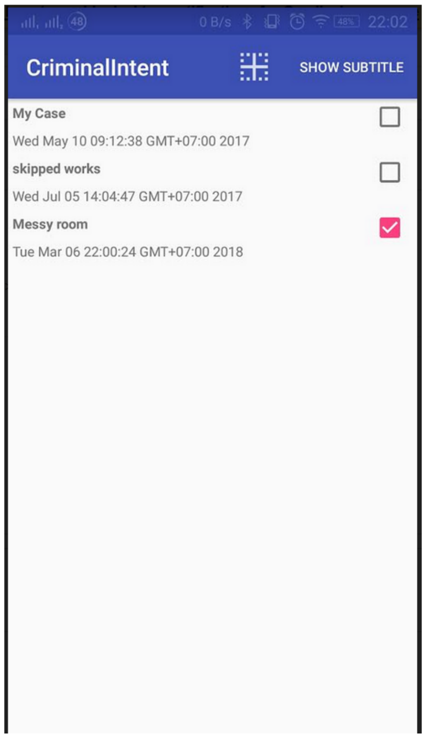
Level/Subject	:	**D3/Programming 6**
Topic	:	UI in Android apps
Week	:	4
Activity	:	Creating Fragments in Android apps using Android Studio
Alocated time	:	60 mins labs
Deliverables	:	Project folder
Due date	:	end of week

Competency :

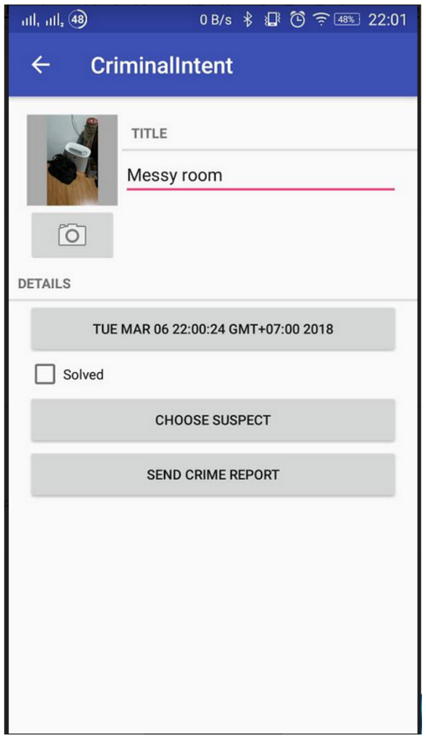
Student expected to be able to create and use Fragments in Android apps using Android Studio IDE.

Example Practice Task:

Create Fragments in Android apps using Android Studio.



My Case
Wed May 10 09:12:38 GMT+07:00 2017
skipped works
Wed Jul 05 14:04:47 GMT+07:00 2017
Messy room
Tue Mar 06 22:00:24 GMT+07:00 2018



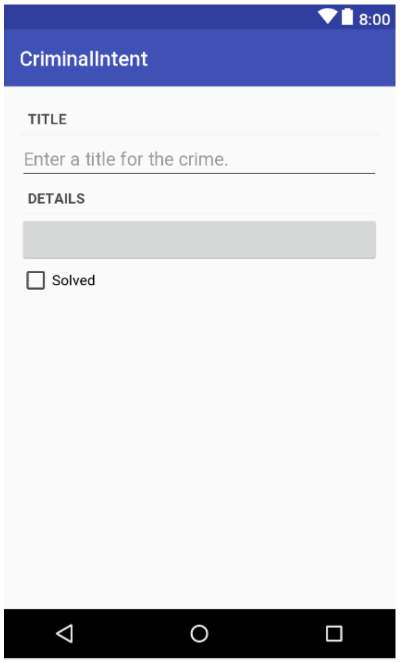
TITLE: Messy room
TUE MAR 06 22:00:24 GMT+07:00 2018
Solved

1. You will start building an application named CriminalIntent. CriminalIntent records the details of "office crimes" – things like leaving dirty dishes in the breakroom sink or walking away from an empty shared printer after documents have printed.
2. With CriminalIntent, you can make a record of a crime including a title, a date, and a photo. You can also identify a suspect from your contacts and lodge a complaint via email, Twitter, Facebook,

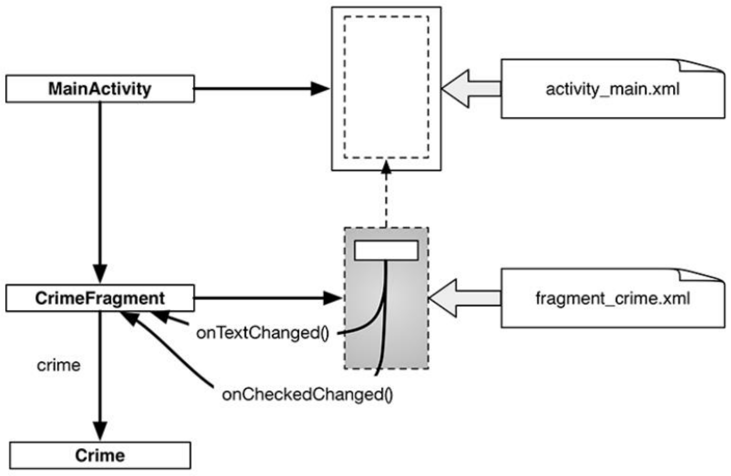
or another app. After documenting and reporting a crime, you can proceed with your work free of resentment and ready to focus on the business at hand.

3. CriminalIntent is a complex app. It will have a list-detail interface: The main screen will display a list of recorded crimes. Users will be able to add new crimes or select an existing crime to view and edit its details.

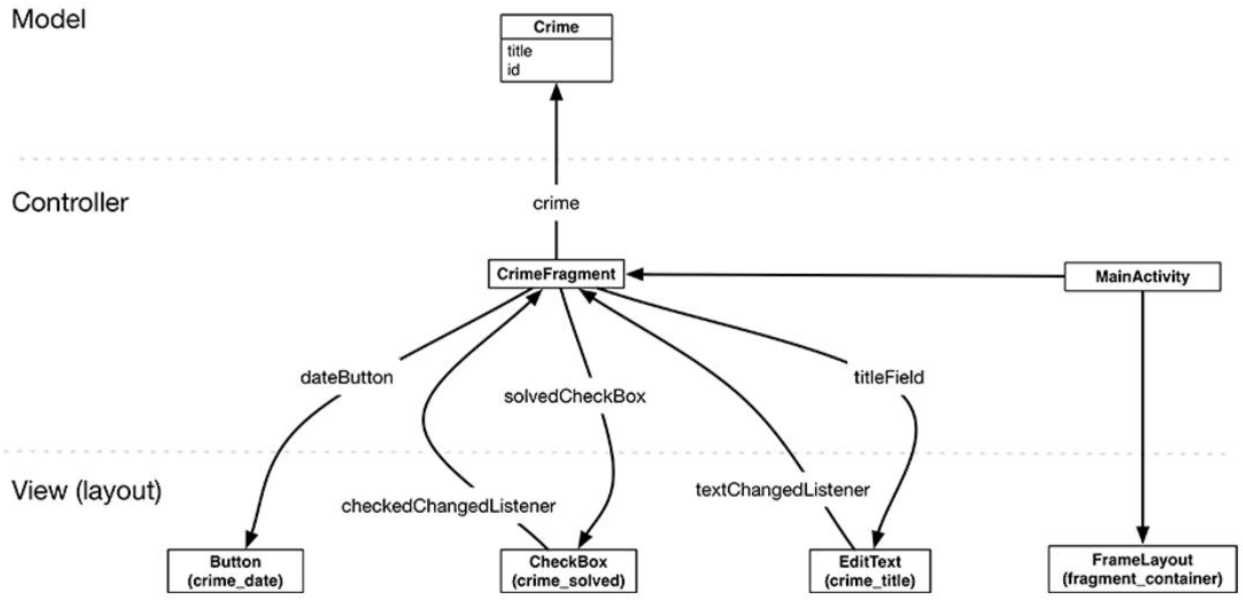
Starting CriminalIntent



4. The screen shown above will be managed by a UI fragment named `CrimeFragment`. An instance of `CrimeFragment` will be `hosted` by an activity named `MainActivity`.
5. For now, think of hosting as the activity providing a spot in its view hierarchy to contain the fragment and its view. A fragment is incapable of getting a view onscreen itself. Only when it is inserted in an activity's hierarchy will its view appear.



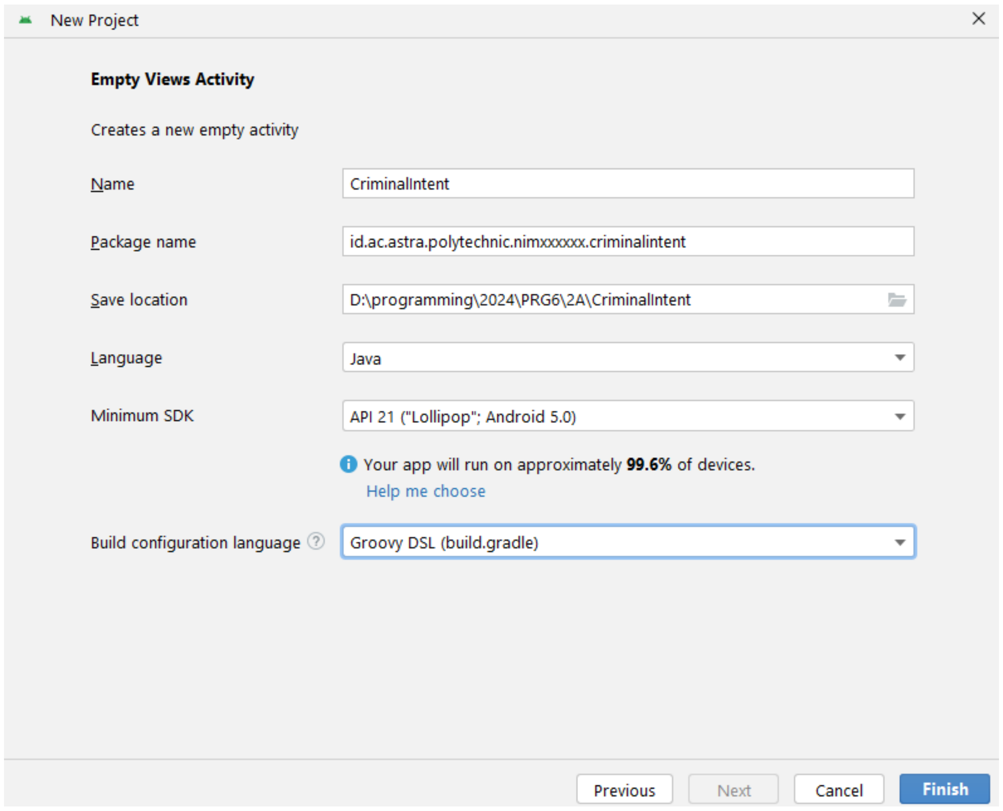
6. CriminalIntent will be a large project, and one way to keep your head wrapped around a project is with an object diagram. Figure below gives you the big picture of CriminalIntent. You do not have to memorize these objects and their relationships, but it is good to have an idea of where you are heading before you start.
7. You can see that `CrimeFragment` will do the sort of work that your activities did in GeoQuiz: create and manage the UI and interact with the model objects.



8. Three of the classes shown are classes that you will write: **Crime**, **CrimeFragment**, and **MainActivity**.
9. An instance of **Crime** will represent a single office crime. In this practice, a crime will have only a title and an ID. The title is a descriptive name, like "Toxic sink dump" or "Someone stole my yogurt!" The ID will uniquely identify an instance of **Crime**.
10. For this practice, you will keep things very simple and use a single instance of **Crime**.
11. **CrimeFragment** will have a member variable (**mCrime**) to hold this isolated incident.
12. **MainActivity**'s view will consist of a **FrameLayout** that defines the spot where the **CrimeFragment**'s view will appear.
13. **CrimeFragment**'s view will consist of a **LinearLayout** with a few child views inside of it, including an **EditText**, a **Button**, and a **CheckBox**. **CrimeFragment** will have member variables for each of these views and will set listeners on them to update the model layer when there are changes.

Creating a new project

14. Create a new Android application (File → New Project...).
15. Choose Empty Views Activity
16. Name the application CriminalIntent and name the package id.ac.polytechnic.astra.nimXXXXXX.criminalintent (where nimXXXXXX is your student ID) and specify a minimum SDK of API 21



Empty Views Activity

Creates a new empty activity

Name: CriminalIntent

Package name: id.ac.astra.polytechnic.nimxxxxxx.criminalintent

Save location: D:\programming\2024\PRG6\2A\CriminalIntent

Language: Java

Minimum SDK: API 21 ("Lollipop"; Android 5.0)

Your app will run on approximately **99.6%** of devices.
Help me choose

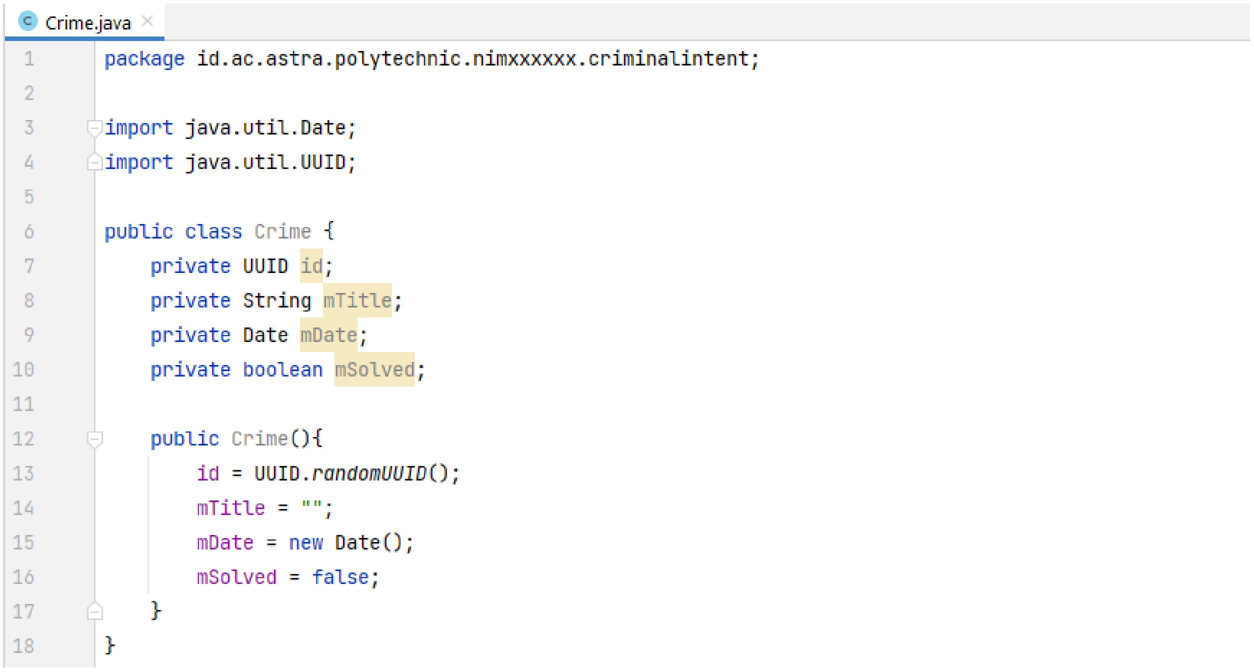
Build configuration language: Groovy DSL (build.gradle)

17. Click Finish

Creating the Crime class

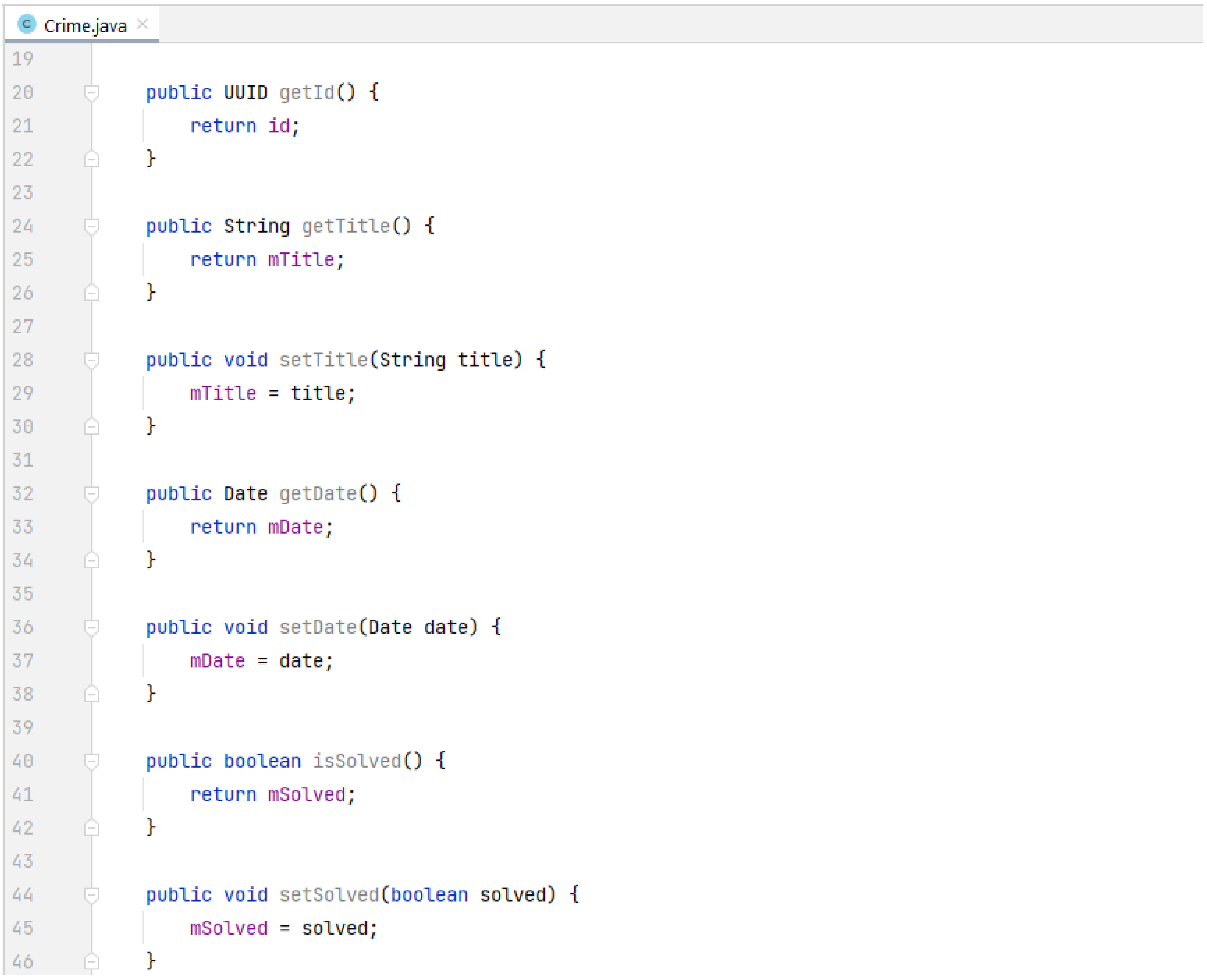
18. In the project tool window, right-click the `id.ac.astra.polytechnic.nimxxxxxxx.criminalintent` package and select New → Java Class.
19. Name the class **Crime** and click OK.

20. In `Crime.java`, add the following code:



```java
package id.ac.astra.polytechnic.nimxxxxxx.criminalintent;

import java.util.Date;
import java.util.UUID;

public class Crime {
    private UUID id;
    private String mTitle;
    private Date mDate;
    private boolean mSolved;

    public Crime(){
        id = UUID.randomUUID();
        mTitle = "";
        mDate = new Date();
        mSolved = false;
    }
}
```

21. Next, you want to generate only a getter for the read-only **id** and both a getter and setter for everything else.
22. Right-click after the constructor and select Generate... → Getter and select the **id** variable.
23. Then, generate the getter and setter for mTitle, mDate, and mSolved by repeating the process, but selecting Getter and Setter in the Generate... menu.



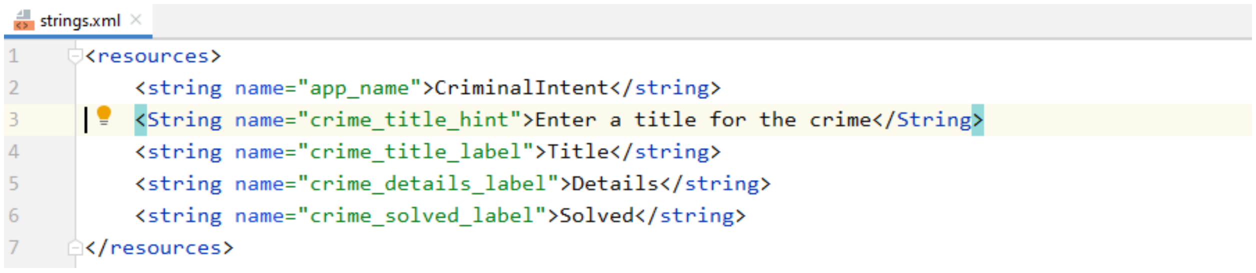
```java

    public UUID getId() {
        return id;
    }

    public String getTitle() {
        return mTitle;
    }

    public void setTitle(String title) {
        mTitle = title;
    }

    public Date getDate() {
        return mDate;
    }

    public void setDate(Date date) {
        mDate = date;
    }

    public boolean isSolved() {
        return mSolved;
    }

    public void setSolved(boolean solved) {
        mSolved = solved;
    }
```

24. That is all you need for the **Crime** class and for CriminalIntent’s model layer in this practice.
25. At this point, you have created the model layer and an activity that is capable of hosting a support fragment. Now you will get into the details of how the activity performs its duties as host.

Creating a UI Fragment : Defining CrimeFragment’s layout

26. **CrimeFragment**’s view will display the information contained within an instance of **Crime**. Eventually, the **Crime** class and **CrimeFragment**’s view will include many interesting pieces, but for this practice you just need a text field to contain the crime’s title.

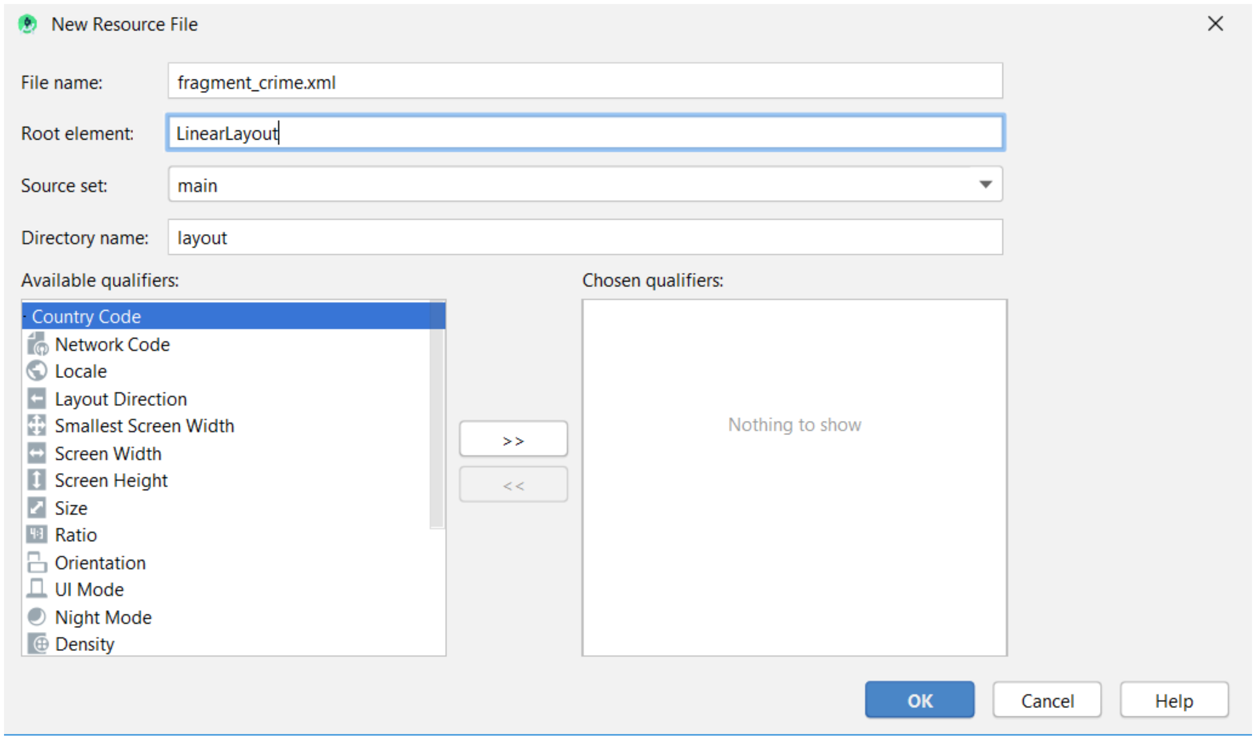
27. Open res/values/strings.xml and add string resource.



strings.xml

```
<resources>
    <string name="app_name">CriminalIntent</string>
    <String name="crime_title_hint">Enter a title for the crime</String>
    <string name="crime_title_label">Title</string>
    <string name="crime_details_label">Details</string>
    <string name="crime_solved_label">Solved</string>
</resources>
```

28. The layout for `CrimeFragment` will consist of a vertical `LinearLayout` that contains two `TextViews`, an `EditText`, a `Button`, and a `CheckBox`.
29. To create a layout file, right-click the `res/layout` folder in the project tool window and select New → Layout resource file.
30. Name this file `fragment_crime.xml` and enter **LinearLayout** as the root element.
31. Click OK and Android Studio will generate the file for you.



New Resource File

File name: fragment_crime.xml

Root element: LinearLayout

Source set: main

Directory name: layout

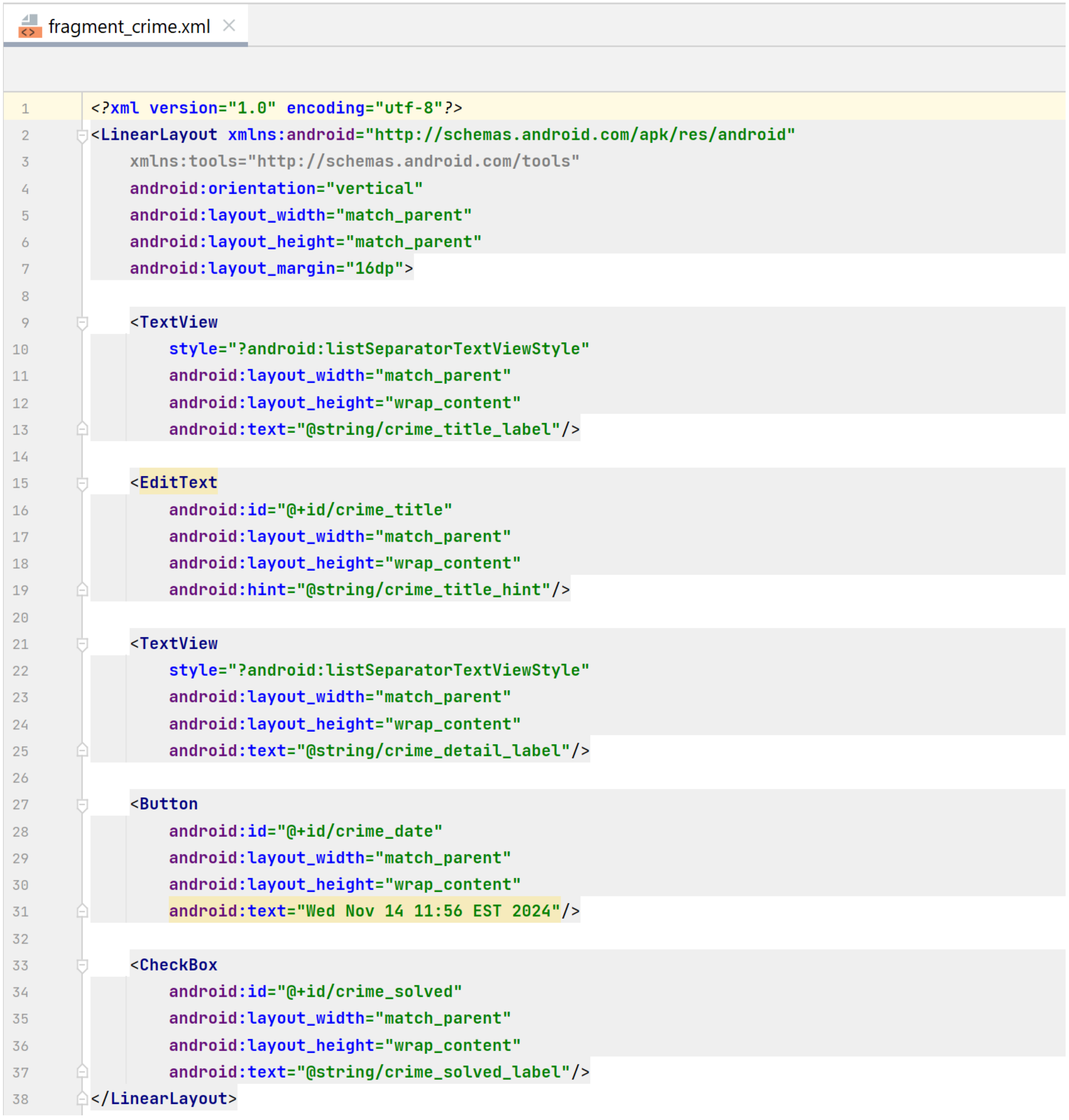
Available qualifiers:

Country Code
Network Code
Locale
Layout Direction
Smallest Screen Width
Screen Width
Screen Height
Size
Ratio
Orientation
UI Mode
Night Mode
Density

Chosen qualifiers:

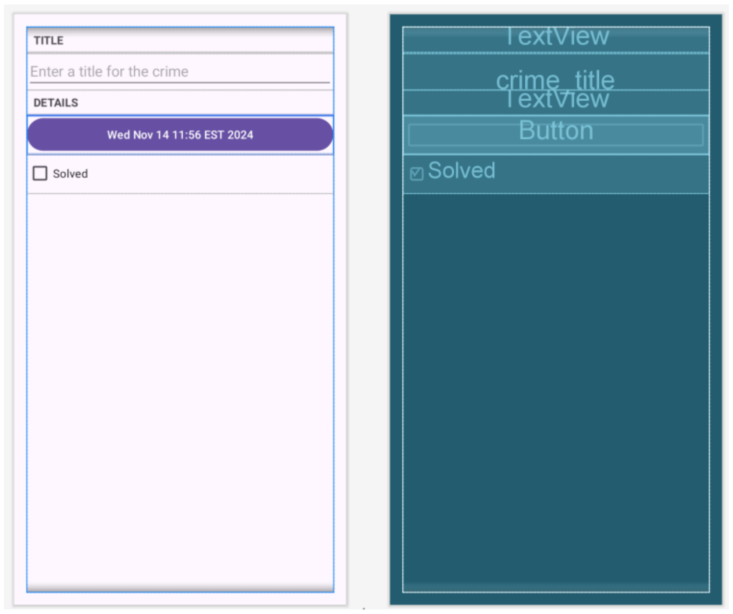
Nothing to show

32. When the file opens, navigate to the XML. The wizard has added the **LinearLayout** for you. Using Figure below as a guide, make the necessary changes to `fragment_crime.xml`.


fragment_crime.xml

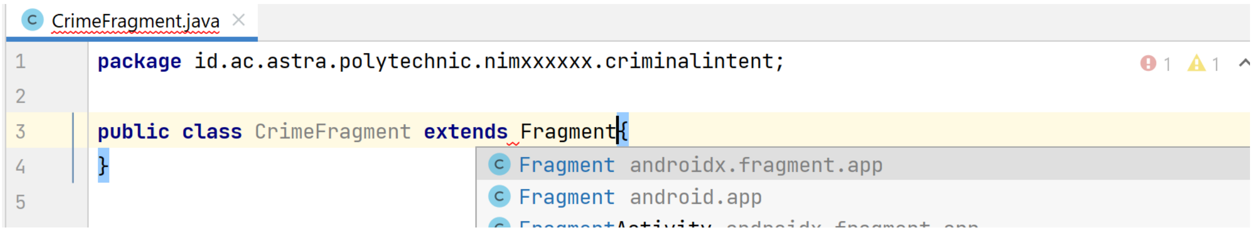
```xml
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:orientation="vertical"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:layout_margin="16dp">

    <TextView
        style="?android:listSeparatorTextViewStyle"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="@string/crime_title_label"/>

    <EditText
        android:id="@+id/crime_title"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:hint="@string/crime_title_hint"/>

    <TextView
        style="?android:listSeparatorTextViewStyle"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="@string/crime_detail_label"/>

    <Button
        android:id="@+id/crime_date"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Wed Nov 14 11:56 EST 2024"/>

    <CheckBox
        android:id="@+id/crime_solved"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="@string/crime_solved_label"/>
</LinearLayout>
```

33. Save your files. Navigate back to `fragment_crime.xml` to see a preview of your fragment's view.



Creating the CrimeFragment class

34. Right-click the id.ac.astra.polytechnic.nimxxxxxxx.criminalintent package and select New → Java Class.
35. Name the class **CrimeFragment** and click OK to generate the class.
36. Now, turn this class into a fragment. Update **CrimeFragment** to subclass the **Fragment** class.

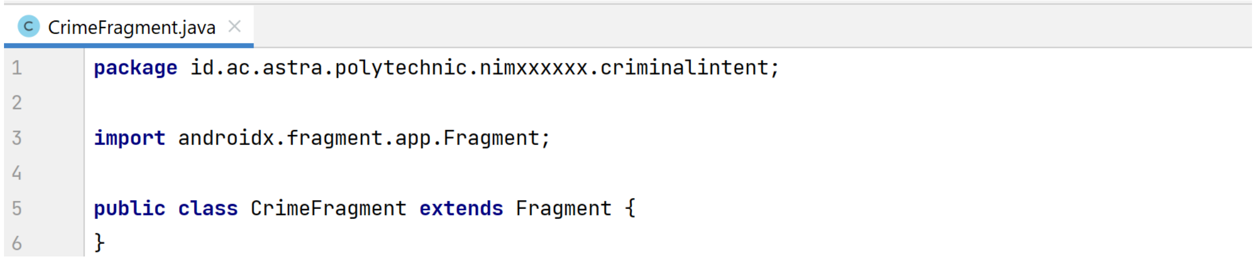


```
package id.ac.astra.polytechnic.nimxxxxxx.criminalintent;

public class CrimeFragment extends Fragment{
}
```

Fragment androidx.fragment.app
Fragment android.app
FragmentActivity androidx.fragment.app

37. As you subclass the **Fragment** class, you will notice that Android Studio finds two classes with the **Fragment** name. You will see **Fragment (android.app)** and **Fragment (androidx.fragment.app)**. The android.app **Fragment** is the version of fragments that are built into the Android OS. We will use the jetpack library version, so be sure to select the androidx.fragment.app version of the **Fragment** class when you see the dialog

38. Your code should be like this:

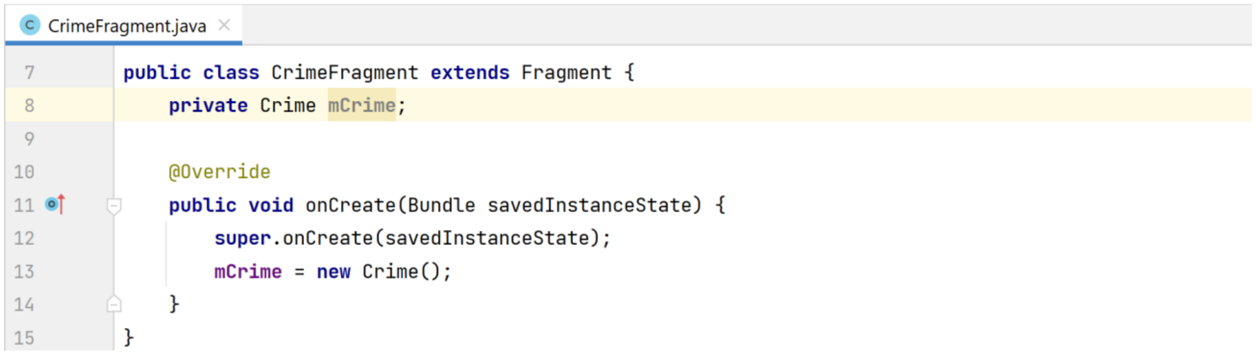


CrimeFragment.java

```java
package id.ac.astra.polytechnic.nimxxxxxx.criminalintent;

import androidx.fragment.app.Fragment;

public class CrimeFragment extends Fragment {
}
```

Implementing fragment lifecycle methods

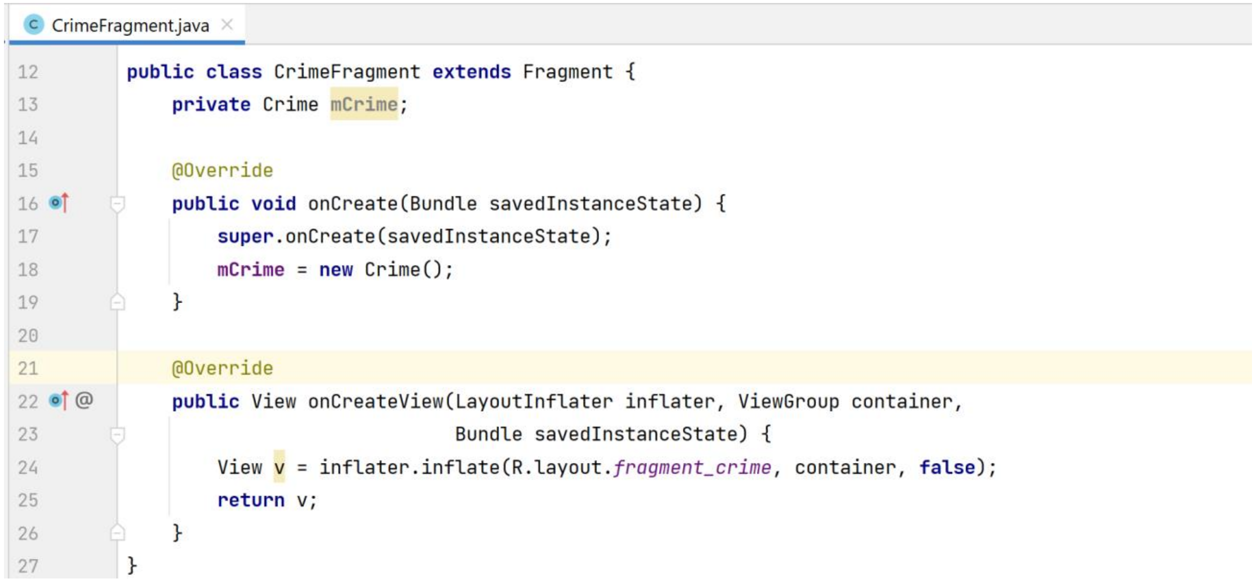
39. **CrimeFragment** is a controller that interacts with model and view objects. Its job is to present the details of a specific crime and update those details as the user changes them.
40. In GeoQuiz, your activities did most of their controller work in activity lifecycle methods. In CriminalIntent this work will be done by fragments in fragment lifecycle methods. Many of these methods correspond to the **Activity** methods you already know, such as **onCreate(Bundle)**.
41. In CrimeFragment.java, add a member variable for the **Crime** instance and an implementation of **Fragment.onCreate(Bundle)**.



CrimeFragment.java

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

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

42. In `CrimeFragment.java`, add an implementation of **onCreateView(…)** that inflates `fragment_crime.xml`.

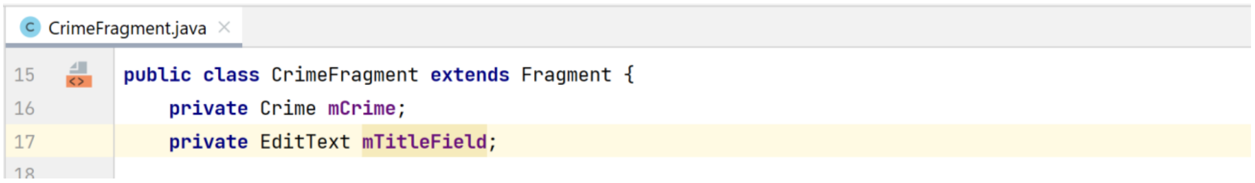


CrimeFragment.java

```java
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;
    }
}
```

Wiring widgets in a fragment

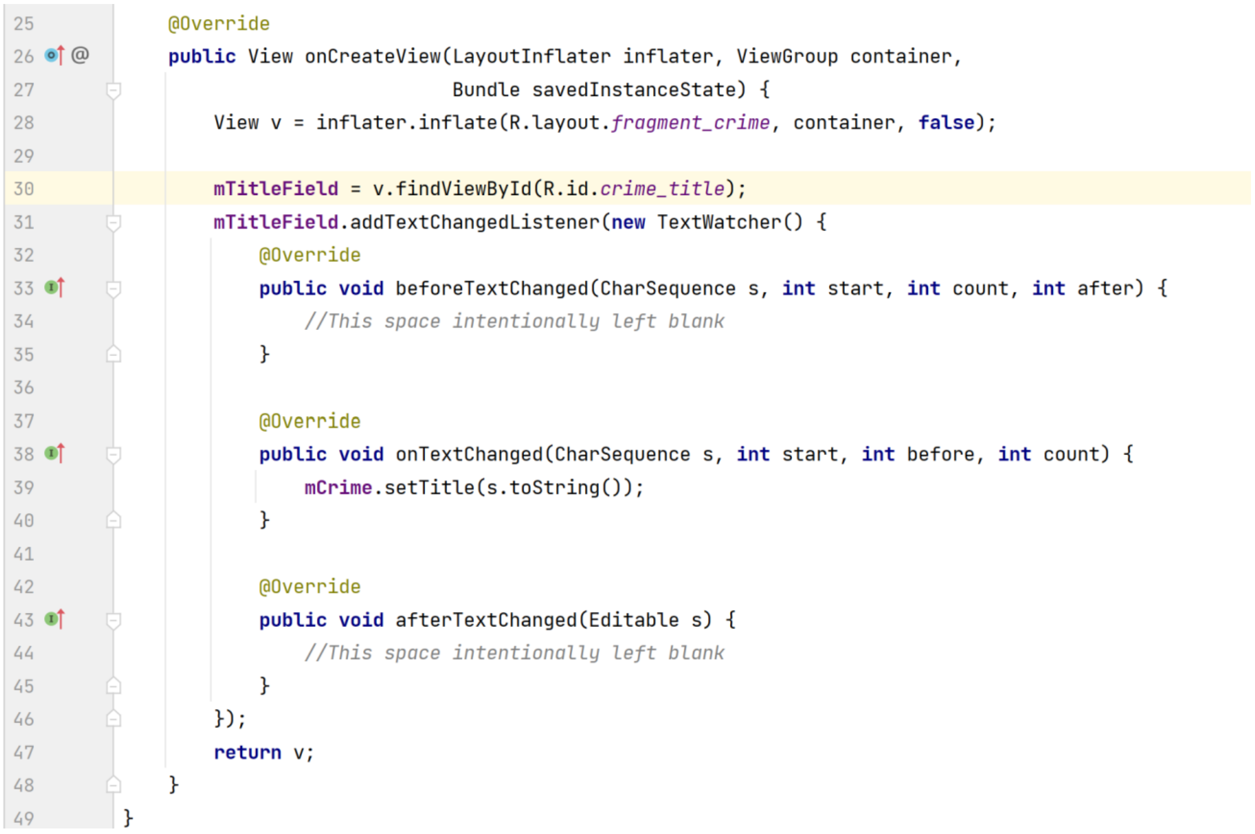
43. The **onCreateView(…)** method is also the place to wire up the **EditText** to respond to user input. After the view is inflated, get a reference to the **EditText** and add a listener.



CrimeFragment.java

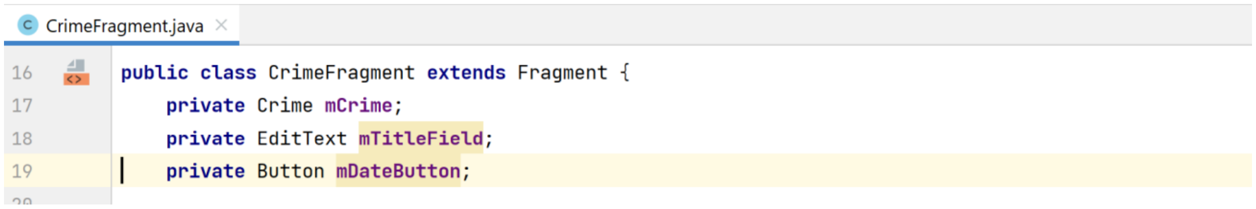
```java
public class CrimeFragment extends Fragment {
    private Crime mCrime;
    private EditText mTitleField;
```

…



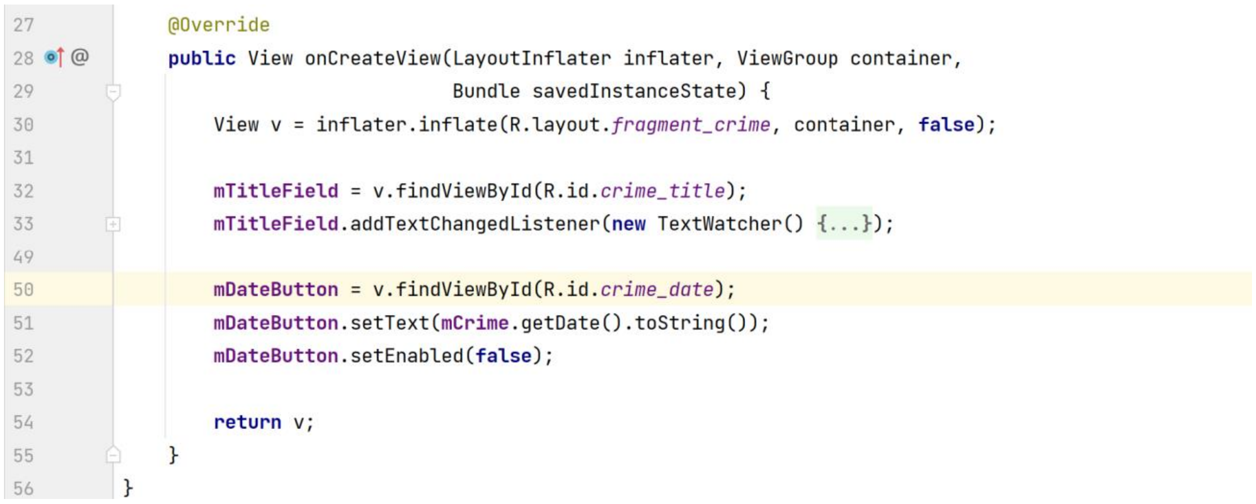
```java
    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container,
                             Bundle savedInstanceState) {
        View v = inflater.inflate(R.layout.fragment_crime, container, false);

        mTitleField = 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 space intentionally left blank
            }
        });
        return v;
    }
}
```

44. Next, connect the **Button** to display the date of the crime


CrimeFragment.java

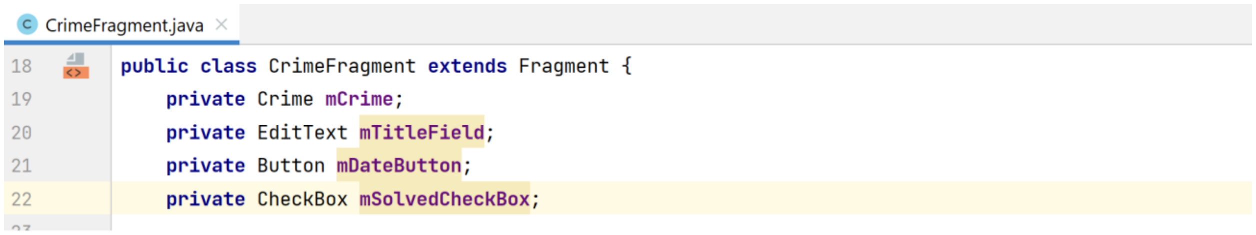
```
public class CrimeFragment extends Fragment {
    private Crime mCrime;
    private EditText mTitleField;
    private Button mDateButton;
```

...



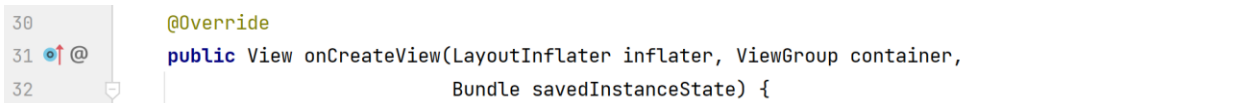
```
    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container,
                             Bundle savedInstanceState) {
        View v = inflater.inflate(R.layout.fragment_crime, container, false);

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

        mDateButton = v.findViewById(R.id.crime_date);
        mDateButton.setText(mCrime.getDate().toString());
        mDateButton.setEnabled(false);

        return v;
    }
}
```

45. Moving on to the **CheckBox**, get a reference and set a listener that will update the mSolved field of the **Crime**,



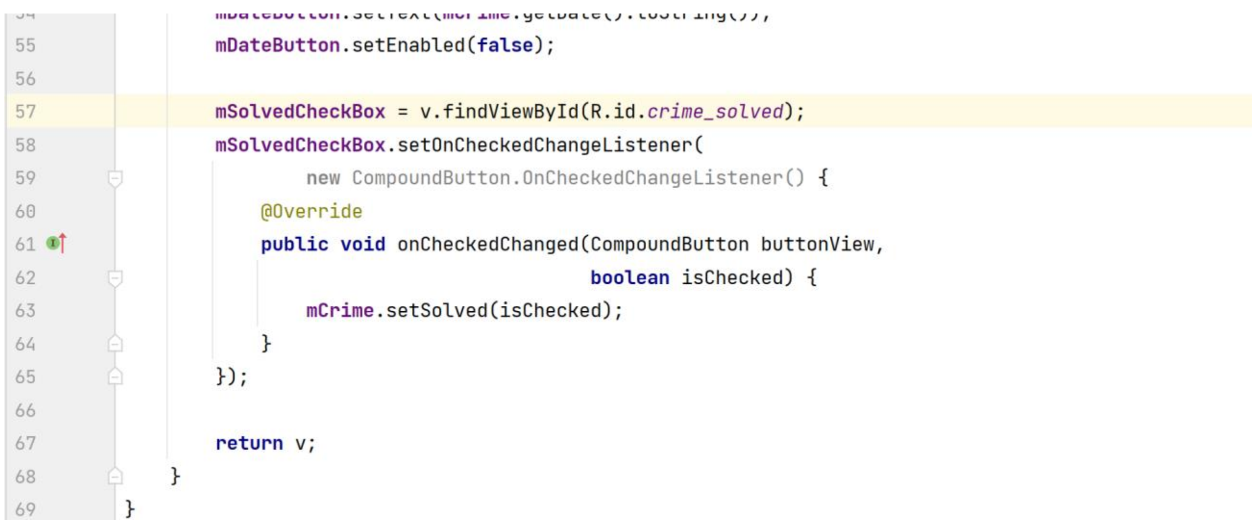
```java
public class CrimeFragment extends Fragment {
    private Crime mCrime;
    private EditText mTitleField;
    private Button mDateButton;
    private CheckBox mSolvedCheckBox;
```

...



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

…



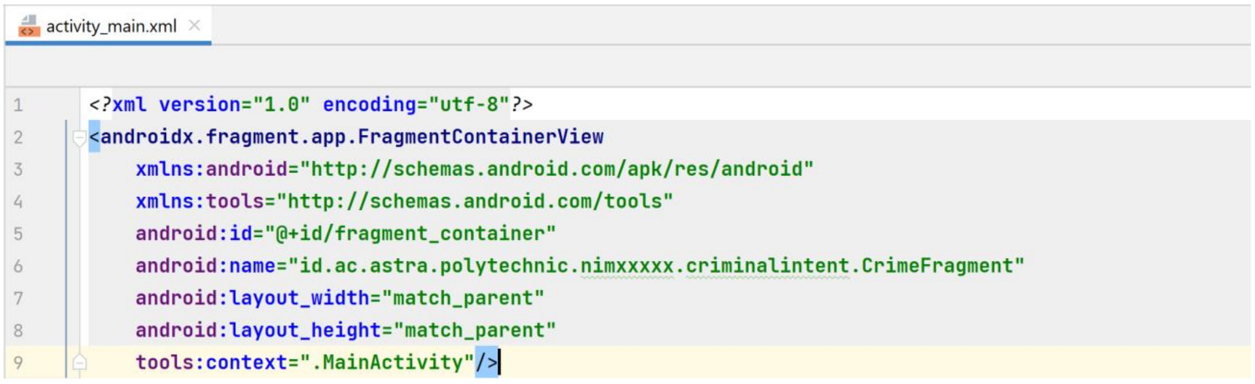
```java
        mDateButton.setText(mCrime.getDate().toString());
        mDateButton.setEnabled(false);

        mSolvedCheckBox = v.findViewById(R.id.crime_solved);
        mSolvedCheckBox.setOnCheckedChangeListener(
                new CompoundButton.OnCheckedChangeListener() {
            @Override
            public void onCheckedChanged(CompoundButton buttonView,
                                         boolean isChecked) {
                mCrime.setSolved(isChecked);
            }
        });

        return v;
    }
}
```

46. Your code for **CrimeFragment** is now complete. It would be great if you could run CriminalIntent now and play with the code you have written. But you cannot. **<u>Fragments cannot put their views on screen</u>**. To realize your efforts, you first have to add a **CrimeFragment** to **MainActivity**.

Hosting a UI Fragment, Defining a container view

47. When fragments were first introduced, developers had to jump through numerous hoops to display them. In 2019, Google introduced the FragmentContainerView, which makes it easier to create host containers for a fragment. In this section, you will use a FragmentContainerView to host your CrimeFragment. Then you will learn about the FragmentManager and the fragment lifecycle. Finally, you will tie up one loose end in CrimeFragment.
48. FragmentContainerView is, as its name suggests, built to contain fragments. Fragments have changed significantly over the years, so FragmentContainerView helps provide a consistent environment for fragments to operate in. Much like the views you have used so far, the FragmentContainerView has common XML attributes to define its ID and its size.

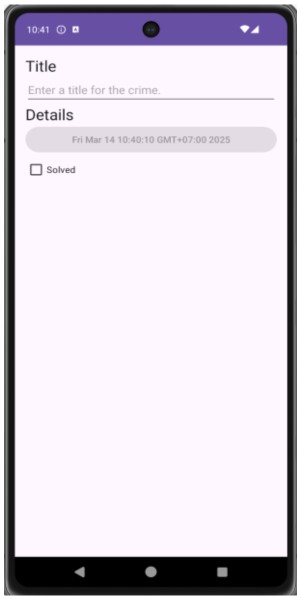
49. Locate and open MainActivity's layout in res/layout/activity_main.xml. Replace the default layout with a FragmentContainerView, as shown



activity_main.xml

```xml
<?xml version="1.0" encoding="utf-8"?>
<androidx.fragment.app.FragmentContainerView
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/fragment_container"
    android:name="id.ac.astra.polytechnic.nimxxxxx.criminalintent.CrimeFragment"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity"/>
```

50. FragmentContainerView has one XML attribute that you have not seen on other views: android:name, whose value here is the full package name for CrimeFragment. With that, the FragmentContainerView will manage creating your CrimeFragment and inserting it in the activity's layout.
51. At last, it is time to run CriminalIntent to check your code.



10:41

Title

Enter a title for the crime.

Details

Fri Mar 14 10:40:10 GMT+07:00 2025

Solved

Notes:

1. Create folder PRG6_M4_P1_XXXXXXXXXX.
2. Zip the folder and submit it to the server.

Bibliography: Marsicano, et. al., "Android Programming – The Big Nerd Ranch", 5th Ed, 2022, Pearson Technology.