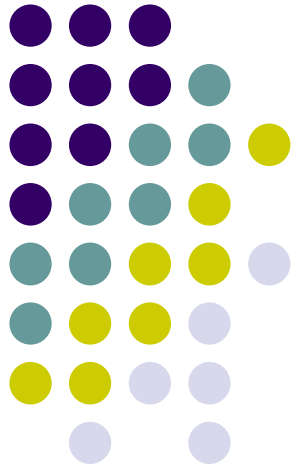
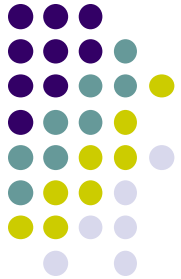
CS 528 Mobile and Ubiquitous Computing

Lecture 4a: Fragments, Database and Firebase Cloud API

Emmanuel Agu

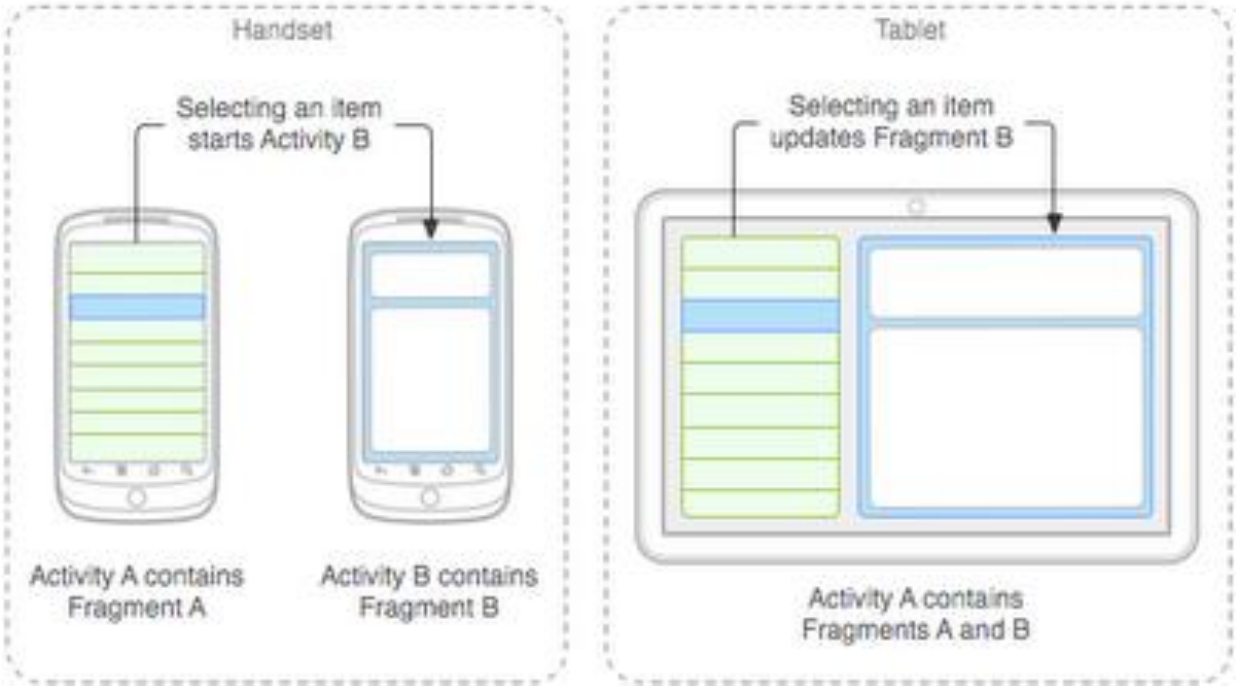




Fragments

Recall: Fragments

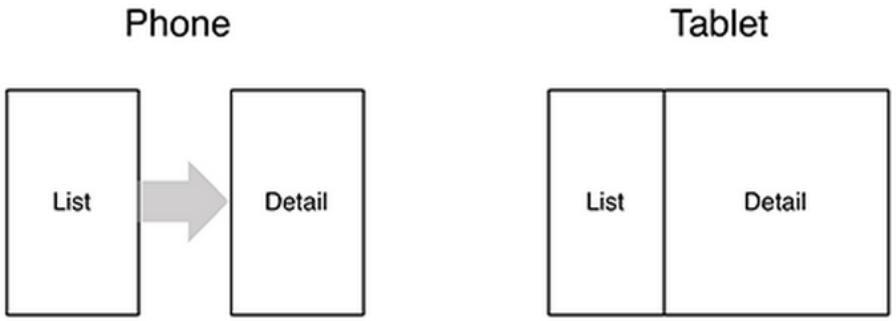
- Sub-components of an Activity (screen)
- An activity can contain multiple fragments, organized differently on different devices (e.g. phone vs tablet)
- Fragments need to be attached to Activities.

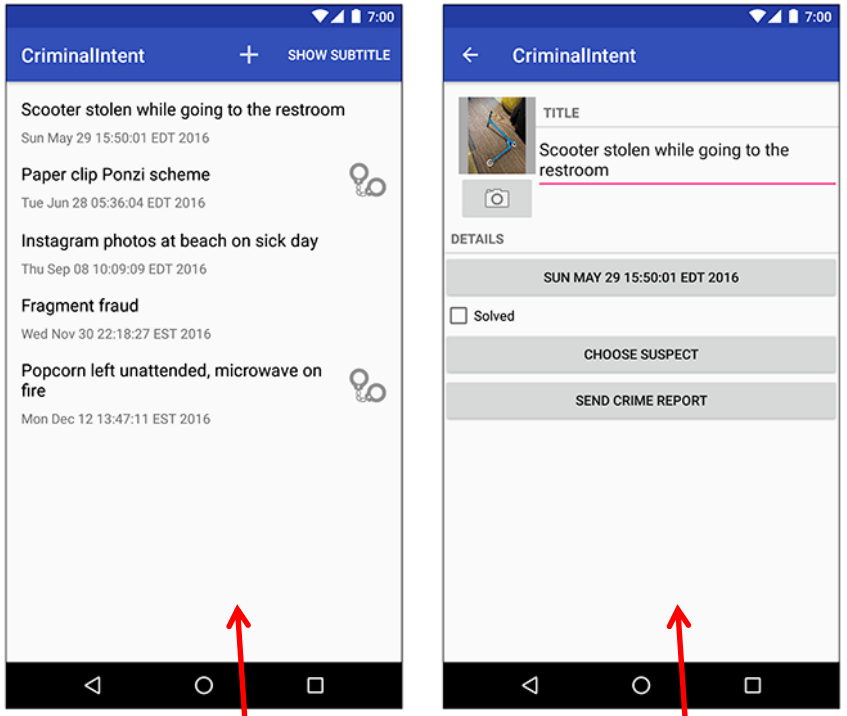


Fragments

Ref: Android Nerd Ranch (3rd ed), Ch 7, pg 123

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





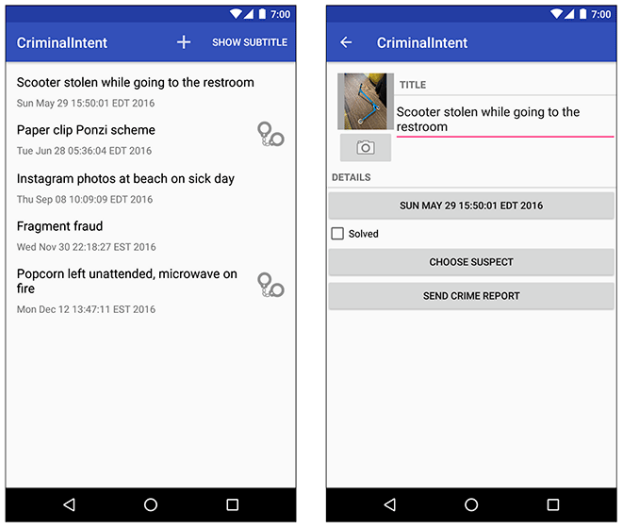
- **On tablet:** show list + detail
- **On phone:** swipe to show next crime

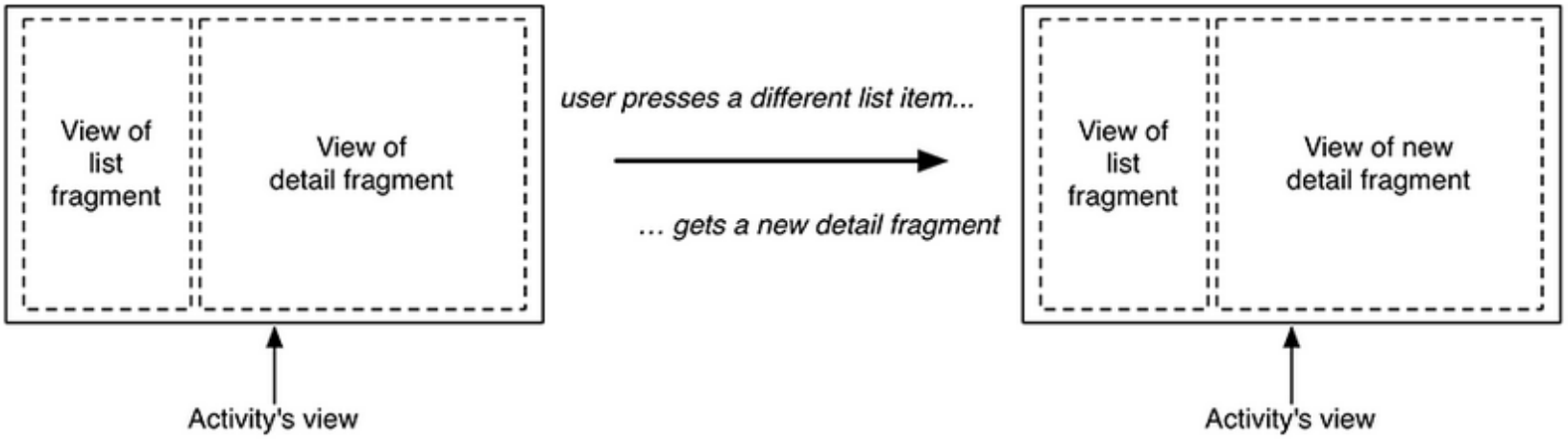
Fragment 1
(list of Crimes)

Fragment 2
(Details of selected Crime)

Fragments

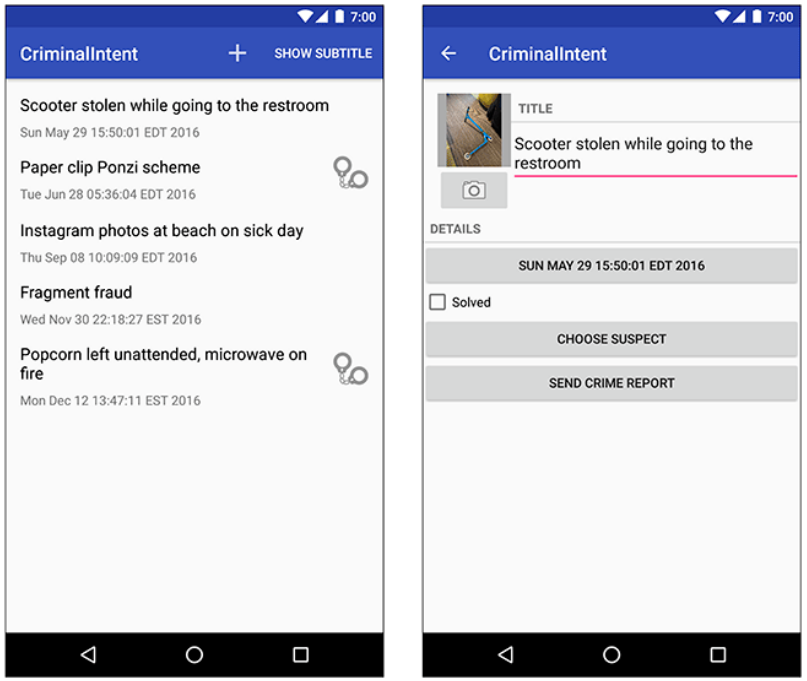
- Activities can contain multiple fragments
- Fragment's views are inflated from a layout file
- Can rearrange fragments as desired on an activity
  - i.e. different arrangement on phone vs tablet



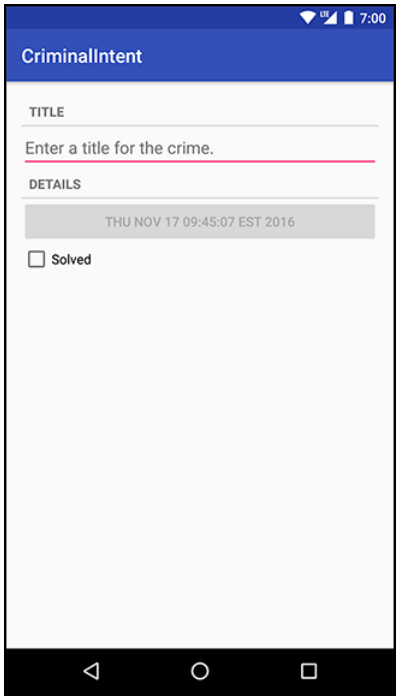


Starting Criminal Intent

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



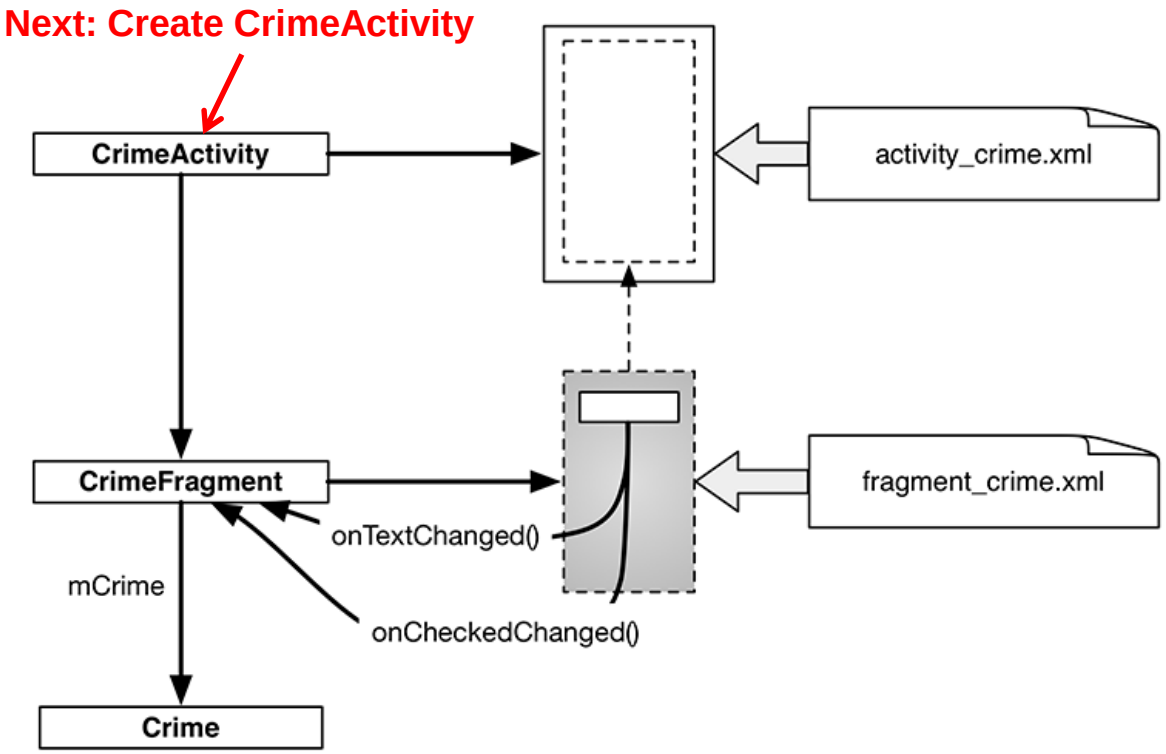
Final Look of CriminalIntent

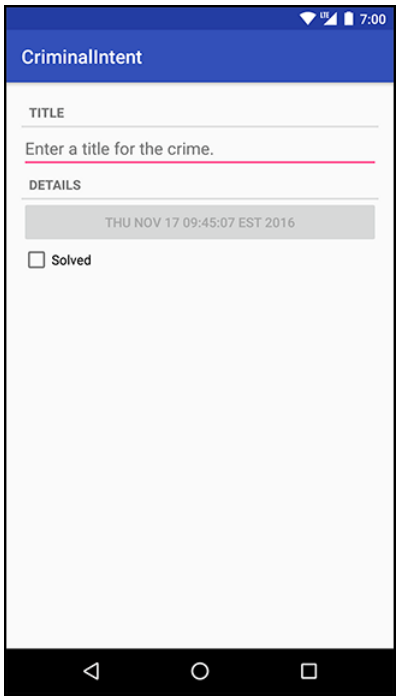


Start small
Develop detail view using Fragments

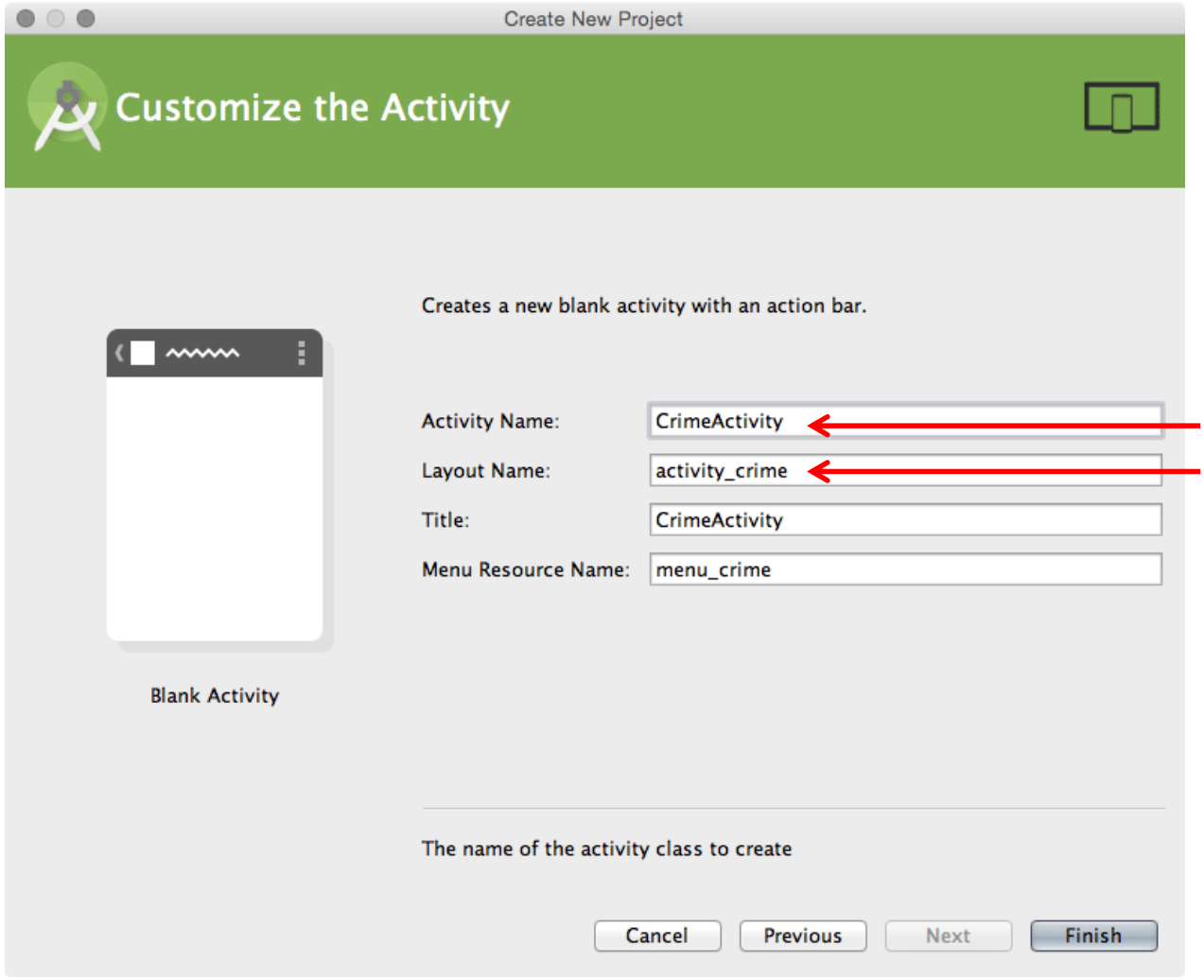
Starting Criminal Intent

- **Crime:** holds record of 1 office crime. Has
  - **Title** e.g. “Someone stole my yogurt!”
  - **ID:** unique identifier of crime
- **CrimeFragment:** UI fragment to display Crime Details
- **CrimeActivity:** Activity that contains **CrimeFragment**



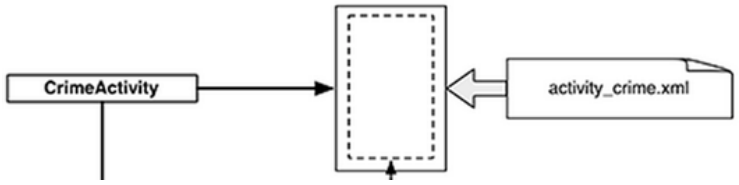


Create CrimeActivity in Android Studio

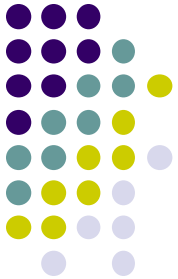


Creates CrimeActivity.java

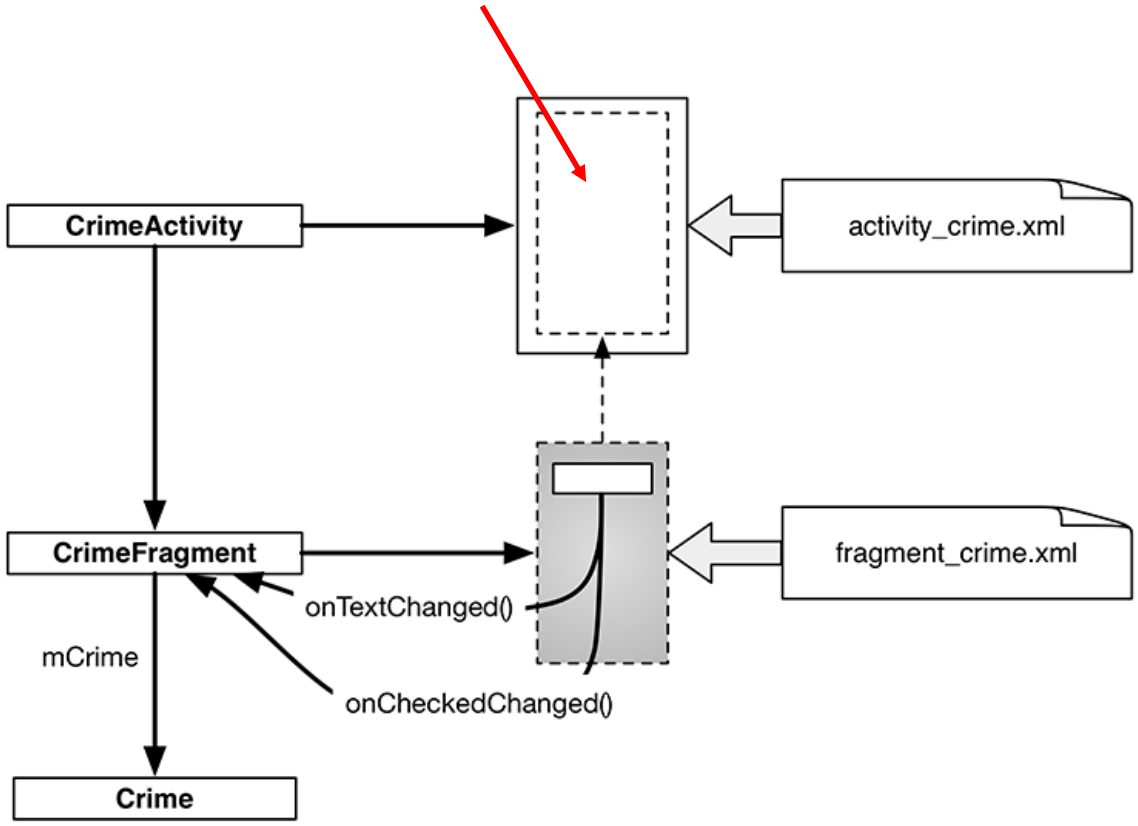
Formatted using activity_crime.xml



Fragment Hosted by an Activity



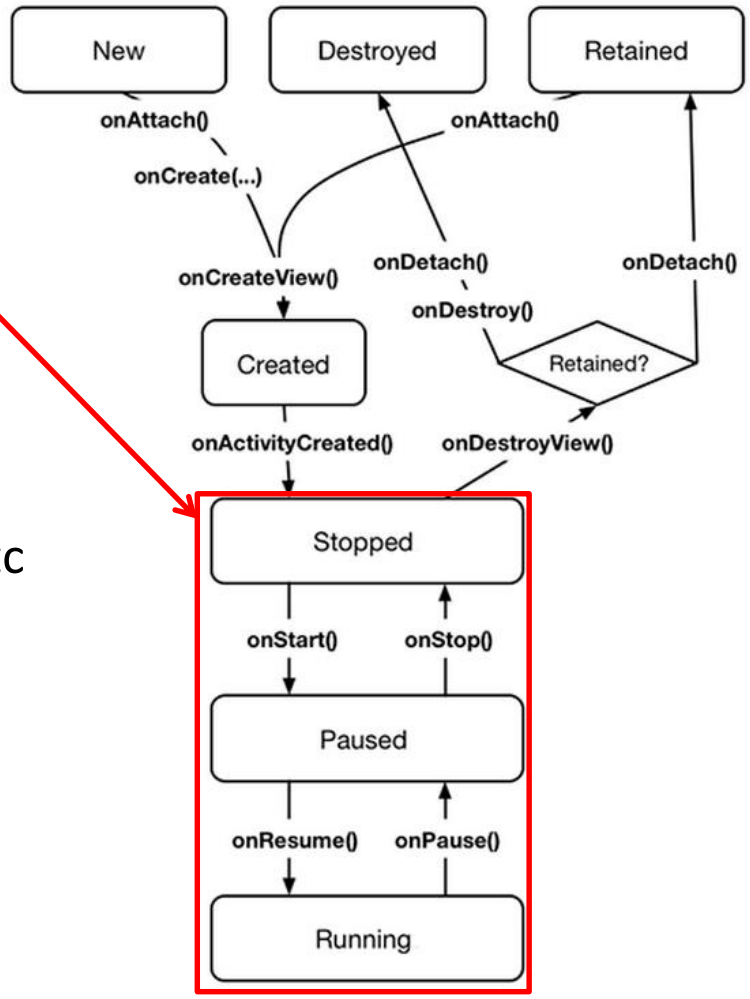
- Each fragment must be hosted by an Activity
- To host a UI fragment, an activity must
  - Define a spot in its layout for the fragment
  - Manage the lifecycle of the fragment instance (next)
- E.g.: **CrimeActivity** defines "spot" for **CrimeFragment**



Fragment's Life Cycle

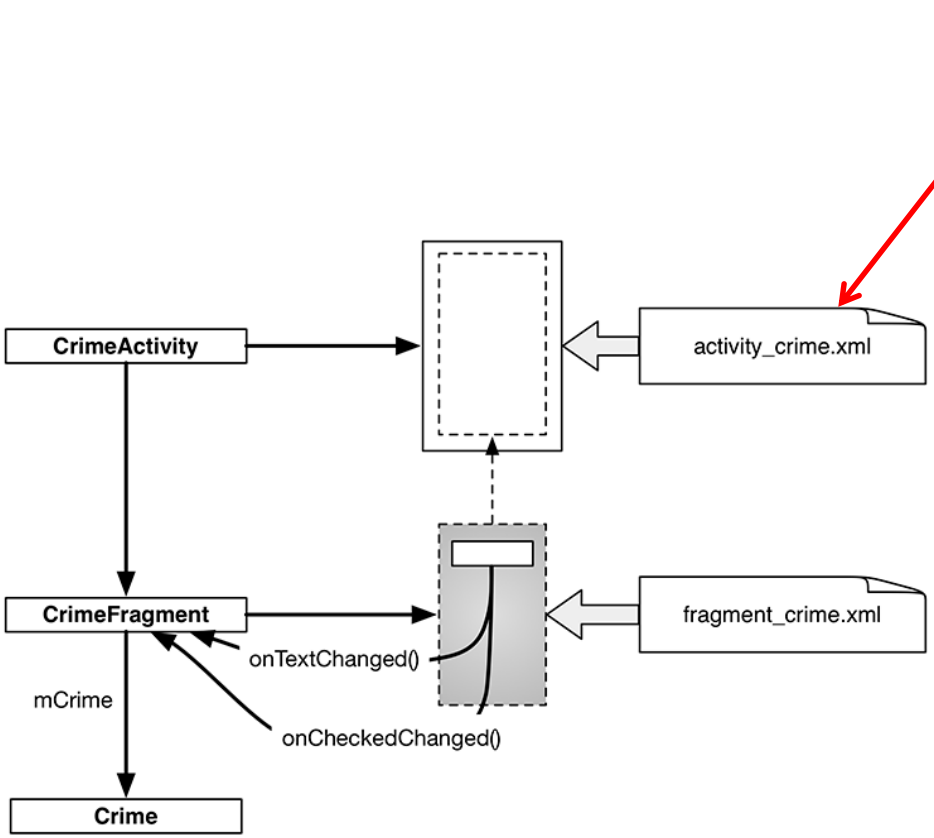
- Fragment's lifecycle similar to activity lifecycle
  - Has states **running**, **paused** and **stopped**
  - Also has some similar activity lifecycle methods (e.g. **onPause()**, **onStop( )**, etc)

- **Key difference:**
  - Android OS calls Activity's onCreate, onPause( ), etc
  - Fragment's **onCreateView( )**, onPause( ), etc **called by hosting activity NOT Android OS!**
  - E.g. Fragment has **onCreateView**



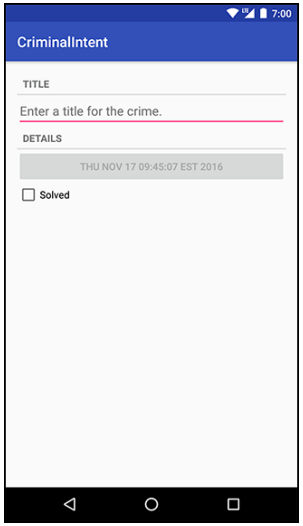
Hosting UI Fragment in an Activity

- 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 XML file now
- First, create a spot for the fragment's view in **CrimeActivity's** XML layout



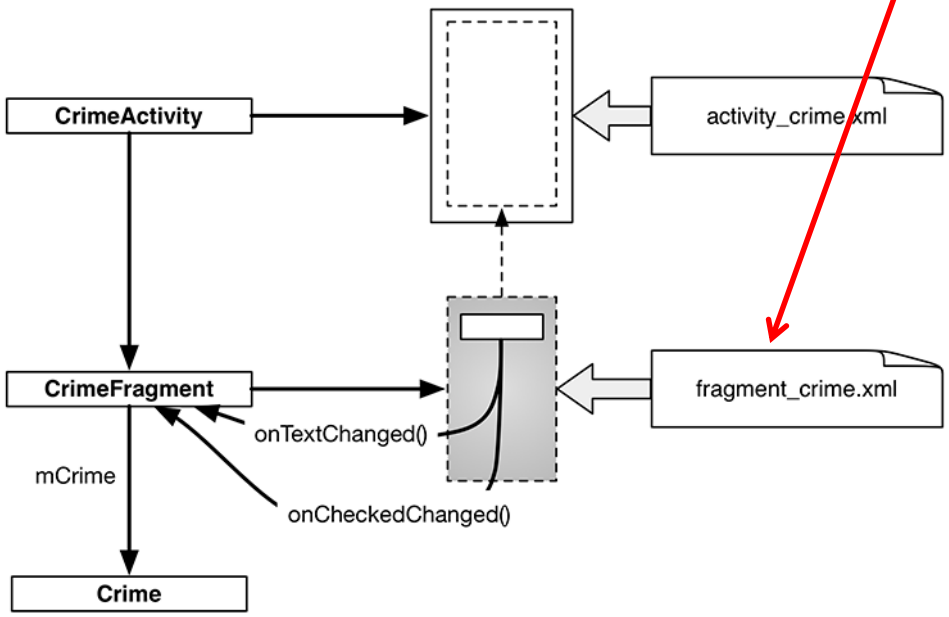
FrameLayout

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

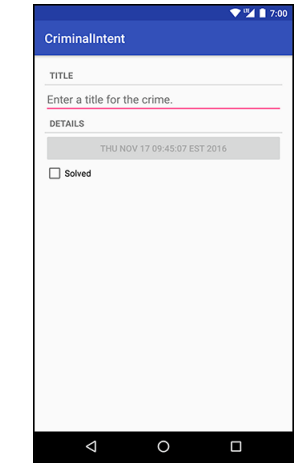


Creating a UI Fragment

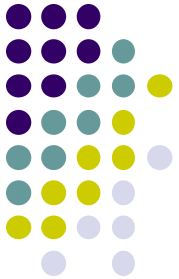
- Creating Fragment is similar to creating activity
  1. Define widgets in a layout (XML) file
  2. Create java class and specify layout file as XML file above
  3. Get references of inflated widgets in java file (findviewbyId), etc
- XML layout file for **CrimeFragment (fragment_crime.xml)**




```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:layout_margin="16dp"
    android:orientation="vertical">

    <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_details_label"/>

    <Button
        android:id="@+id/crime_date"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"/>

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

</LinearLayout>
```



Java File for CrimeFragment



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

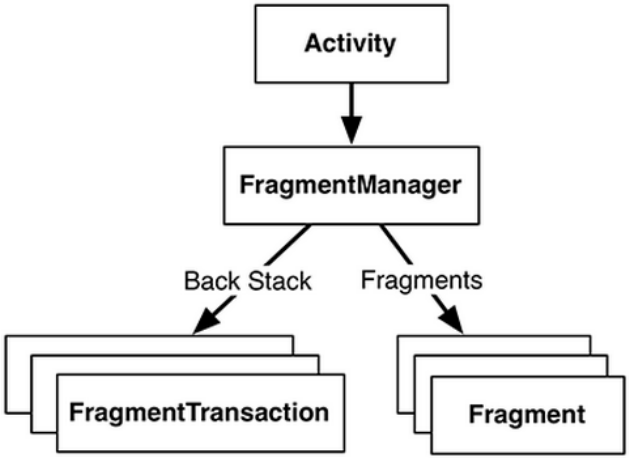
```
public class CrimeFragment extends Fragment {
    private Crime mCrime;

    @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;
    }
}
```

Format Fragment using fragment_crime.xml

- **Note:** Fragment's view inflated in **Fragment.onCreateView()**, NOT **onCreate**

Adding UI Fragment to FragmentManager

- An activity adds new fragment to activity using **FragmentManager**
- **FragmentManager**
  - Manages fragments
  - Adds fragment's views to activity's view
  - Handles
    - List of fragments
    - Back stack of fragment transactions



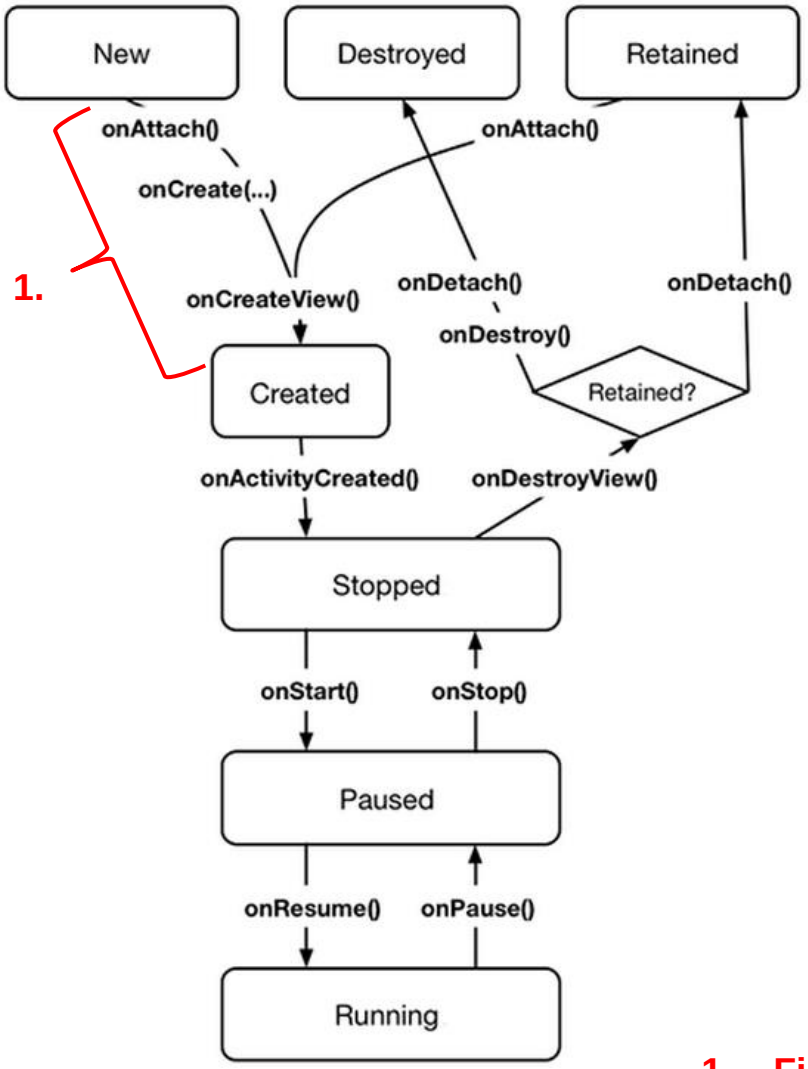

```
public class CrimeActivity extends AppCompatActivity {

    @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

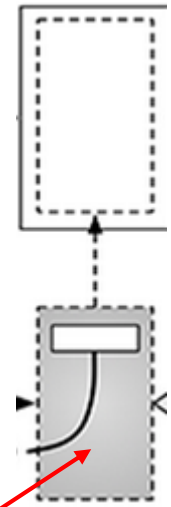
Interactions with FragmentManager are done using transactions

Add Fragment to activity's view

Examining Fragment's Lifecycle

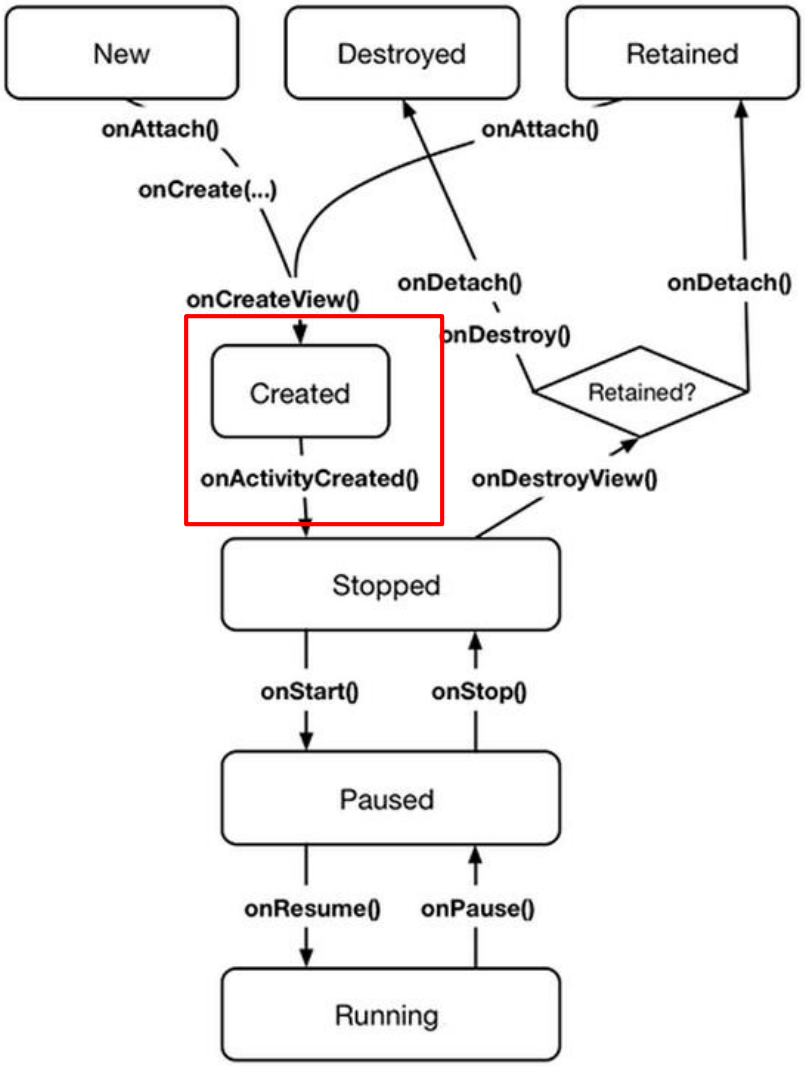


- **FragmentManager** calls fragment lifecycle methods
- **onAttach( ), onCreate( )** and **onCreateView()** called when a fragment is added to **FragmentManager**

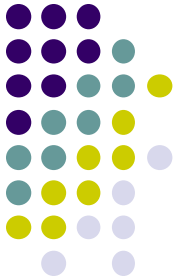


1. **First create fragment**
   ….. then wait for Activity to add fragment

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



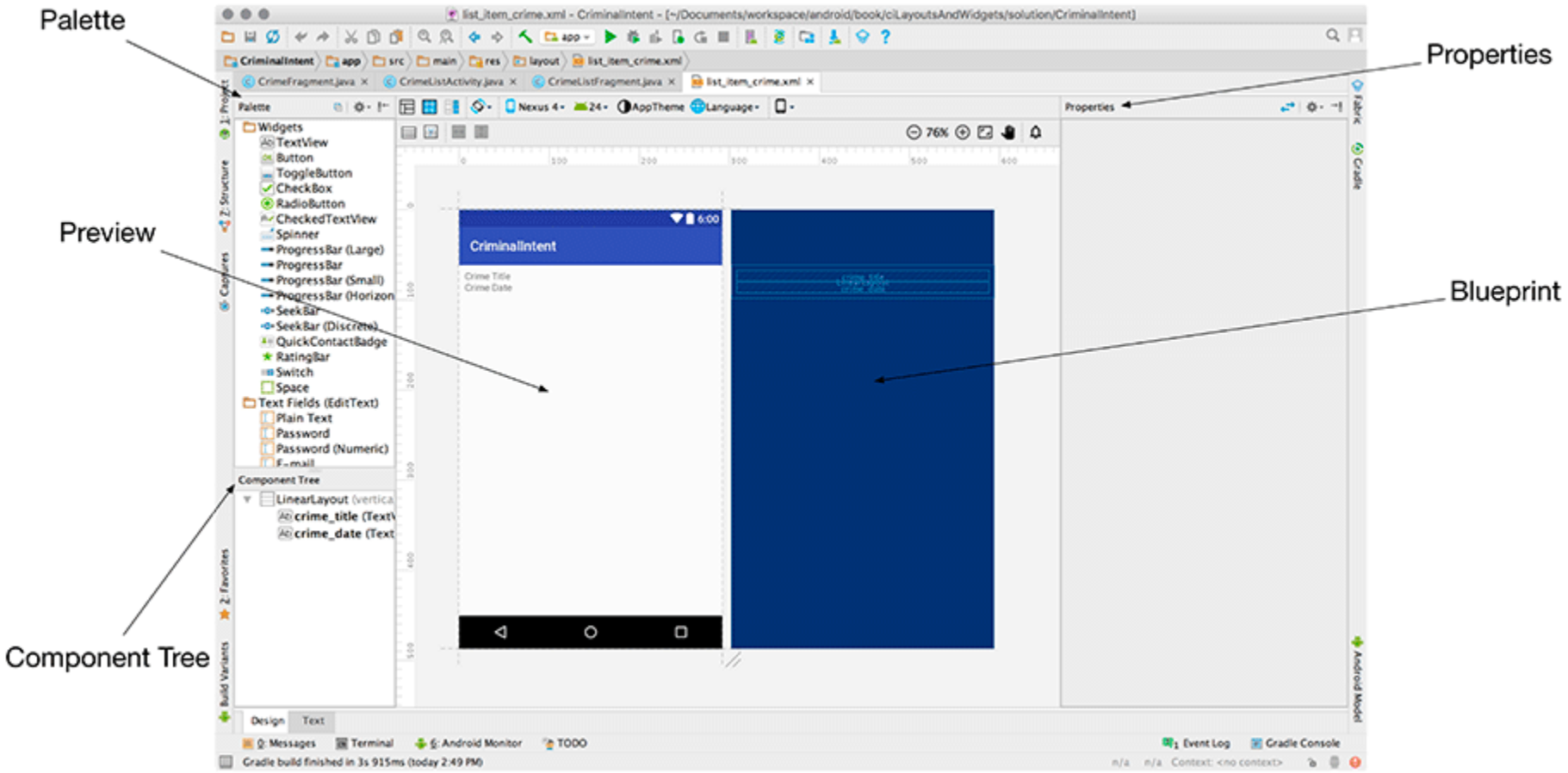
Android Nerd Ranch CriminalIntent Chapters Skipped

Chapter 8: Displaying Lists with RecyclerView

- Skipped several **UI chapters**
- These features are programmed into the **CriminalIntent** code you will be given for project 2
- RecyclerView facilitates view of large dataset
- E.g Allows crimes (title, date) in **CriminalIntent** to be listed

Chapter 9: Creating Android Layouts & Widgets

- Mostly already covered
- Does introduce Contraint Layout (specify widget positions using constraints)

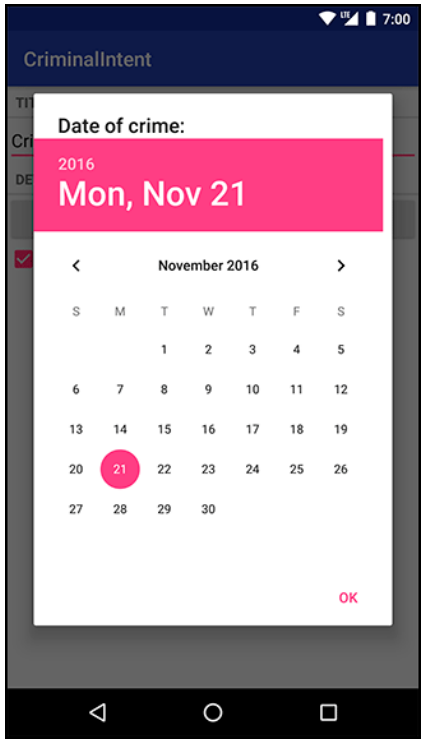


Chapter 11: Using ViewPager

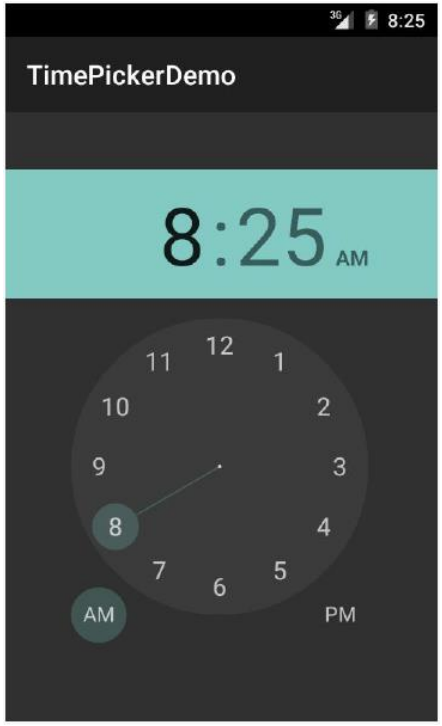
- ViewPager allows users swipe left-right between screens
  - Similar to Tinder
- E.g. Users can swipe left-right between Crimes in CriminalIntent

Chapter 12: Dialogs

- Dialogs present users with a choice or important information
- DatePicker allows users pick date
- Users can pick a date on which a crime occurred in **CriminalIntent**



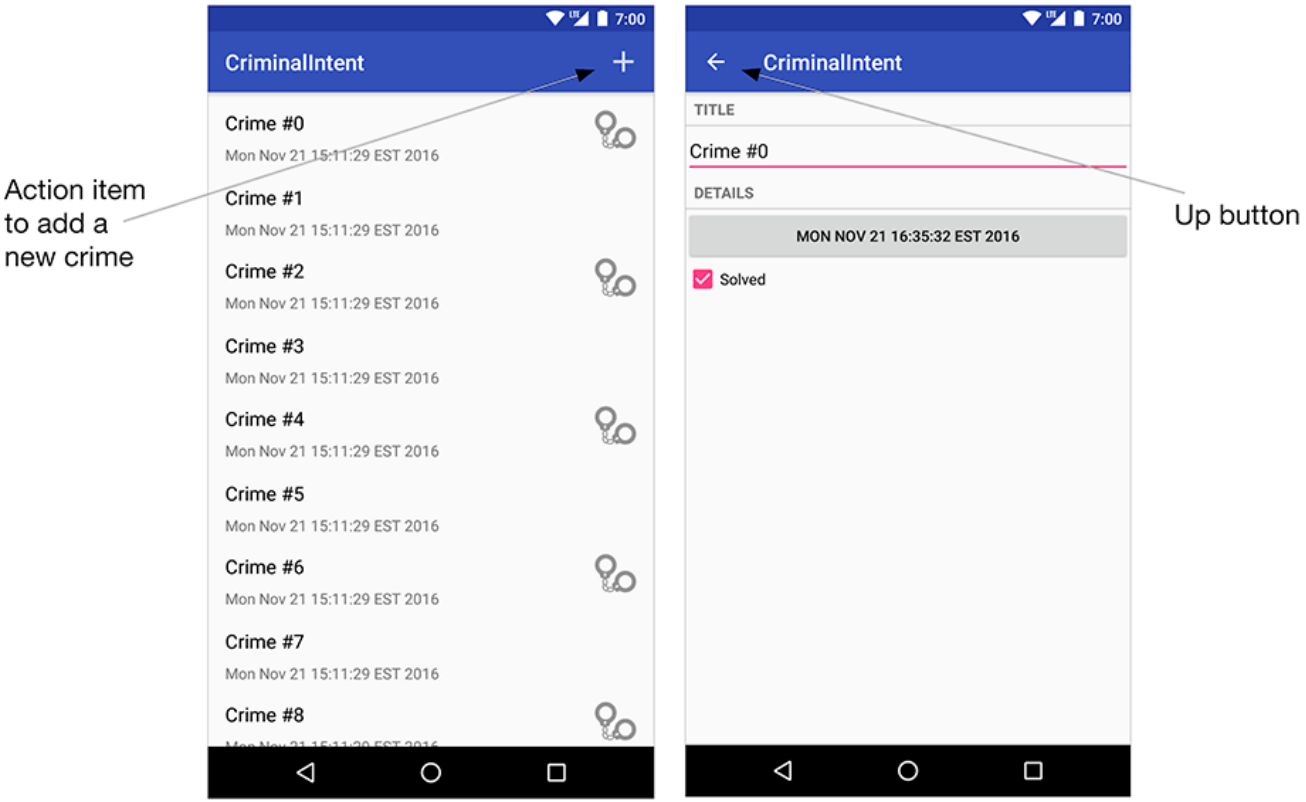
DatePicker

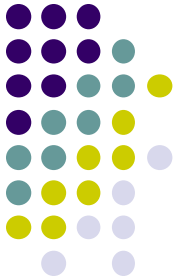


TimePicker also exists

Chapter 13: The Toolbar

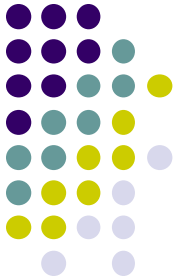
- Toolbar includes actions user can take
- In CriminalIntent, menu items for adding crime, navigate up the screen hierarchy



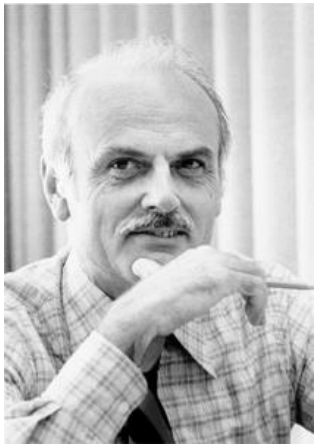


Android Nerd Ranch Ch 14
SQLite Databases

Background on Databases



- **Note:** Google now have new database API (Room)
  - But we will use SQLite here, as book uses it
- Relational DataBase Management System (R
  - Introduced by E. F. Codd (Turing Award Winner)



- Relational Database
  - data stored in tables
  - relationships among data stored in tables
  - data can be accessed and viewed in different ways

Example Wines Database

- **Relational Data:** Data in different tables can be related

Winery Table

Winery ID	Winery name	Address	Region ID
1	Moss Brothers	Smith Rd.	3
2	Hardy Brothers	Jones St.	1
3	Penfolds	Arthurton Rd.	1
4	Lindemans	Smith Ave.	2
5	Orlando	Jones St.	1

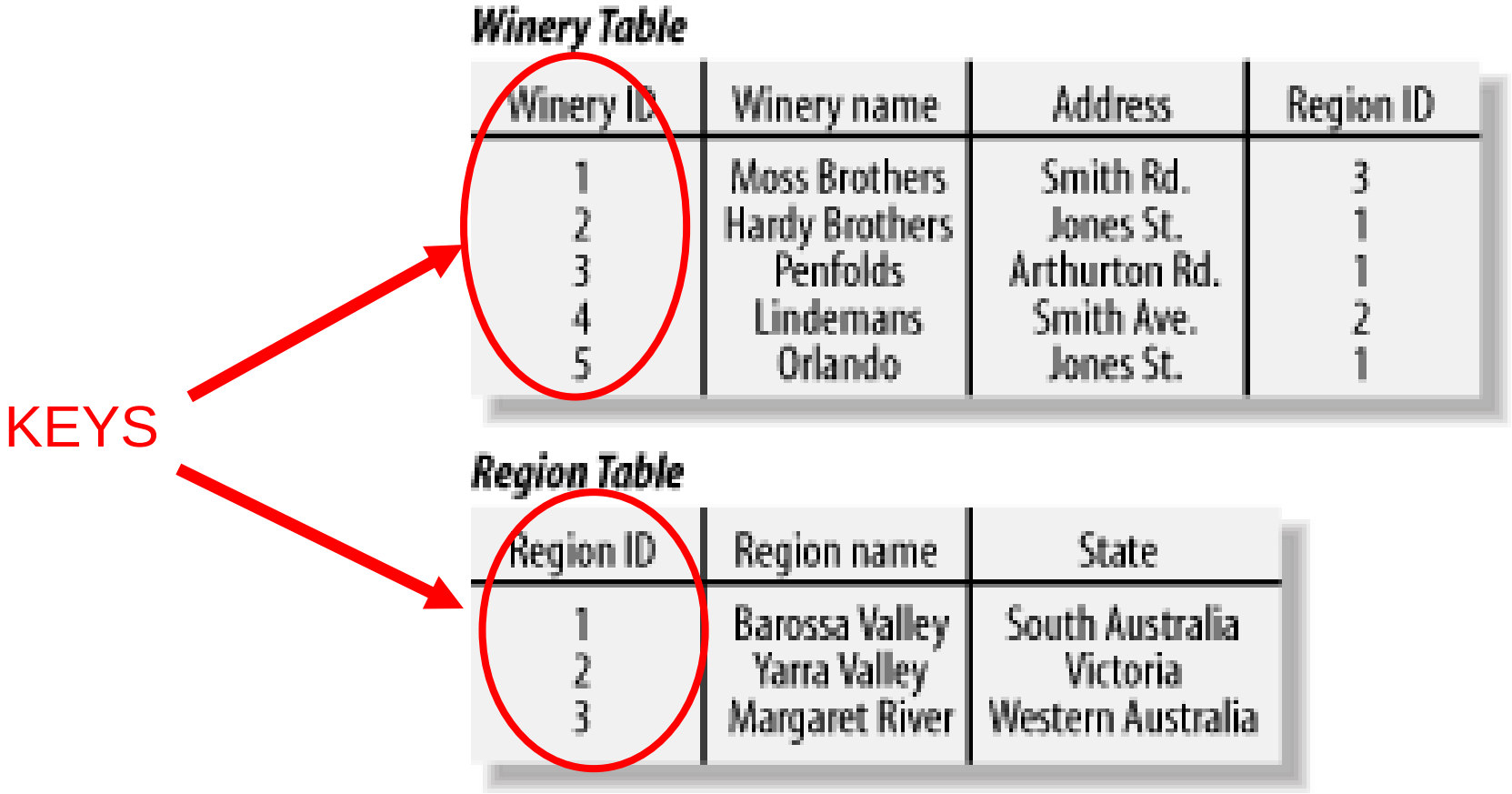
Region Table

Region ID	Region name	State
1	Barossa Valley	South Australia
2	Yarra Valley	Victoria
3	Margaret River	Western Australia

Ref: Web Database Applications with PHP and MySQL, 2nd Edition , by Hugh E. Williams, David Lane

Keys

- Each table has a key
- **Key:** column used to uniquely identify each row



SQL and Databases

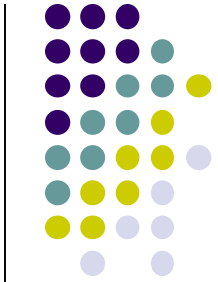
- **SQL:** language used to manipulate Relational Database (RDBMS)

- SQL Commands:
  - **CREATE TABLE** - creates new database table
  - **ALTER TABLE** - alters a database table
  - **DROP TABLE** - deletes a database table
  - **SELECT** - get data from a database table
  - **UPDATE** - change data in a database table
  - **DELETE** - remove data from a database table
  - **INSERT INTO** - insert new data in a database table

Region Table

Region ID	Region name	State
1	Barossa Valley	South Australia
2	Yarra Valley	Victoria
3	Margaret River	Western Australia

CriminalIntent Database



- **SQLite:** open source relational database
- SQLite implements subset of SQL (most but not all)
  - http://www.sqlite.org/
- Android includes a SQLite database
- **New:** Android higher level database API called Room
- **Goal:** Store crimes in CriminalIntent in SQLite database
- First step, define database table of **crimes**

_id	uuid	title	date	solved
1	13090636733242	Stolen yogurt	13090636733242	0
2	13090732131909	Dirty sink	13090732131909	1

CriminalIntent Database Schema

- Create **CrimeDbSchema** class to store **crime** database
- Define fields/columns of the Crimes database table

```
public class CrimeDbSchema {
    public static final class CrimeTable {
        public static final String NAME = "crimes";

        public static final class Cols {
            public static final String UUID = "uuid";
            public static final String TITLE = "title";
            public static final String DATE = "date";
            public static final String SOLVED = "solved";
        }
    }
}
```

Name of Table

Each Crimes Table has the following fields/columns

_id	uuid	title	date	solved
1	13090636733242	Stolen yogurt	13090636733242	0
2	13090732131909	Dirty sink	13090732131909	1

Crimes Table

SQLiteOpenHelper

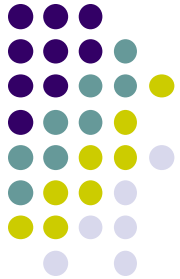
- **SQLiteOpenHelper** class used for database creation, opening and updating a **SQLiteDatabase**
- In **CriminalIntent**, create subclass of **SQLiteOpenHelper** called **CrimeBaseHelper**

```
public class CrimeBaseHelper extends SQLiteOpenHelper {
    private static final int VERSION = 1;
    private static final String DATABASE_NAME = "crimeBase.db";

    public CrimeBaseHelper(Context context) {
        super(context, DATABASE_NAME, null, VERSION);
    }


    @Override
    public void onCreate(SQLiteDatabase db) {

    }

    @Override
    public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {

    }
}
```

Used to create the database (to store Crimes)

Called the first time database is created

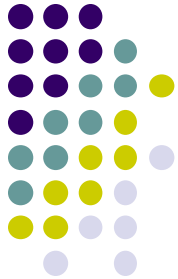
Use CrimeBaseHelper to open SQLite Database



```
public class CrimeLab {
    private static CrimeLab sCrimeLab;

    private List<Crime> mCrimes;
    private Context mContext;
    private SQLiteDatabase mDatabase;
    ...
    private CrimeLab(Context context) {
      mContext = context.getApplicationContext();
      mDatabase = new CrimeBaseHelper(mContext)
              .getWritableDatabase();
      mCrimes = new ArrayList<>();
    }
```

Open new writeable Database

Create CrimeTable in onCreate( )

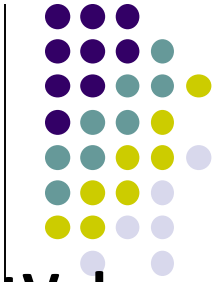


onCreate called first time database is created

```
@Override
public void onCreate(SQLiteDatabase db) {
    db.execSQL("create table " + CrimeTable.NAME + "(" +
            " _id integer primary key autoincrement, " +
            CrimeTable.Cols.UUID + ", " +
            CrimeTable.Cols.TITLE + ", " +
            CrimeTable.Cols.DATE + ", " +
            CrimeTable.Cols.SOLVED +
            ")"
    );
}
```

Create CrimeTable in our new Crimes Database

Writing Crimes to Database using ContentValues



- In Android, writing to databases is done using class **ContentValues**
- **ContentValues** is key-value pair
- Create method to create **ContentValues** instance from a **Crime**

```
    public Crime getCrime(UUID id) {
        return null;
    }

    private static ContentValues getContentValues(Crime crime) {
        ContentValues values = new ContentValues();
        values.put(CrimeTable.Cols.UUID, crime.getId().toString());
        values.put(CrimeTable.Cols.TITLE, crime.getTitle());
        values.put(CrimeTable.Cols.DATE, crime.getDate().getTime());
        values.put(CrimeTable.Cols.SOLVED, crime.isSolved() ? 1 : 0);

        return values;
    }
}
```

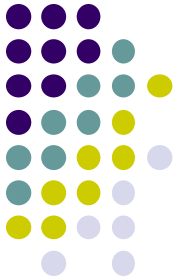
Takes Crime as input

key

value

Converts Crime to ContentValues

Returns values as output



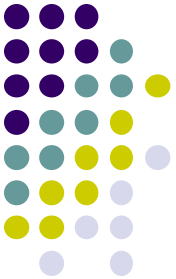
Firebase Cloud API

Firebase

- Mobile cloud backend service for
  - Analytics
  - Messaging
  - Authentication
  - Database
  - Crash reporting, etc

- Previously 3rd party company
- Acquired by Google in 2014
  - Now part of Google. See https://firebase.google.com/
  - Fully integrated, could speed up development. E.g. final project

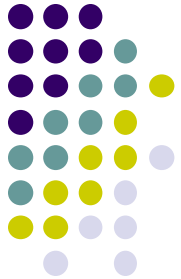
Firebase



- Relatively easy programming, few lines of code
- E.g. to create database

```
FirebaseDatabase database = FirebaseDatabase.getInstance()
// write
database.child("users").child("userId").setValue(user);

// read / listen
database.child("users").addValueEventListener(new ValueEventListener() {
    @Override
    public void onDataChange(DataSnapshot dataSnapshot) {
        // ...
    }

    @Override
    public void onCancelled(DatabaseError databaseError) {}
});
```

References



- 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