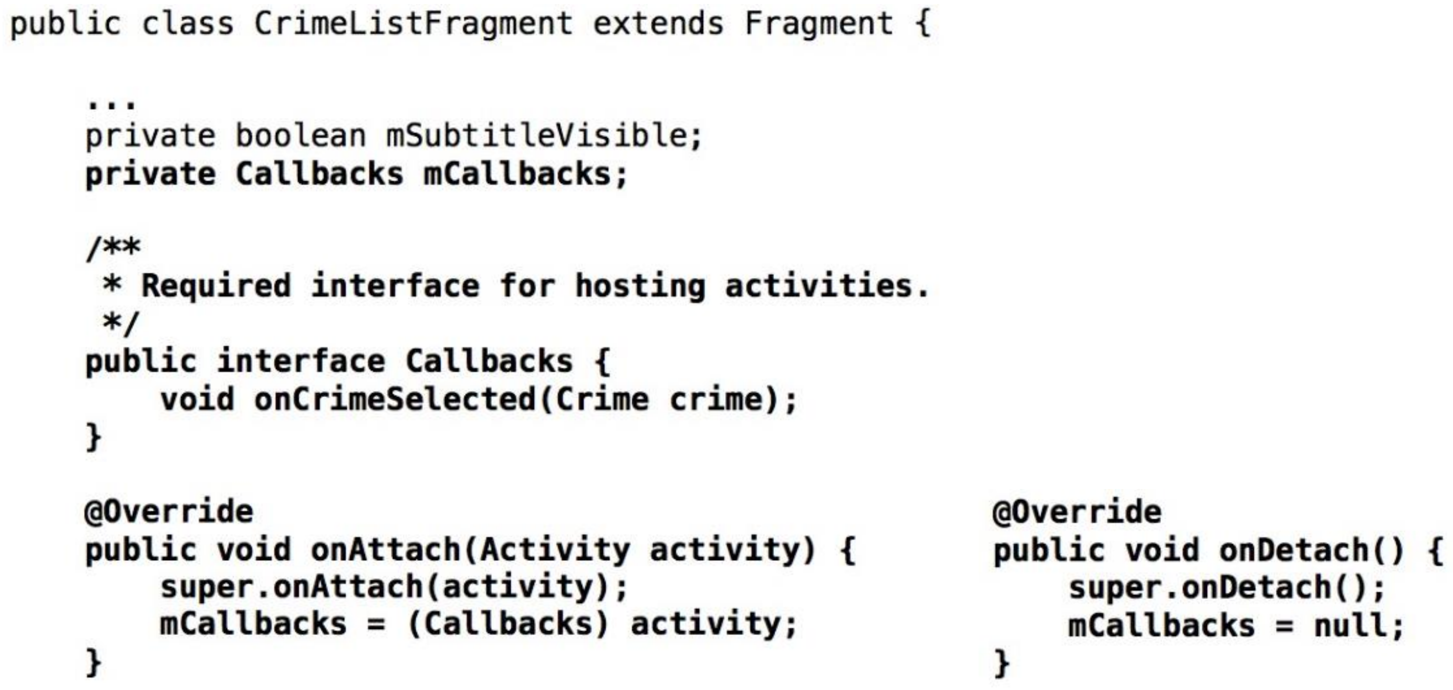


```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <item name="activity_masterdetail" type="layout">@layout/activity_fragment</item>
</resources>
```



```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <item name="activity_masterdetail" type="layout">@layout/activity_twopane</item>
</resources>
```

Fragment Callback Interfaces

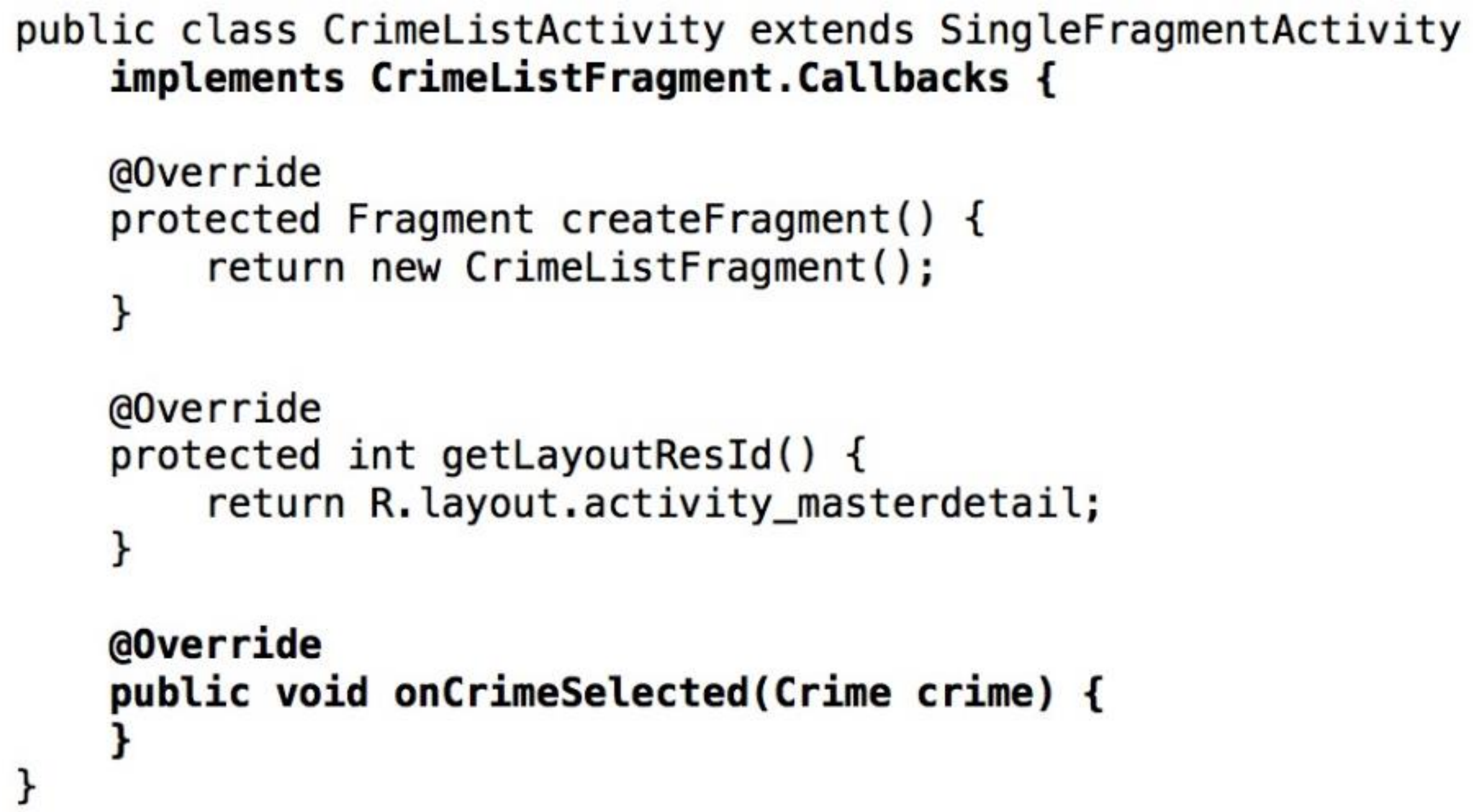


```
public class CrimeListFragment extends Fragment {

    ...
    private boolean mSubtitleVisible;
    private Callbacks mCallbacks;

    /**
     * Required interface for hosting activities.
     */
    public interface Callbacks {
        void onCrimeSelected(Crime crime);
    }

    @Override
    public void onAttach(Activity activity) {
        super.onAttach(activity);
        mCallbacks = (Callbacks) activity;
    }
```

```
    @Override
    public void onDetach() {
        super.onDetach();
        mCallbacks = null;
    }
```

- Fragment callback interface defines work that needs to be done by the hosting activity.
- With a callback interface, a fragment is able to call its hosting activity without knowing anything about its hosting activity.
- To implement **Callbacks**, first define a member field holding an object that implements **Callbacks**; then cast the hosting activity to **Callbacks** to assign it to the field.

Implementing Callbacks in Activity



```
public class CrimeListActivity extends SingleFragmentActivity
    implements CrimeListFragment.Callbacks {

    @Override
    protected Fragment createFragment() {
        return new CrimeListFragment();
    }

    @Override
    protected int getLayoutResId() {
        return R.layout.activity_masterdetail;
    }

    @Override
    public void onCrimeSelected(Crime crime) {
    }
}
```

- When onCrimeSelected() is called, CrimeListActivity needs to
  - Start **CrimePagerActivity** if using phone interface, or
  - Put **CrimeFragment** in **detail_fragment_container** if using tablet interface

Implementing Callbacks in Activity

- When onCrimeSelected() is called, CrimeListActivity needs to
  - Start **CrimePagerActivity** if using phone interface, or
  - Put **CrimeFragment** in **detail_fragment_container** if using tablet interface
- How?
  - Check if the layout has a detail_fragment_container. If yes, add CrimeFragment if it does not exist.

```
@Override
public void onCrimeSelected(Crime crime) {
    if (findViewById(R.id.detail_fragment_container) == null) {
        Intent intent = CrimePagerActivity.newIntent(this, crime.getId());
        startActivity(intent);
    } else {
        Fragment newDetail = CrimeFragment.newInstance(crime.getId());

        getSupportFragmentManager().beginTransaction()
                .replace(R.id.detail_fragment_container, newDetail)
                .commit();
    }
}
```

Implementing Callbacks in Activity

- In CrimeListFragment

```
@Override
public boolean onOptionsItemSelected(MenuItem item) {
    switch (item.getItemId()) {
        case R.id.menu_item_new_crime:
            Crime crime = new Crime();
            CrimeLab.get(getActivity()).addCrime(crime);
            ~~Intent intent = CrimePagerActivity~~
                    ~~.newIntent(getActivity(), crime.getId());~~
            ~~startActivity(intent);~~
            updateUI();
            mCallbacks.onCrimeSelected(crime);
            return true;
```

```
private class CrimeHolder extends RecyclerView.ViewHolder
        implements View.OnClickListener {

    ...

    @Override
    public void onClick(View v) {
        ~~Intent intent = CrimePagerActivity.newIntent(getActivity(), mCrime.getId());~~
        ~~startActivity(intent);~~
        mCallbacks.onCrimeSelected(mCrime);
    }
}
```

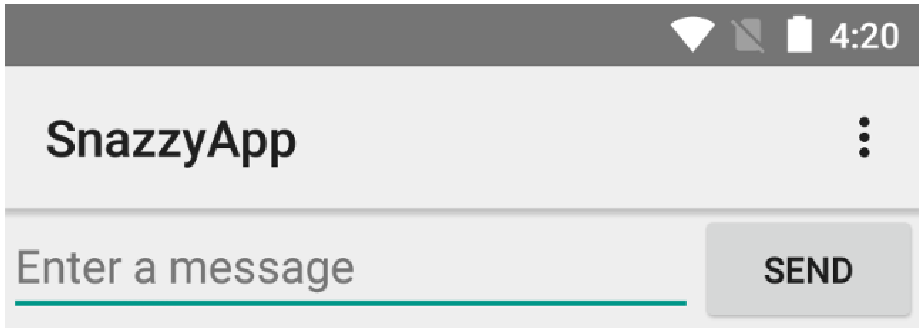
Toolbar

Toolbar vs. Action Bar

- Action Bar was added in Android 3.0 (API level 11). However, the native Action Bar behaves differently on different Android versions.

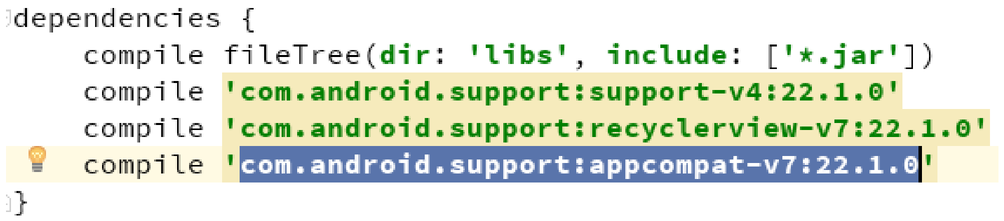
- Toolbar is a new addition to Android as of Android 5.0 (Lollipop). But the **v7 appcompat** support library's Toolbar has consistent behavior across the widest range of devices.

- Should always use the support library's **Toolbar**.



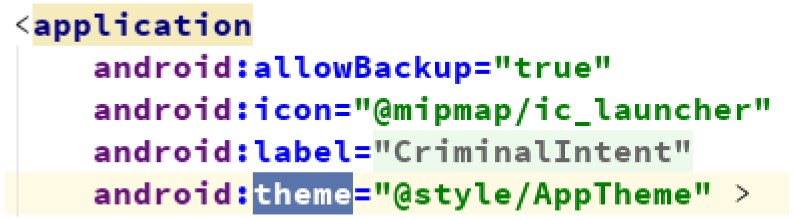
Using AppCompat Library

- Add AppCompat dependency in app/gradle.build

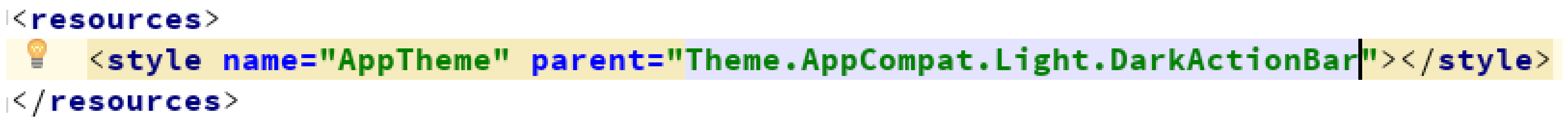


```
dependencies {
    compile fileTree(dir: 'libs', include: ['*.jar'])
    compile 'com.android.support:support-v4:22.1.0'
    compile 'com.android.support:recyclerview-v7:22.1.0'
    compile 'com.android.support:appcompat-v7:22.1.0'
}
```

- Use one of AppCompat themes
  - Theme.AppCompat – a dark theme
  - Theme.AppCompat.Light – a light theme
  - Theme.AppCompat.Light.DarkActionBar – a light theme with a dark toolbar

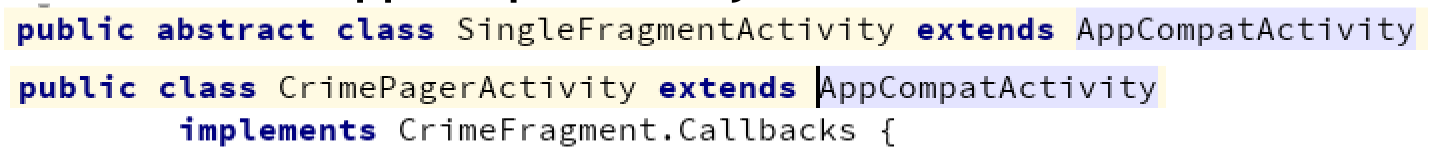


```
<application
    android:allowBackup="true"
    android:icon="@mipmap/ic_launcher"
    android:label="CriminalIntent"
    android:theme="@style/AppTheme" >
```



```
<resources>
    <style name="AppTheme" parent="Theme.AppCompat.Light.DarkActionBar"></style>
</resources>
```

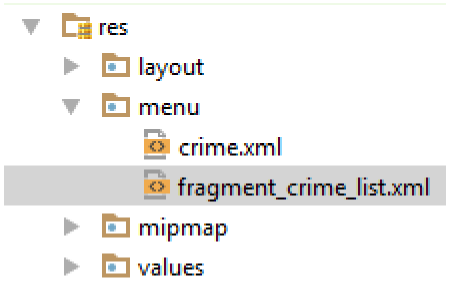
- Activities extend **AppCompatActivity**



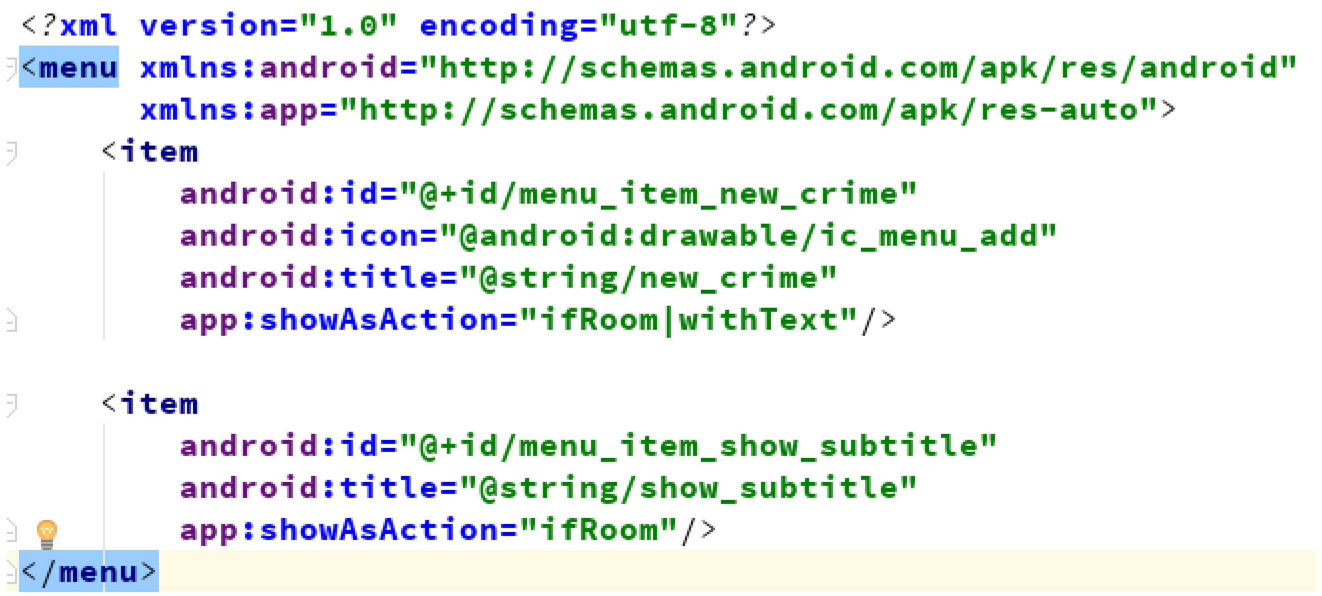
```
public abstract class SingleFragmentActivity extends AppCompatActivity

public class CrimePagerActivity extends AppCompatActivity
        implements CrimeFragment.Callbacks {
```

Menus

- The top-right area of a toolbar is reserved for the toolbar's menu.
- A menu consists of action items.
  - showAsAction attribute refers to whether an item appears in the toolbar or in the overflow menu.
  - The overflow menu is accessed through the three dots on the far-right side of the toolbar.



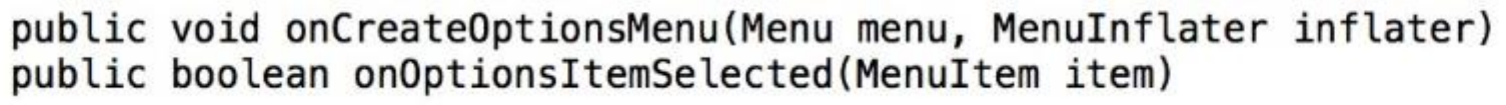
```
res
  layout
  menu
    crime.xml
    fragment_crime_list.xml
  mipmap
  values
```



```xml
<?xml version="1.0" encoding="utf-8"?>
<menu xmlns:android="http://schemas.android.com/apk/res/android"
      xmlns:app="http://schemas.android.com/apk/res-auto">
    <item
        android:id="@+id/menu_item_new_crime"
        android:icon="@android:drawable/ic_menu_add"
        android:title="@string/new_crime"
        app:showAsAction="ifRoom|withText"/>

    <item
        android:id="@+id/menu_item_show_subtitle"
        android:title="@string/show_subtitle"
        app:showAsAction="ifRoom"/>
</menu>
```

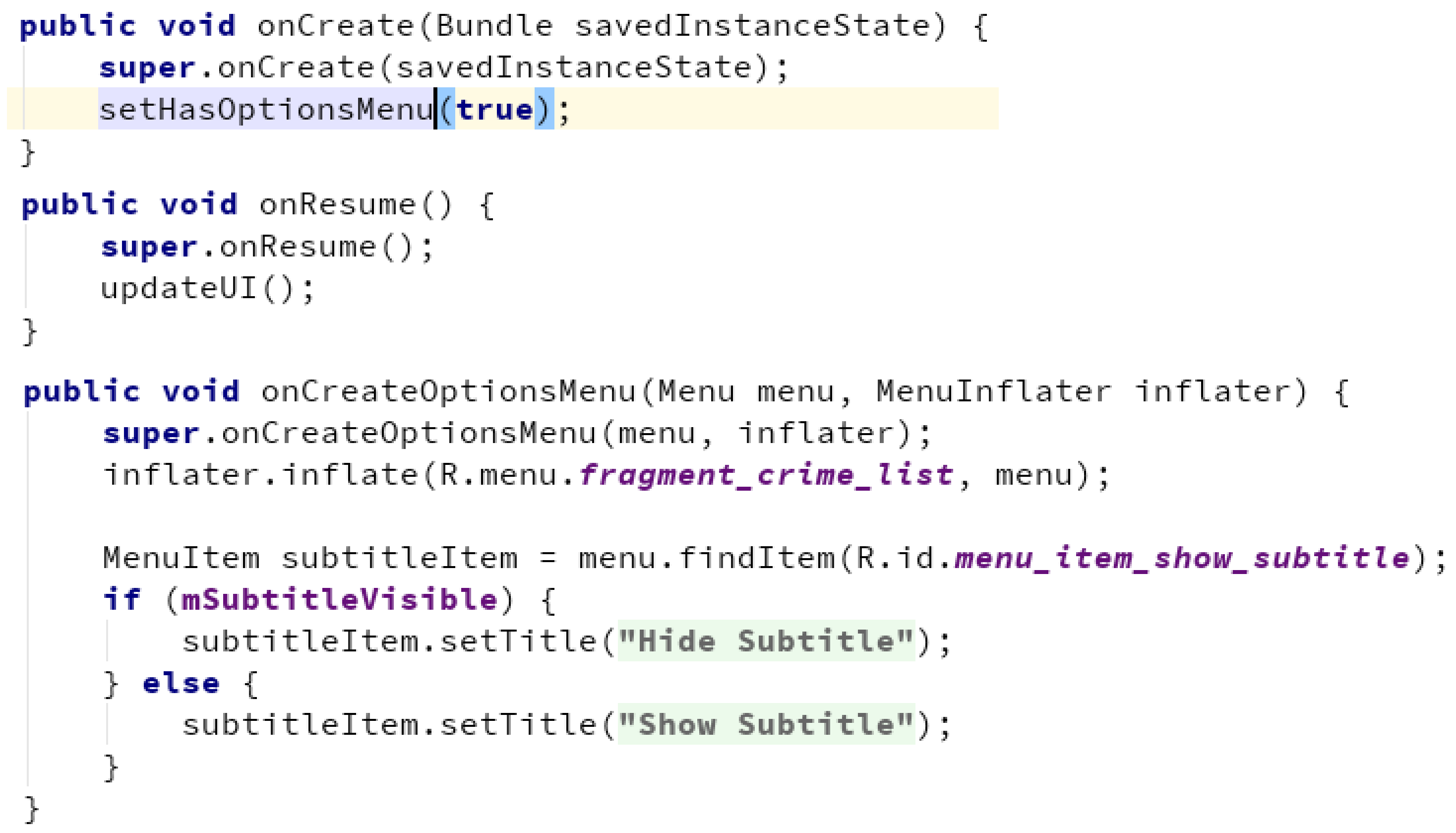
Menu Creation

- Implement menu callbacks in Fragment/Activity



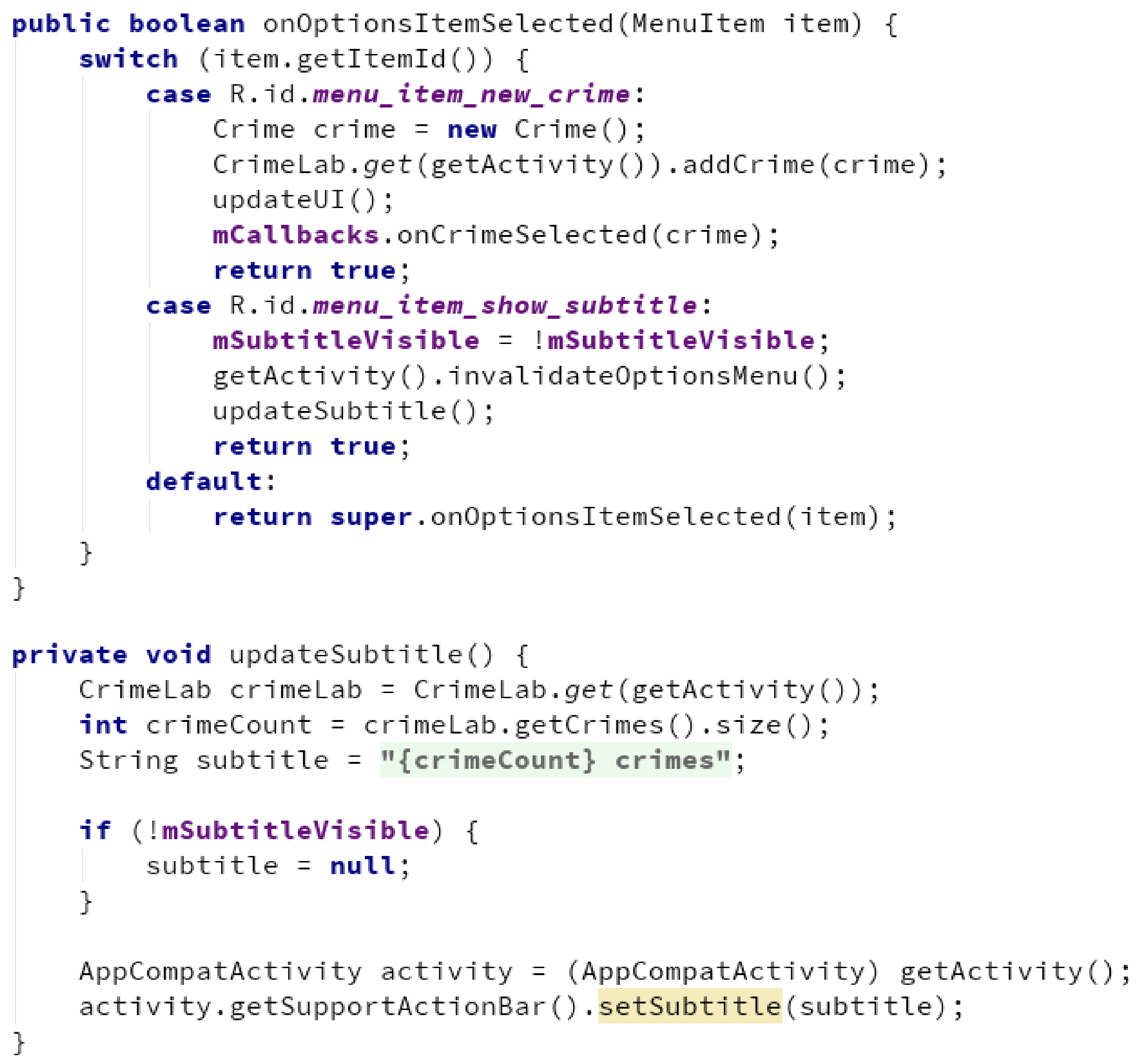
```
public void onCreateOptionsMenu(Menu menu, MenuInflater inflater)
public boolean onOptionsItemSelected(MenuItem item)
```

In CrimeListFragment.java



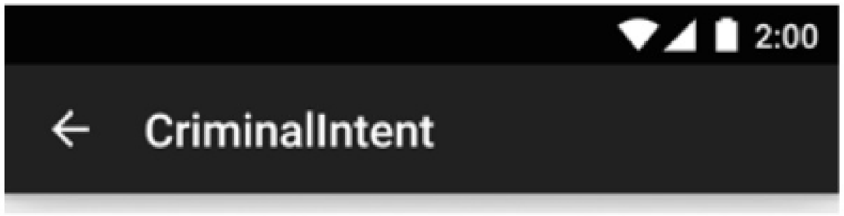
```
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setHasOptionsMenu(true);
}
public void onResume() {
    super.onResume();
    updateUI();
}

public void onCreateOptionsMenu(Menu menu, MenuInflater inflater) {
    super.onCreateOptionsMenu(menu, inflater);
    inflater.inflate(R.menu.fragment_crime_list, menu);

    MenuItem subtitleItem = menu.findItem(R.id.menu_item_show_subtitle);
    if (mSubtitleVisible) {
        subtitleItem.setTitle("Hide Subtitle");
    } else {
        subtitleItem.setTitle("Show Subtitle");
    }
}
```

Responding to Menu Selection



```java
public boolean onOptionsItemSelected(MenuItem item) {
    switch (item.getItemId()) {
        case R.id.menu_item_new_crime:
            Crime crime = new Crime();
            CrimeLab.get(getActivity()).addCrime(crime);
            updateUI();
            mCallbacks.onCrimeSelected(crime);
            return true;
        case R.id.menu_item_show_subtitle:
            mSubtitleVisible = !mSubtitleVisible;
            getActivity().invalidateOptionsMenu();
            updateSubtitle();
            return true;
        default:
            return super.onOptionsItemSelected(item);
    }
}

private void updateSubtitle() {
    CrimeLab crimeLab = CrimeLab.get(getActivity());
    int crimeCount = crimeLab.getCrimes().size();
    String subtitle = "{crimeCount} crimes";

    if (!mSubtitleVisible) {
        subtitle = null;
    }

    AppCompatActivity activity = (AppCompatActivity) getActivity();
    activity.getSupportActionBar().setSubtitle(subtitle);
}
```

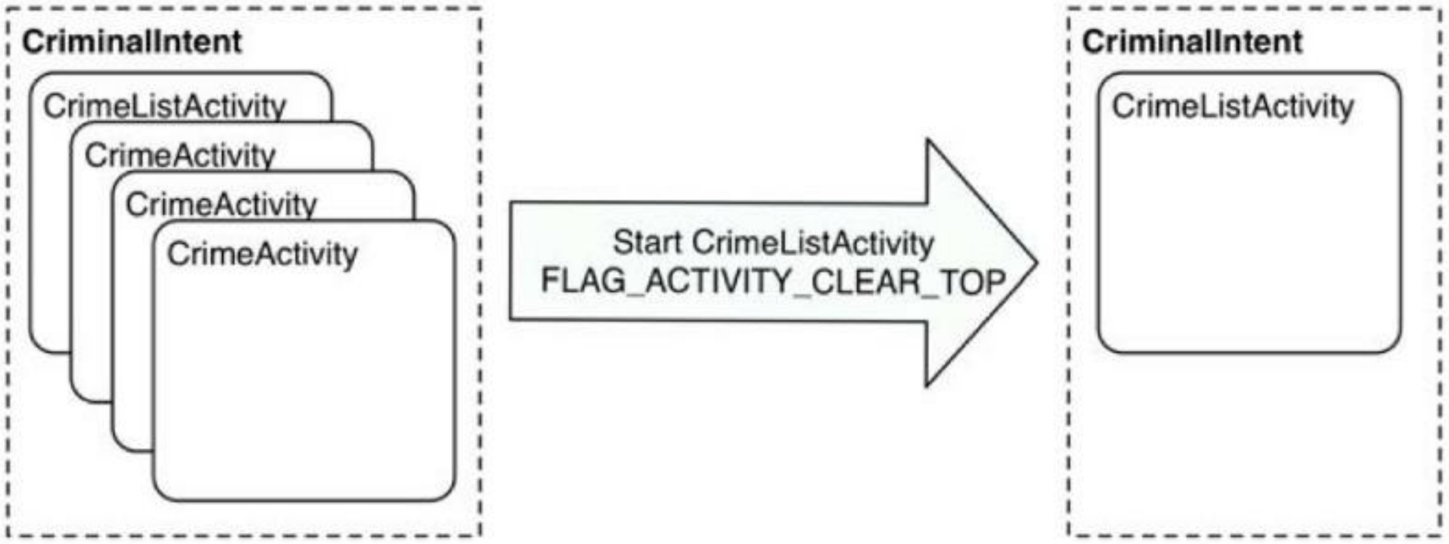
Hierarchical Navigation



```
<activity
    android:name=".CrimePagerActivity"
    android:label="CriminalIntent"
    android:parentActivityName=".CrimeListActivity">
</activity>
```

- Enable hierarchical navigation: add android:parentActivityName attribute in <activity> element in AndroidManifest.xml

- Hierarchical navigation (‘Up’ button) is different from temporal navigation (Back button).


CriminalIntent
CrimeListActivity
CrimeActivity
CrimeActivity
CrimeActivity
Start CrimeListActivity
FLAG_ACTIVITY_CLEAR_TOP
CriminalIntent
CrimeListActivity