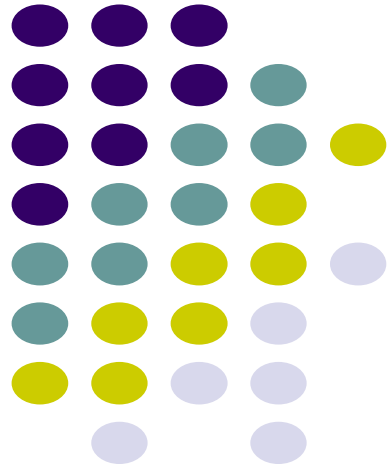
CS 528 Mobile and Ubiquitous Computing

Lecture 4a: Intents & Fragments

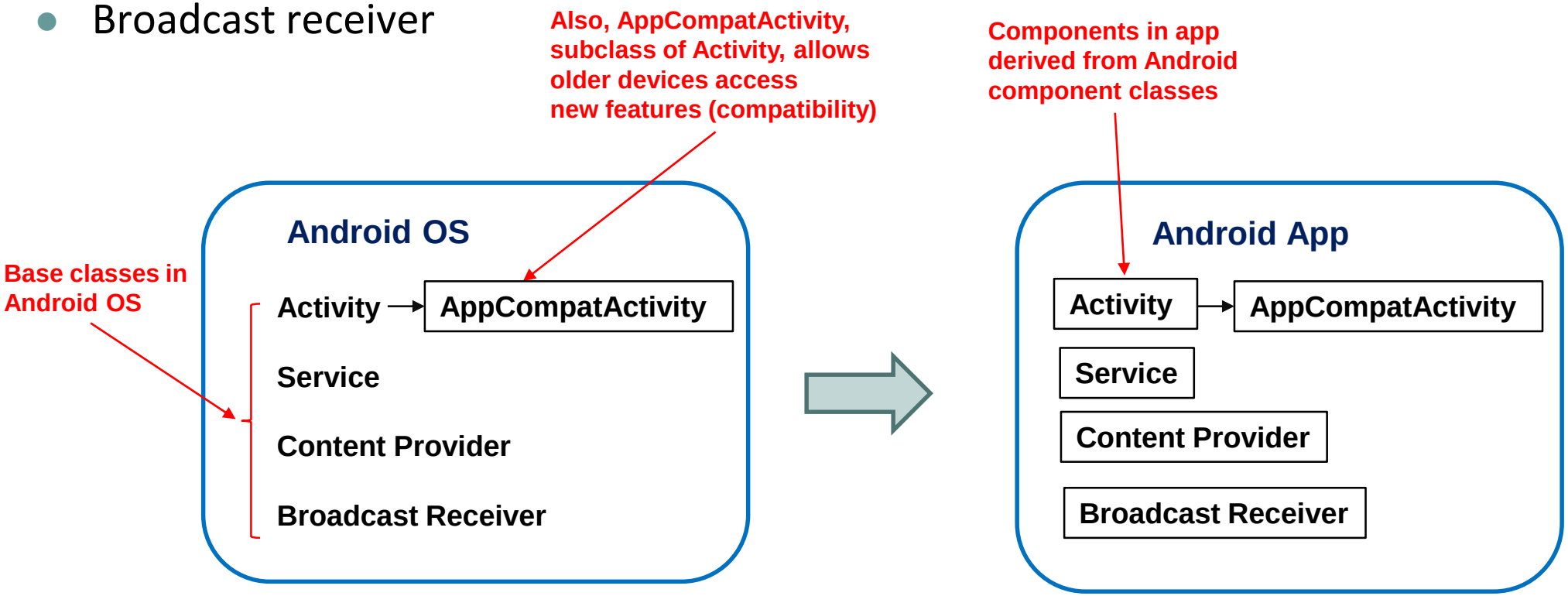
Emmanuel Agu



Intents

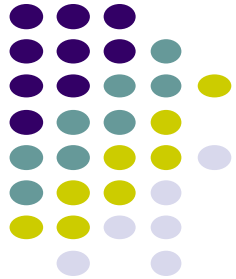
Recall: Android App Components

- 4 main types of Android app components:
  - Activity (already seen this), or AppCompatActivity
  - Service
  - Content provider
  - Broadcast receiver

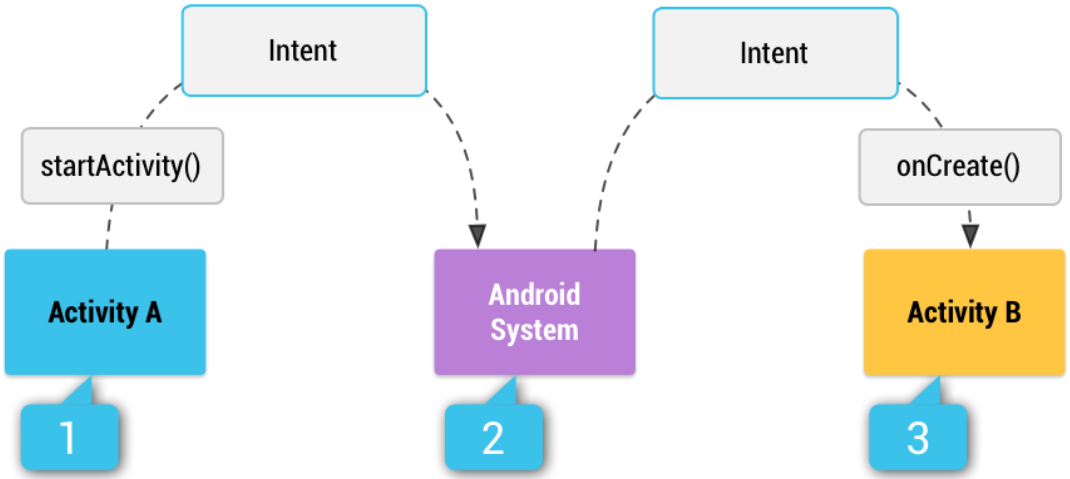


Intent

https://developer.android.com/training/basics/intents
https://developer.android.com/guide/components/intents-filters

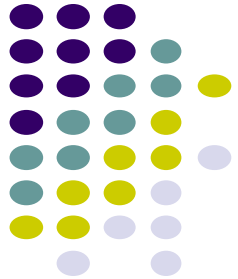


- **Intent:** a messaging object used by a component to request action from another app or component
- 3 main use cases for Intents
- **Case 1a (Activity A starts Activity B, no result back):**
  - **Note:** Activity A cannot start Activity B directly. Asks Android system to create Activity B
  - Activity A calls **startActivity( )**, passes an Intent
  - Intent contains information about Activity B + any necessary data

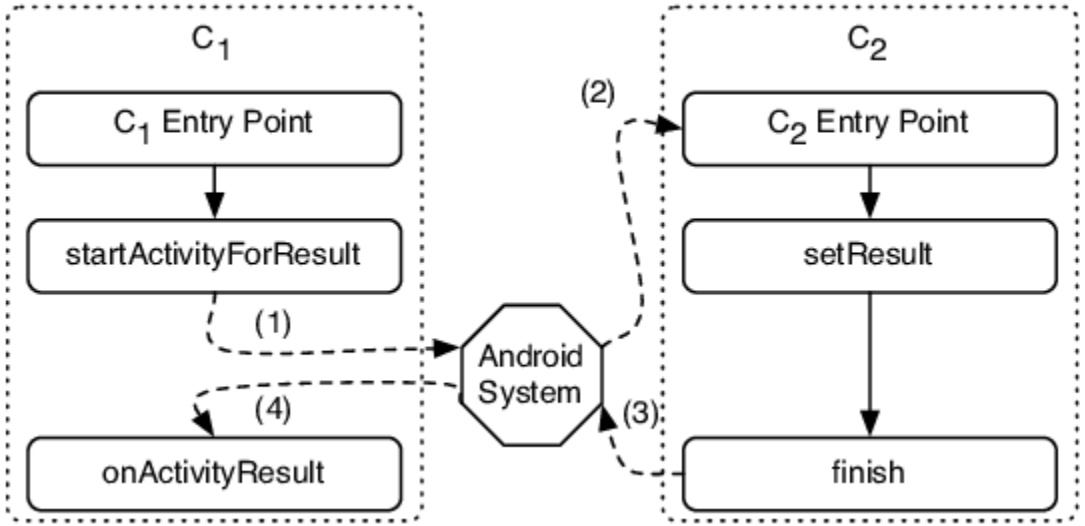


Intent: Result Received Back

https://developer.android.com/training/basics/intents/result

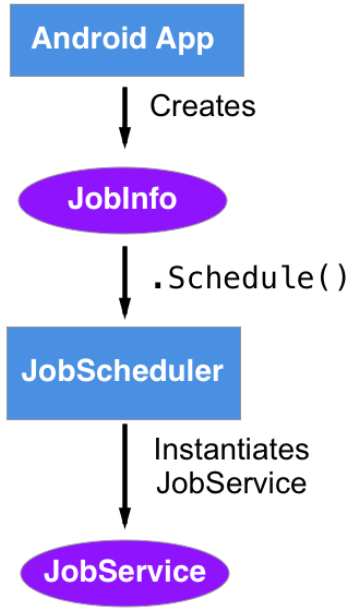


- **Case 1b (Activity A starts Activity B, gets result back):**
  - Activity A calls **startActivityForResult( )**, pass an Intent (1)
  - Separate Intent (4) received in Activity A's **onActivityResult( )** callback

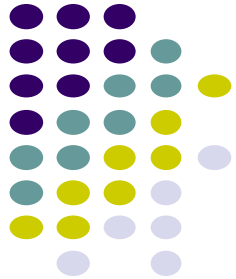


Intent: Result Received Back

- **Case 2 (Activity A starts a Service):**
  - E.g. Activity A starts service to download big file in the background
  - Before Android 5.0
    - Activity A calls **StartService( )**, passes an Intent
    - Intent contains information about Service to start, plus any necessary data
  - From Android 5.0:
    - Use **JobScheduler** to start service
- **Case 3 (Delivering a broadcast)**
  - Pass intent to **send Broadcast( )** or **sendOrderedBroadcast( )**



Ref: https://medium.com/mindorks/android-jobschedulers-usages-3c241d795d0f

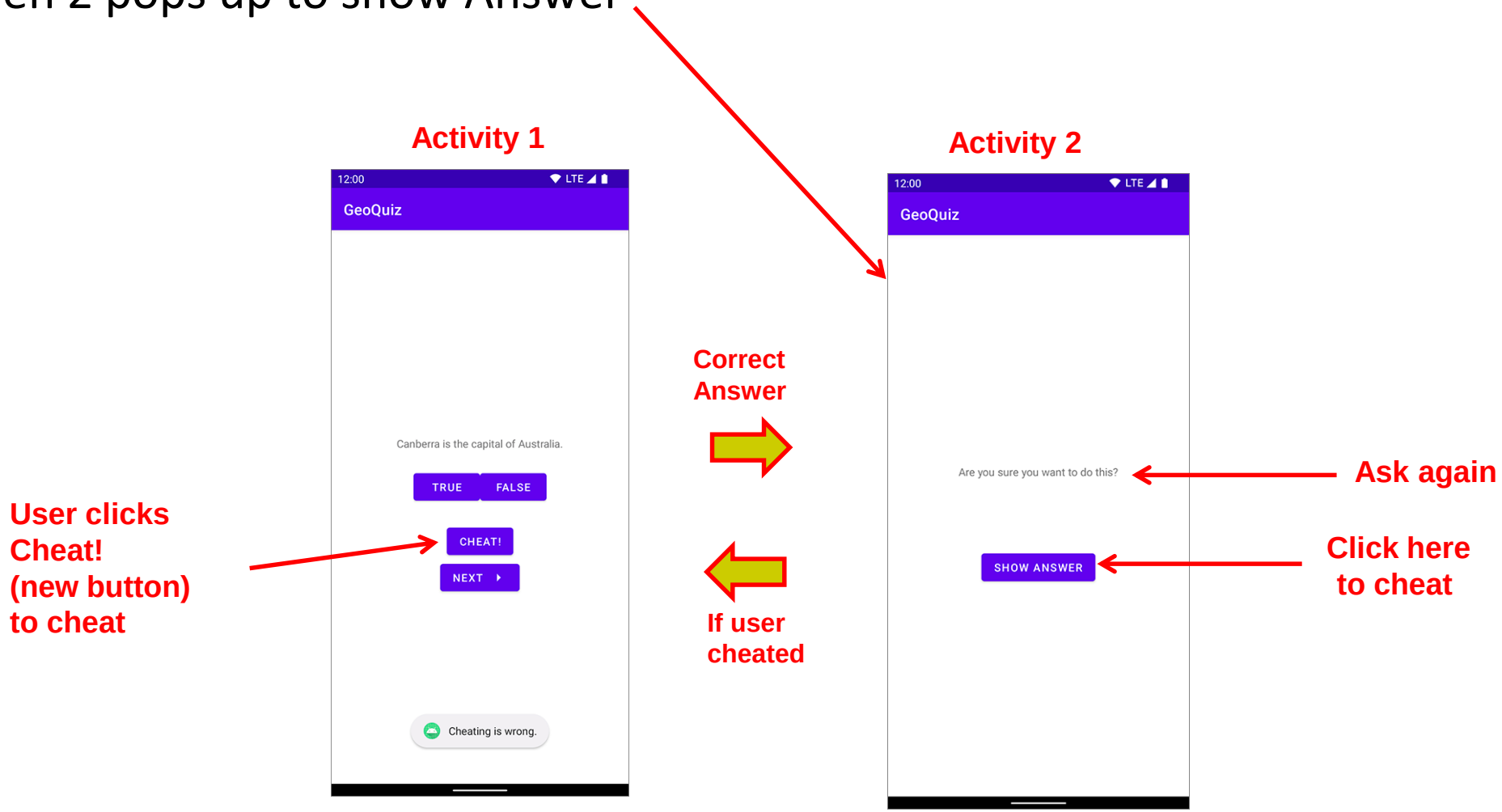


Intent Example: Starting Activity 2 from Activity 1

Allowing User to Cheat

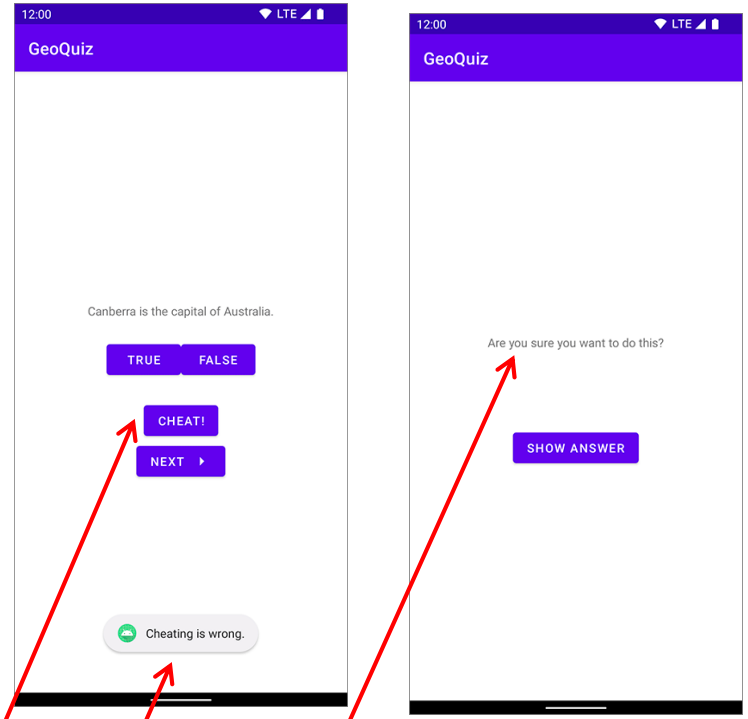
Ref: Android Nerd Ranch (5th edition), Chapter 7

- **Goal:** Allow user to cheat by getting answer to quiz
- Screen 2 pops up to show Answer



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

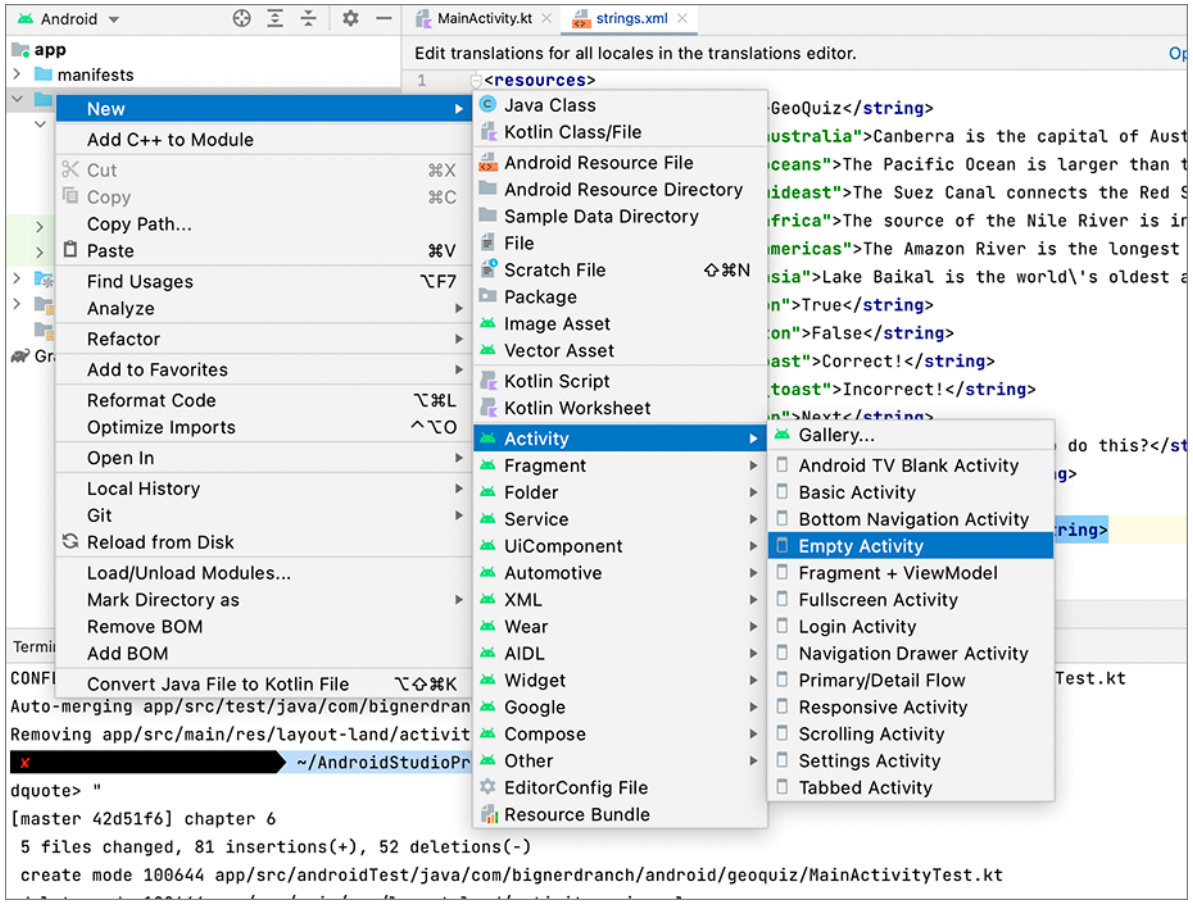
Ref: Android Nerd Ranch (5th edition), Chapter 7

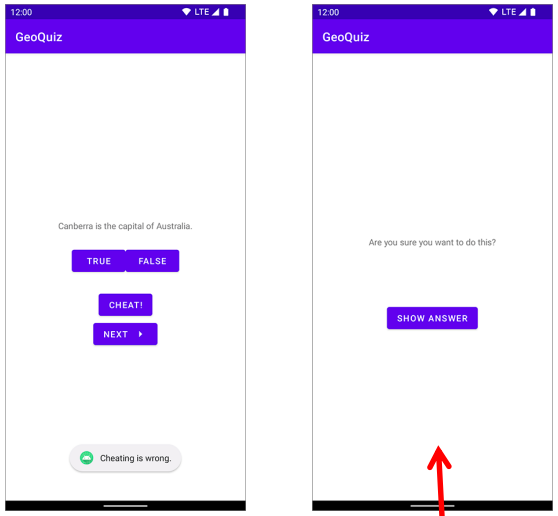



```
<resources>
    ...
    <string name="incorrect_toast">Incorrect!</string>
    <string name="warning_text">Are you sure you want to do this?</string>
    <string name="show_answer_button">Show Answer</string>
    <string name="cheat_button">Cheat!</string>
    <string name="judgment_toast">Cheating is wrong.</string>
</resources>
```

Create Empty Activity (for Activity 2, 2nd screen) in Android Studio

Ref: Android Nerd Ranch (5th edition), Chapter 7

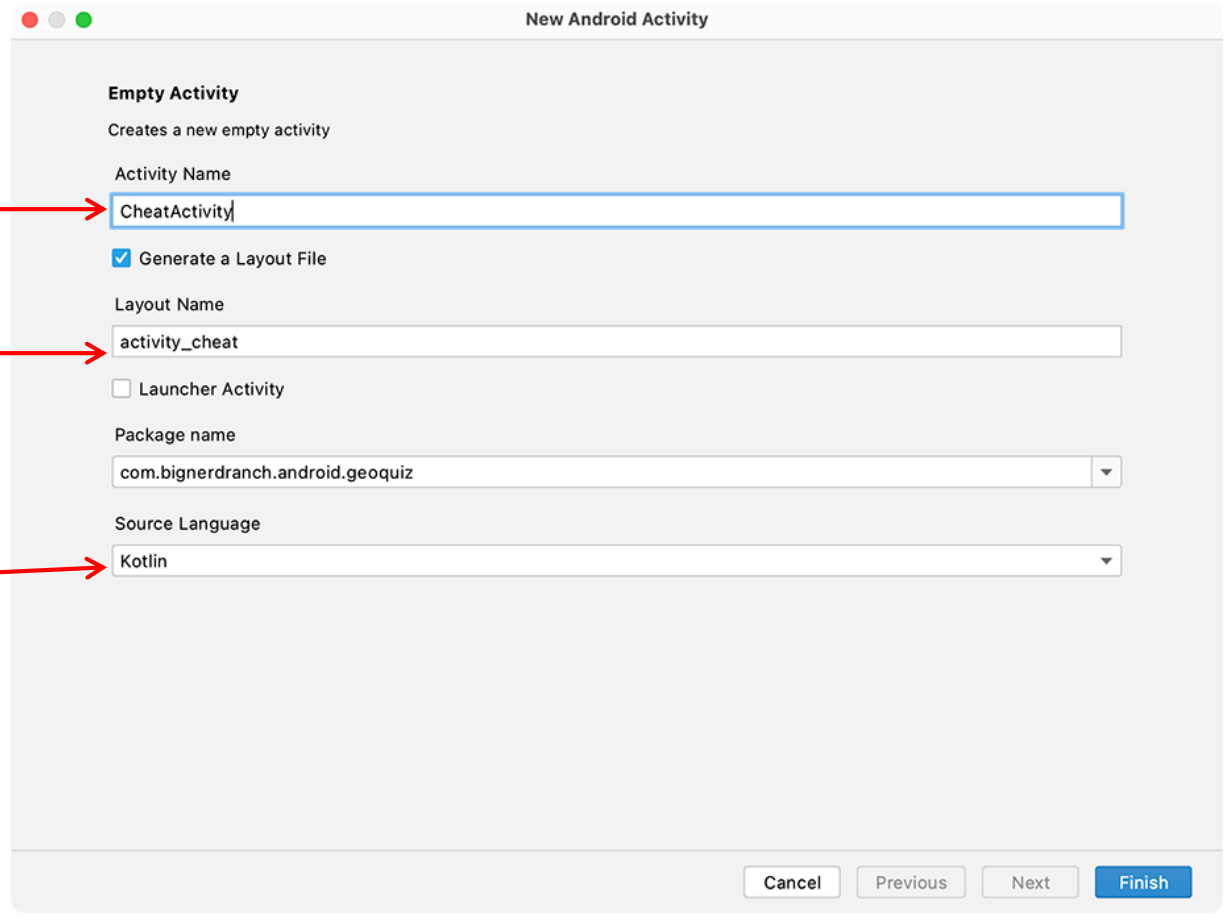




Create empty Activity for second screen

Specify Kotlin and XML file names for Activity 2

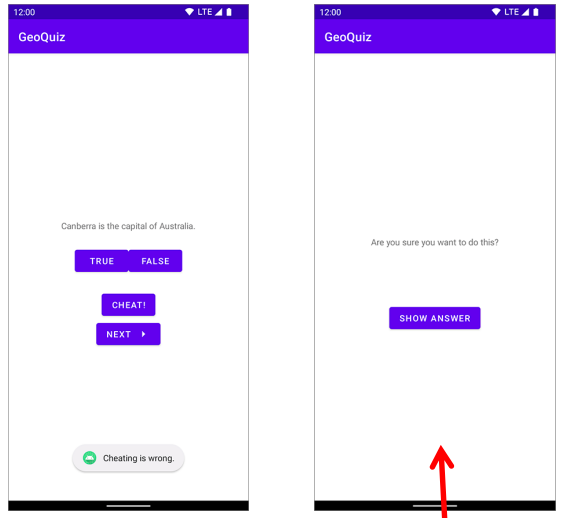
Ref: Android Nerd Ranch (5th edition), Chapter 7



Screen 2 kotlin code in CheatActivity.kt

Layout uses activity_cheat.xml

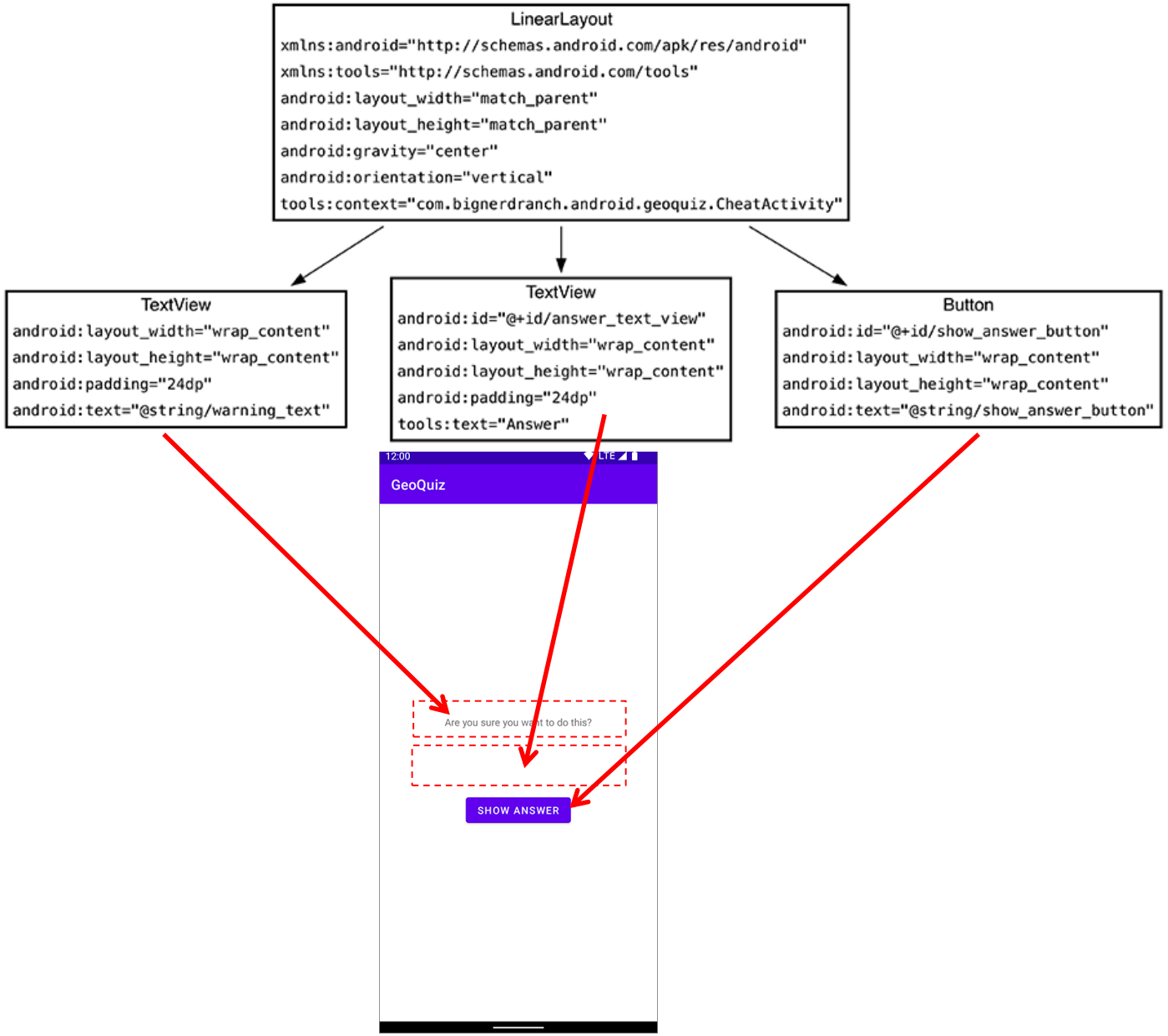
Select kotlin as language



2 Files:
1.CheatActivity.kt
2. activity_cheat.xml

Design Layout for Screen 2

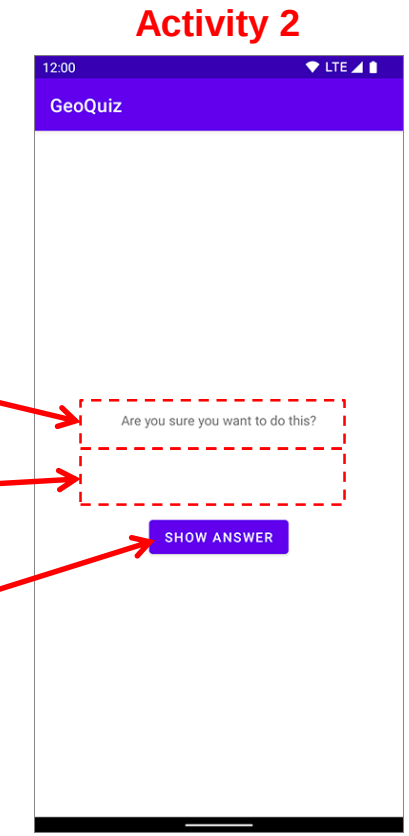
Ref: Android Nerd Ranch (5th edition), Chapter 7



Write XML Layout Code for Screen (Activity) 2

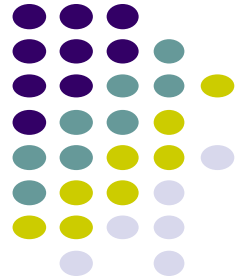
Ref: Android Nerd Ranch (5th edition), Chapter 7

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

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

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

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

</LinearLayout>
```



Use View Binding in CheatActivity (CheatActivity.kt)

Ref: Android Nerd Ranch (5th edition), Chapter 7



- Android study generates code for older approach (without view bindings)
- In code auto-generated by Android Studio, modify **setContentView( )** to use view bindings
- **Note:** activity_cheat.xml was already set as layout file for ActivityCheat.kt when creating new activity

```
class CheatActivity : AppCompatActivity() {
    private lateinit var binding: ActivityCheatBinding

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_cheat)
        binding = ActivityCheatBinding.inflate(layoutInflater)
        setContentView(binding.root)
    }
}
```

Old way (auto-generated)

New way

Declare New Activity (CheatActivity) in AndroidManifest.xml

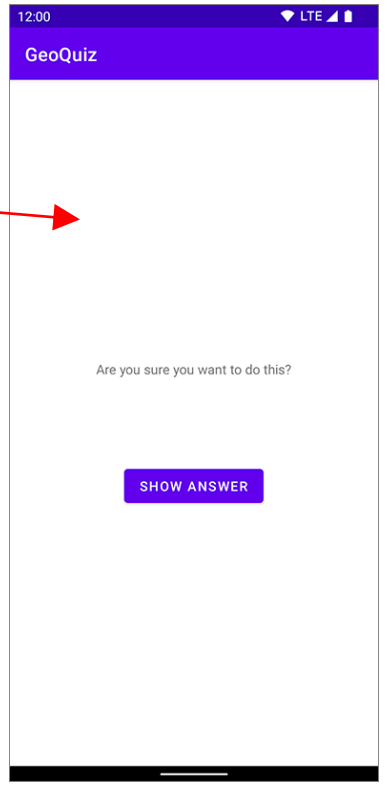
Ref: Android Nerd Ranch (5th edition), Chapter 7

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

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/Theme.GeoQuiz">
        <activity
            android:name=".CheatActivity"
            android:exported="false" />
        <activity
            android:name=".MainActivity"
            android:exported="true">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

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

</manifest>
```

Add Activity 2 (CheatActivity)

Activity 1

Activity 2 (CheatActivity)



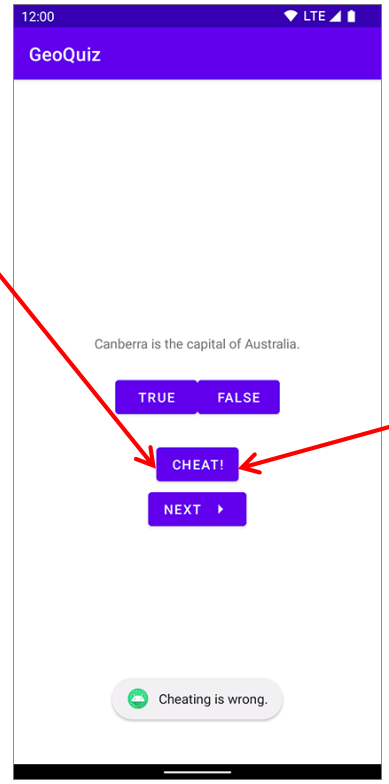
Add Cheat Button

Ref: Android Nerd Ranch (5th edition), Chapter 7

- Add cheat button code to XML layout file

```
        ...
    </LinearLayout>

    <Button
        android:id="@+id/cheat_button"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginTop="24dp"
        android:text="@string/cheat_button" />

    <Button
        android:id="@+id/next_button"
        .../>

</LinearLayout>
```

Activity 1



User clicks Cheat! (new button) to cheat

- Add click Listener code to kotlin file
  (kotlin code that responds to user click)

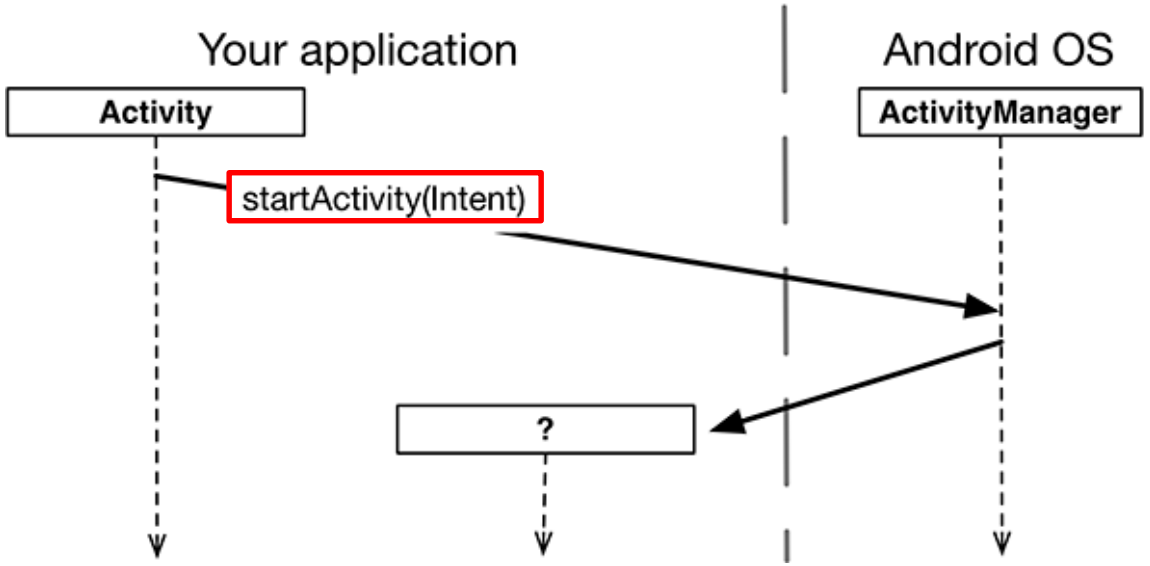
```
class MainActivity : AppCompatActivity() {

    private lateinit var binding: ActivityMainBinding

    private val quizViewModel: QuizViewModel by viewModels()

    override fun onCreate(savedInstanceState: Bundle?) {
        ...
        binding.nextButton.setOnClickListener {
            quizViewModel.moveToNext()
            updateQuestion()
        }

        binding.cheatButton.setOnClickListener {
            // Start CheatActivity
        }

        updateQuestion()
    }
    ...
}
```

Starting Activity 2 from Activity 1

Ref: Android Nerd Ranch (5th edition), Chapter 7

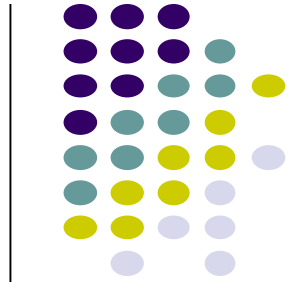
- Activity 1 starts activity 2 by calling **startActivity(Intent)**
  - **through** the Android OS

- Passes Intent (object for communicating) to Android OS
  - Intent specifies which Activity Android ActivityManager should start



Starting Activity 2 from Activity 1

Ref: Android Nerd Ranch (5th edition), Chapter 7



- Code to create intent, tell Android OS to start CheatActivity:

```
binding.cheatButton.setOnClickListener {
    // Start CheatActivity
    val intent = Intent(this, CheatActivity::class.java)
    startActivity(intent)
}
```

Build Intent

Send Intent to Android OS to Start new Activity

Parent Activity

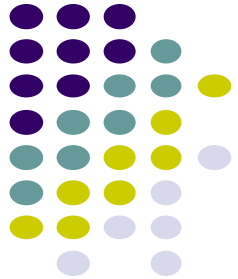
Class for new Activity 2 Android OS should create

- This type of intent called **explicit intent**
  - New activity to start (CheatActivity) is specified
  - Activity 1 and activity 2 are in same app

Question: what does the word **val** mean?

Implicit vs Explicit Intents

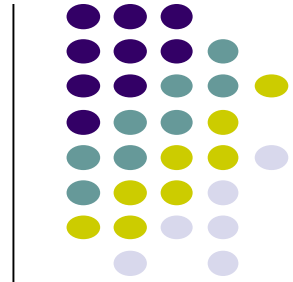
Ref: Android Nerd Ranch (5th edition), Chapter 7



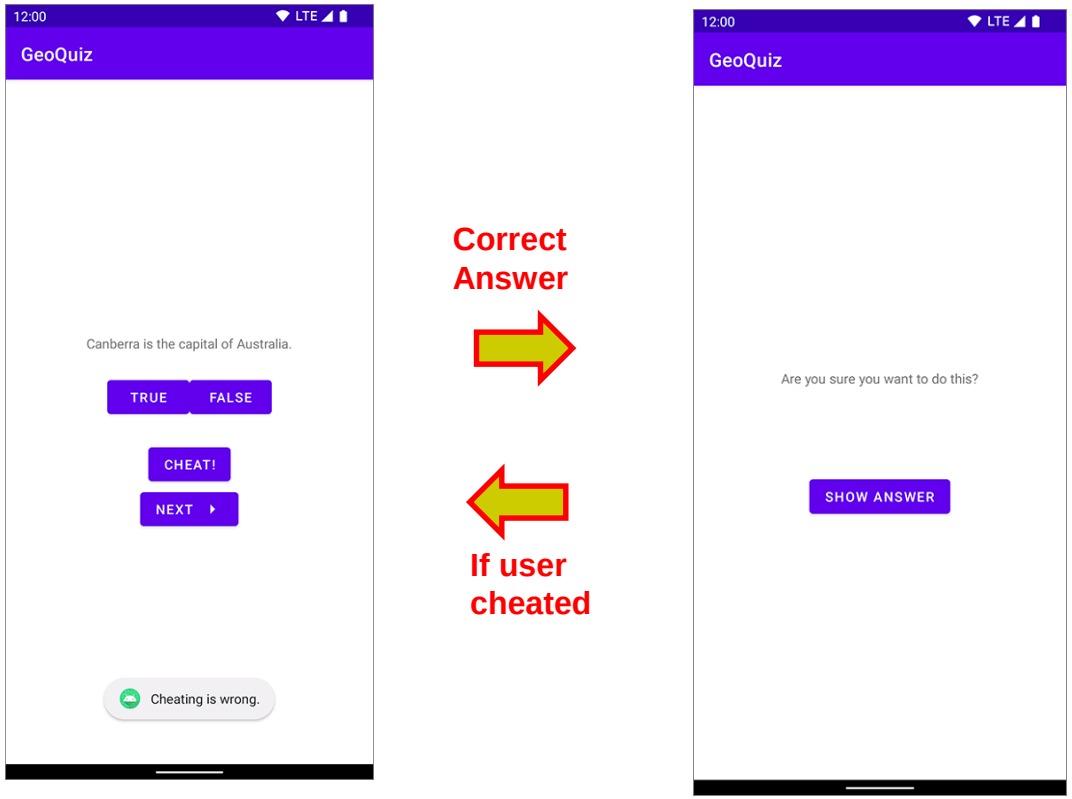
- If Activity 1 and 2 were in different apps, an **implicit intent** would have to be created instead
- Implicit intent specifies action to be taken (e.g. take picture) but not which specific activity will perform the action

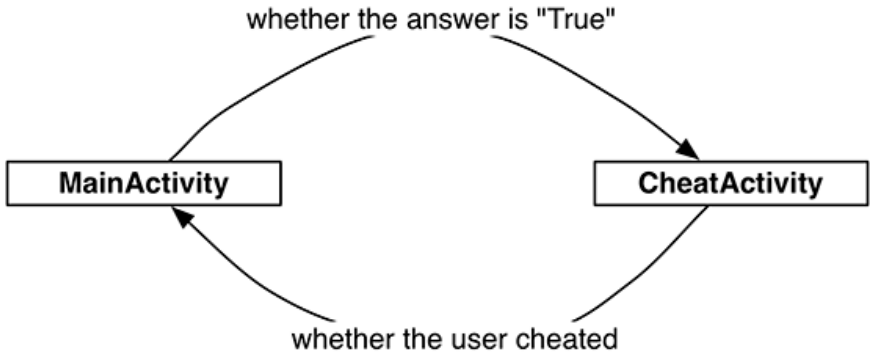
Allowing User to Cheat

Ref: Android Nerd Ranch (5th edition), Chapter 7



- Can also pass data between Activities 1 and 2 (MainActivity and CheatActivity)
  - E.g. Activity 1 can tell Activity 2 correct answer (True/False)

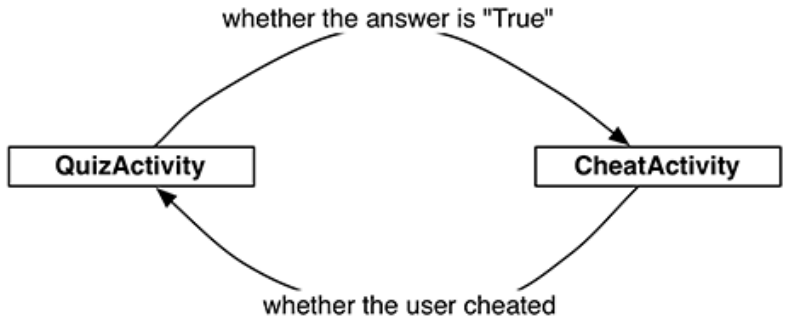




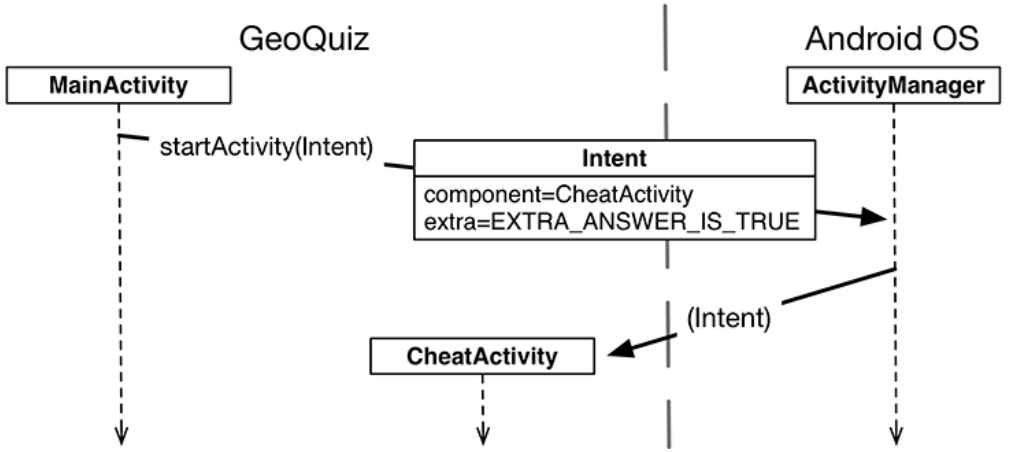
Passing Data Between Activities

Ref: Android Nerd Ranch (5th edition), Chapter 7

- Want to pass answer (True/False from QuizActivity to CheatActivity)



- Pass answer as **extra** in Intent passed into **StartActivity**
- **Extras** are arbitrary data calling activity can include in intent



Passing Answer (True/False) as Intent Extra

Ref: Android Nerd Ranch (5th edition), Chapter 7

- To add **extra** to Intent, use **putExtra( )** command

```
private const val EXTRA_ANSWER_IS_TRUE =
        "com.bignerdranch.android.geoquiz.answer_is_true"
```

```
companion object {
    fun newIntent(packageContext: Context, answerIsTrue: Boolean): Intent {
        return Intent(packageContext, CheatActivity::class.java).apply {
            putExtra(EXTRA_ANSWER_IS_TRUE, answerIsTrue)
        }
    }
}
```

key

value

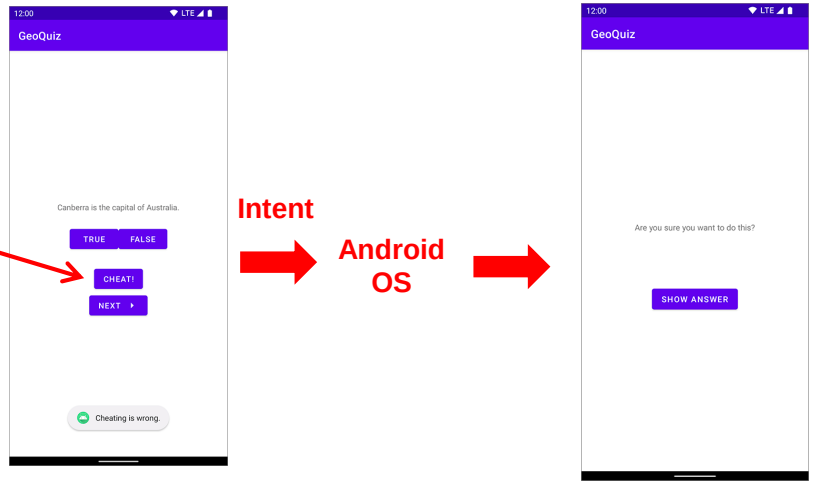
- When user clicks cheat button, build Intent, start new Activity

```
binding.cheatButton.setOnClickListener {
    // Start CheatActivity
    val intent = Intent(this, CheatActivity::class.java)
    val answerIsTrue = quizViewModel.currentQuestionAnswer
    val intent = CheatActivity.newIntent(this@MainActivity, answerIsTrue)
    startActivity(intent)
}
```

Get current answer

Build intent

Use Intent to start activity



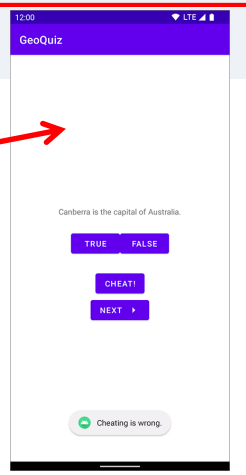
Passing Answer (True/False) as Intent Extra

Ref: Android Nerd Ranch (5th edition), Chapter 7

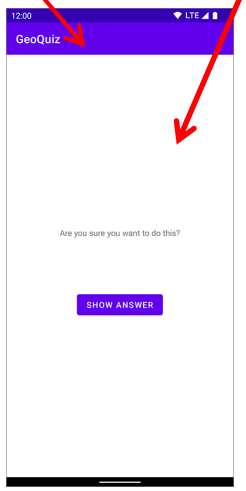
- Activity receiving the Intent (CheatActivity) retrieves it using **getBooleanExtra( )**

```
class CheatActivity : AppCompatActivity() {

    private lateinit var binding: ActivityCheatBinding

    private var answerIsTrue = false

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        binding = ActivityCheatBinding.inflate(layoutInflater)
        setContentView(binding.root)

        answerIsTrue = intent.getBooleanExtra(EXTRA_ANSWER_IS_TRUE, false)
    }
    ...
```

MainActivity Calls startActivity(Intent)



Intent (Answer = Extra)



CheatActivity Calls Intent.getBooleanExtra( )

Important: Read Android Nerd Ranch (5th edition), Chapter 7

Implicit Intents

- Typically, multiple components (apps) can take a given action.
  - E.g. viewing images
- **Implicit Intent:** Does not name component to start, lets system decide
- Specifies
  - **Action** (what to do, example visit a web page)
  - **Data** (to perform operation on, e.g. web page url)
- System decides component to receive implicit intent based on **action**, **data, category**
- Example Implicit Intent to send text

```
// Create the text message with a string.
val sendIntent = Intent().apply {
    action = Intent.ACTION_SEND
    putExtra(Intent.EXTRA_TEXT, textMessage)
    type = "text/plain"
}
```

ACTION (No receiving Activity specified)

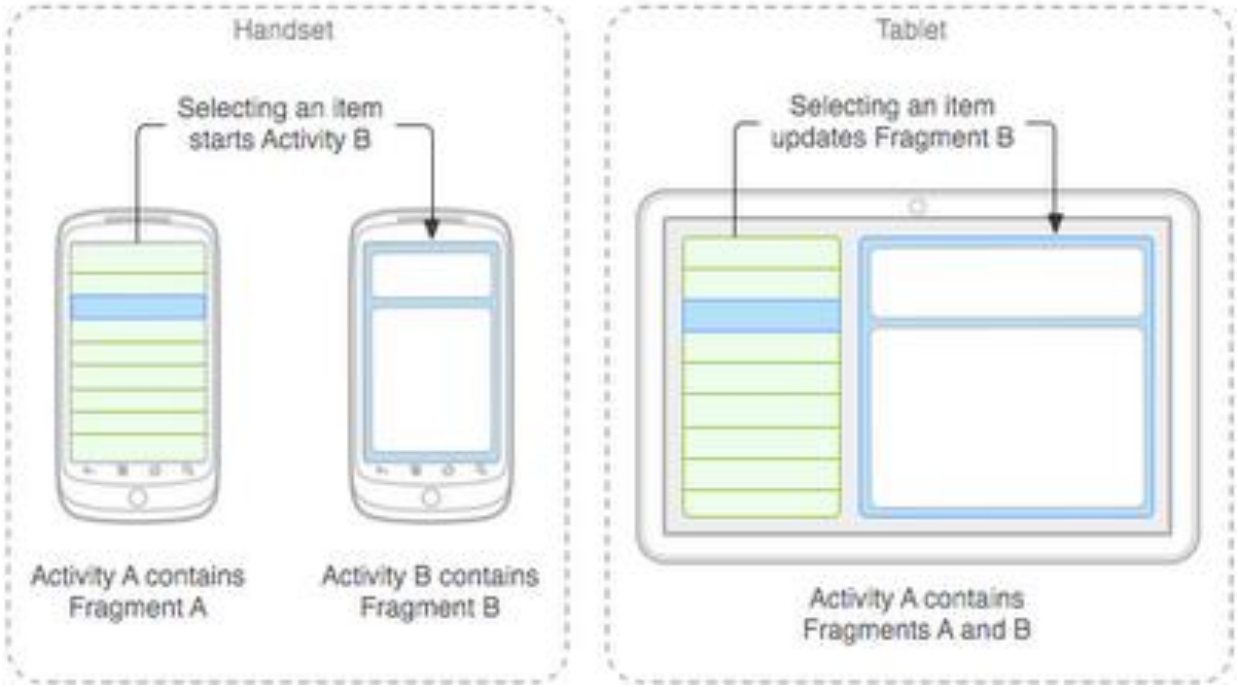
Data type

Ref: https://developer.android.com/guide/components/intents-filters#kotlin

Fragments

Recall: Fragments

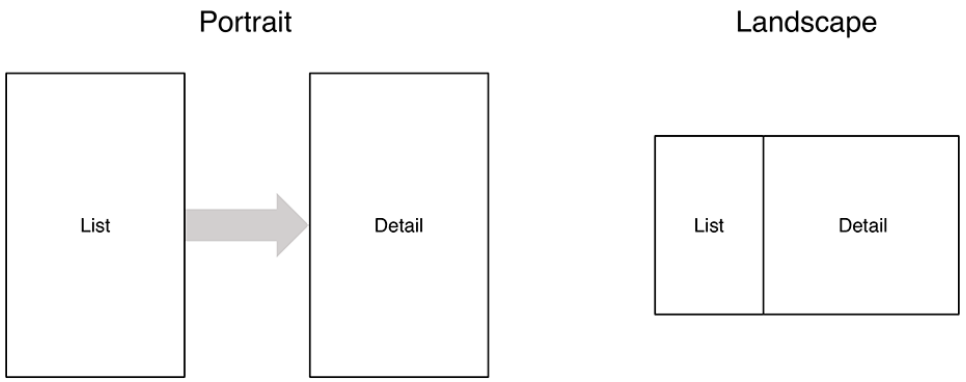
- Sub-components of an Activity (screen)
  - Reusable
- An activity can contain multiple fragments, organized differently on different devices (e.g. phone vs tablet)
- Fragments must be attached to Activities (Screens)



Fragments

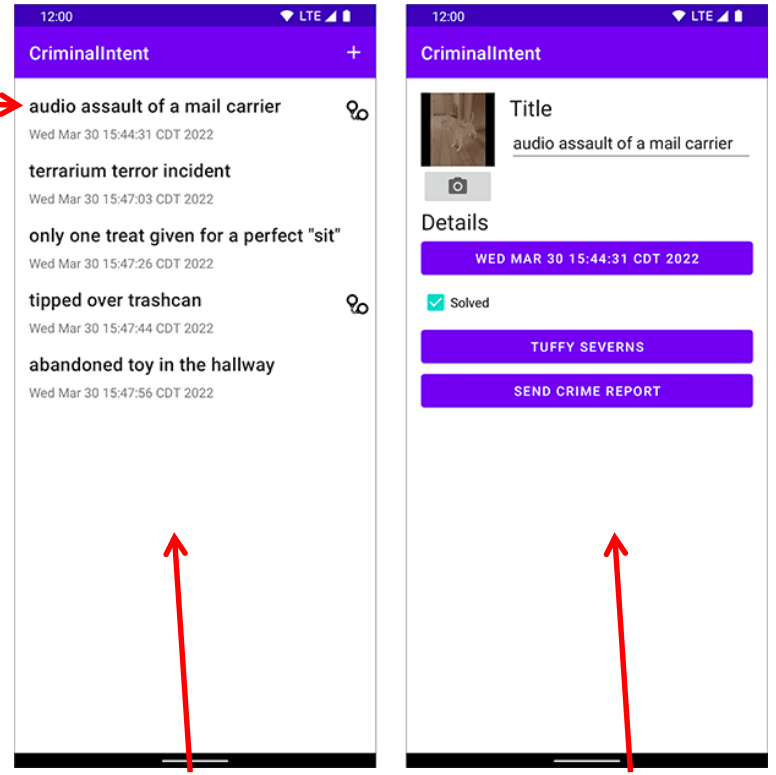
Ref: Android Nerd Ranch (5th edition), Chapter 9

- 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, suspect
- List-detail app using fragments



- **Landscape:** show list + detail
- **Portrait:** swipe to show next crime

Click on Crime to see detail



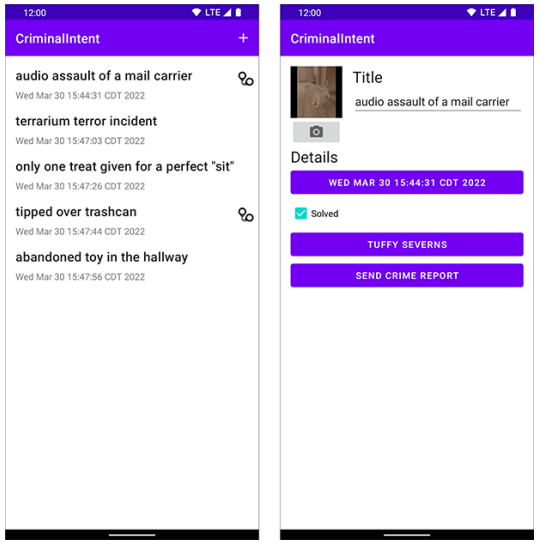
Fragment 1
(list of Crimes)

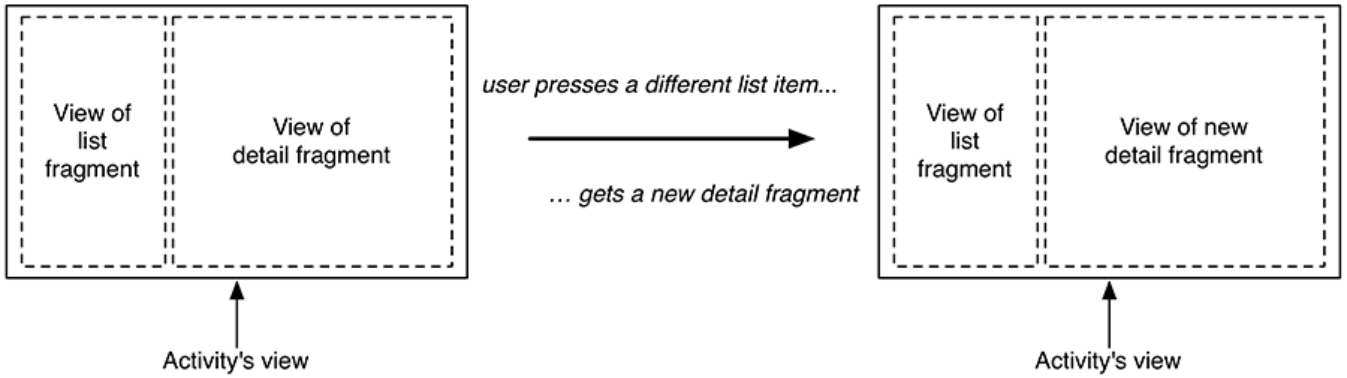
Fragment 2
(Details of selected Crime)

Fragments

Ref: Android Nerd Ranch (5th edition), Chapter 9

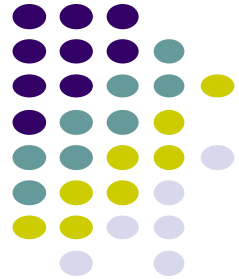
- 1 activity can contain multiple fragments
- Fragment's views are inflated from a layout file
- Can rearrange fragments as desired on an activity
  - E.g. different arrangement in portrait vs. landscape



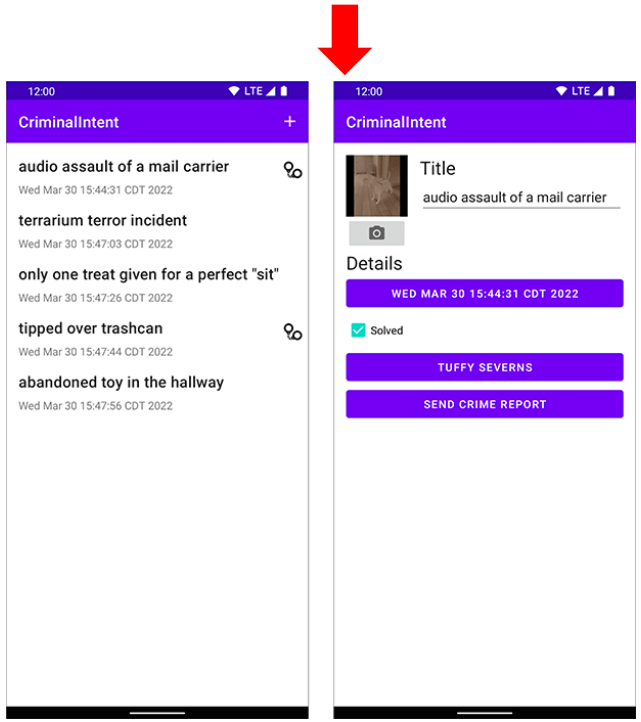


Starting Criminal Intent

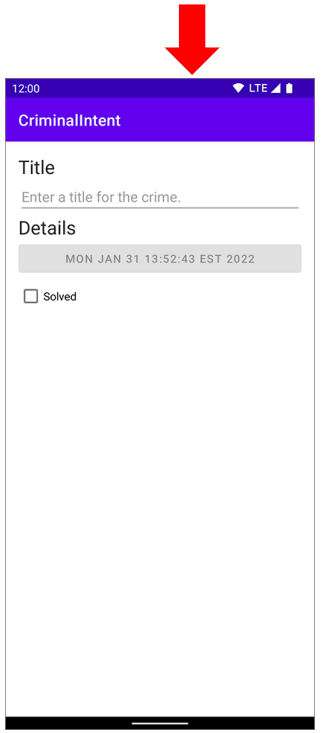
Ref: Android Nerd Ranch (5th edition), Chapter 9



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



Final Look of CriminalIntent
(at end of Chapter 19)

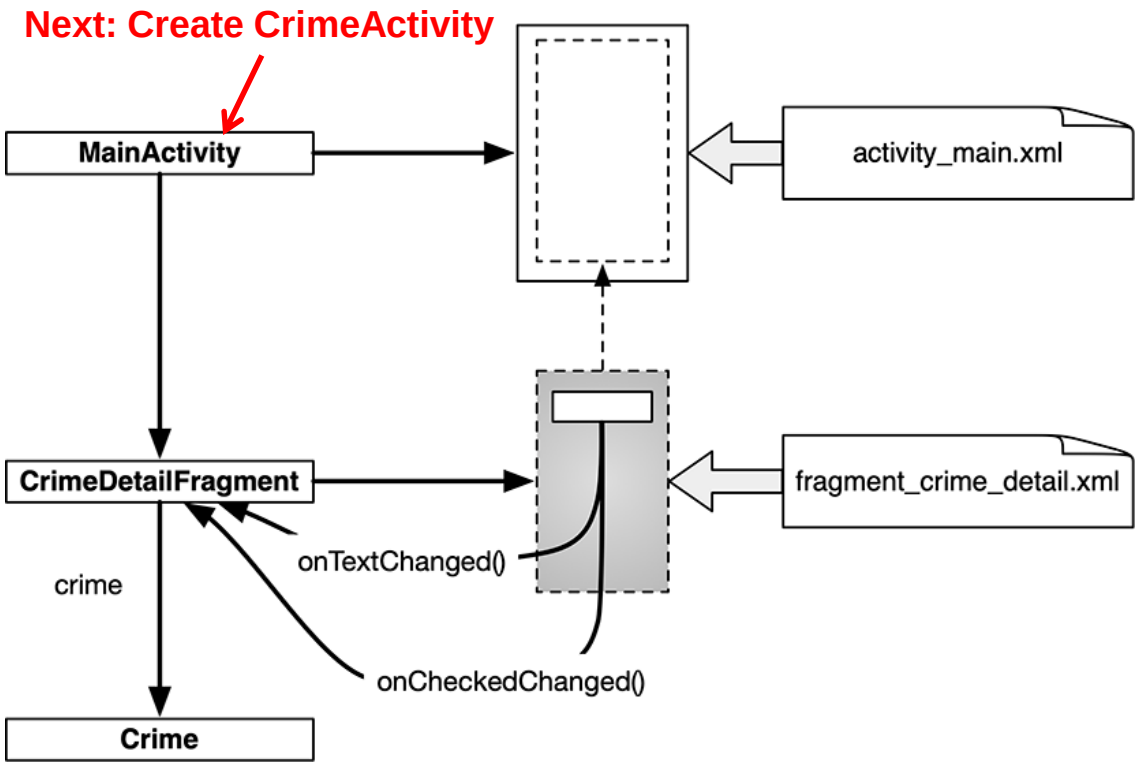


Start small
Develop detail view using Fragments
(CriminalIntent look at end of Chapter 9)

Starting Criminal Intent

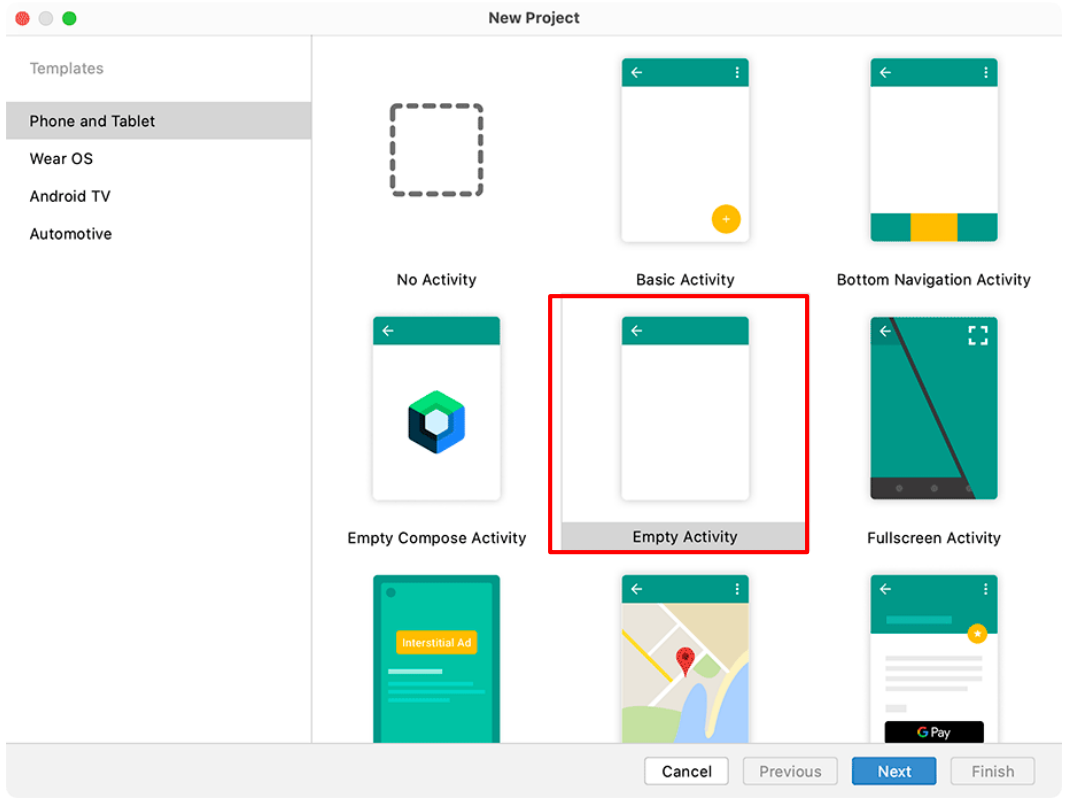
Ref: Android Nerd Ranch (5th edition), Chapter 9

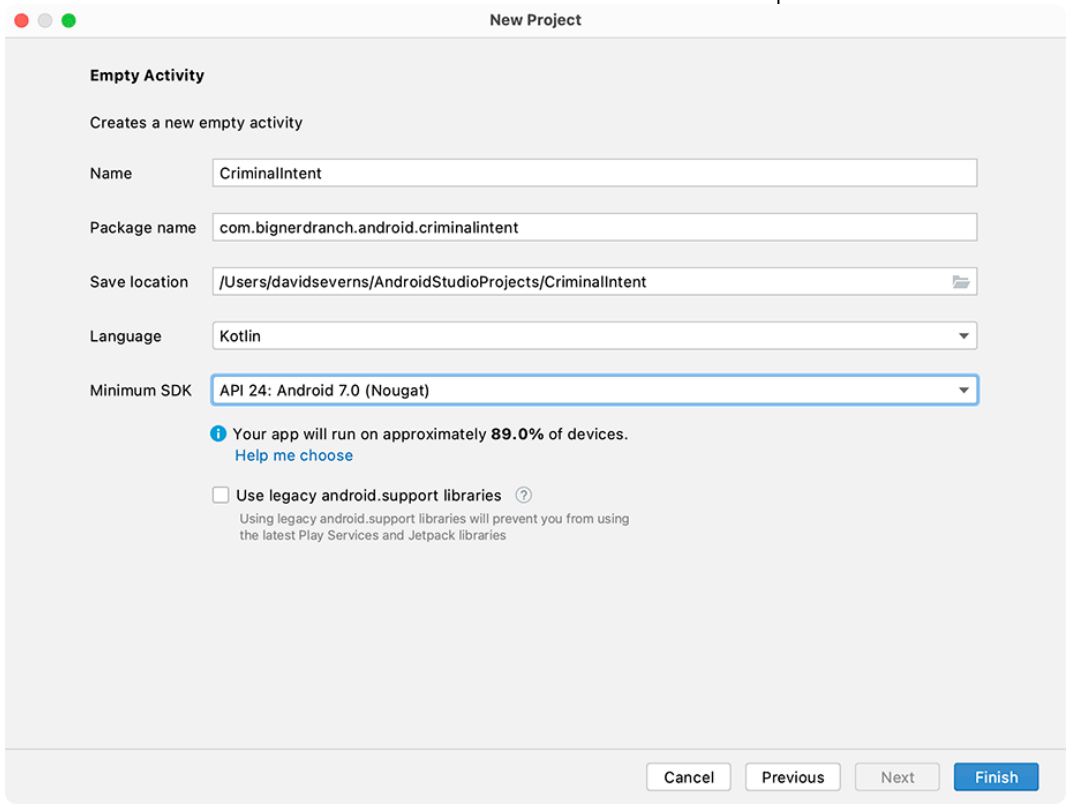
- **Crime:** holds record of 1 office crime. Has
  - **Title** e.g. “Someone stole my yogurt!”
  - **ID:** unique identifier of crime
- **CrimeDetailFragment:** UI fragment to display Crime Details
- **MainActivity:** Activity that contains **CrimeDetailFragment**



Create CriminalIntent in Android Studio

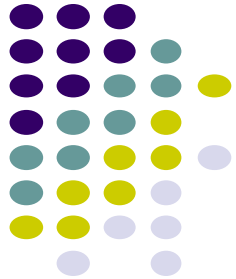
Ref: Android Nerd Ranch (5th edition), Chapter 9



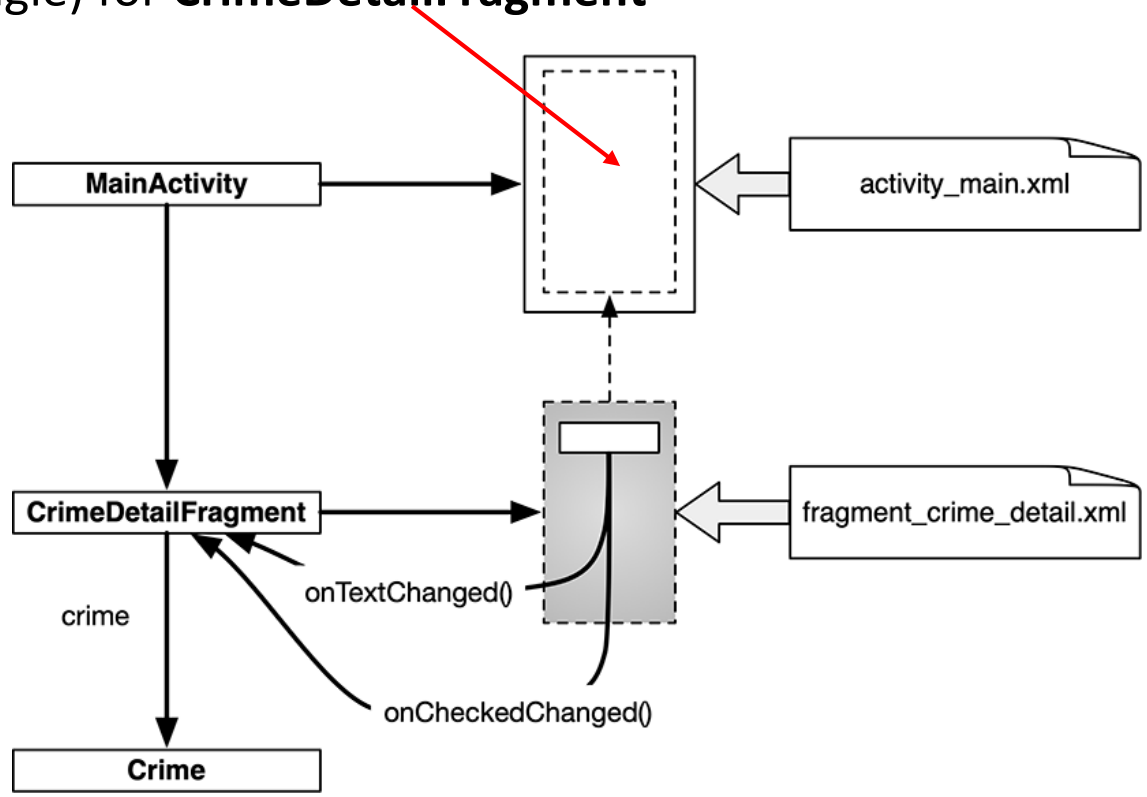


Fragment Hosted by an Activity

Ref: Android Nerd Ranch (5th edition), Chapter 9



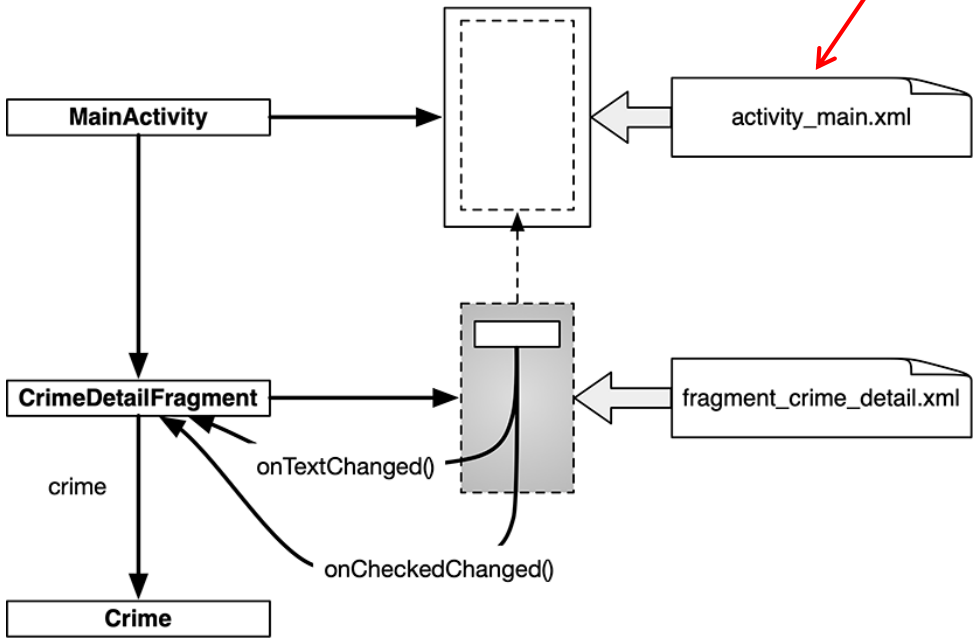
- Each fragment must be hosted by (contained inside) an Activity
- To host a UI fragment, an activity must
  - Define a spot (rectangle) in its layout for the fragment
  - Manage the lifecycle of the fragment instance (next)
- E.g.: **MainActivity** defines "spot" (rectangle) for **CrimeDetailFragment**



Hosting UI Fragment in an Activity

Ref: Android Nerd Ranch (5th edition), Chapter 9

- **FragmentContainerView** created in 2019 to make it easy for Activity to host Fragment
  - Use it in activity_main.xml to host **CrimeDetailFragment**
- First, create a spot (rectangle) for the fragment's view in **activity_crime.xml** XML layout



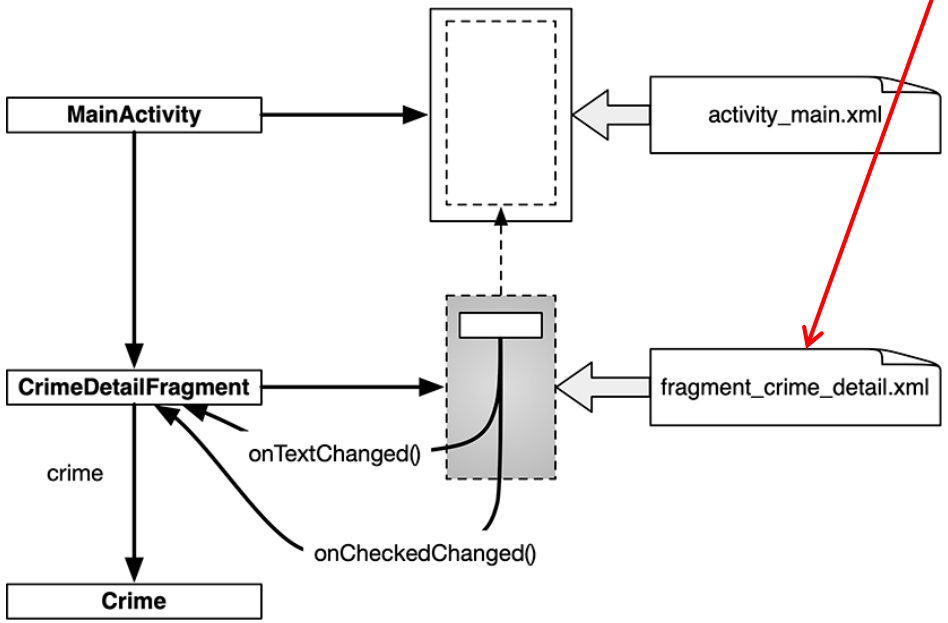

```
<androidx.fragment.app.FragmentContainerView
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/fragment_container"
    android:name="com.bignerdranch.android.criminalintent.CrimeDetailFragment"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity" />
```

Fragment as big as Parent (screen)

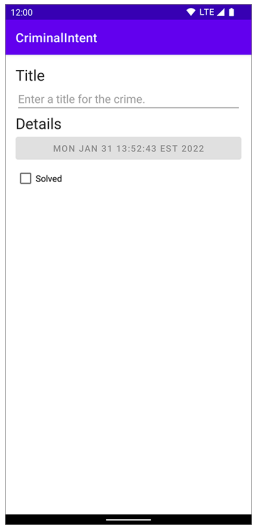
Creating a UI Fragment

Ref: Android Nerd Ranch (5th edition), Chapter 9

- Creating Fragment is similar to creating activity
  1. Compose UI by defining widgets in a layout (XML) file (same as before)
  2. Create kotlin class and specify layout file as XML file above
  3. Get references of inflated widgets in kotlin file (findviewbyId, view bindings), etc
- XML layout file for **CrimeDetailFragment (fragment_crime_detail.xml)**

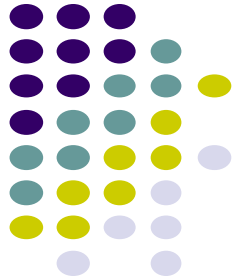



```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:orientation="vertical"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:layout_margin="16dp">

    <TextView
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:textAppearance="?attr/textAppearanceHeadline5"
        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"
        android:importantForAutofill="no"
        android:inputType="text" />

    <TextView
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:textAppearance="?attr/textAppearanceHeadline5"
        android:text="@string/crime_details_label" />

    <Button
        android:id="@+id/crime_date"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        tools:text="Wed May 11 11:56 EST 2022" />

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



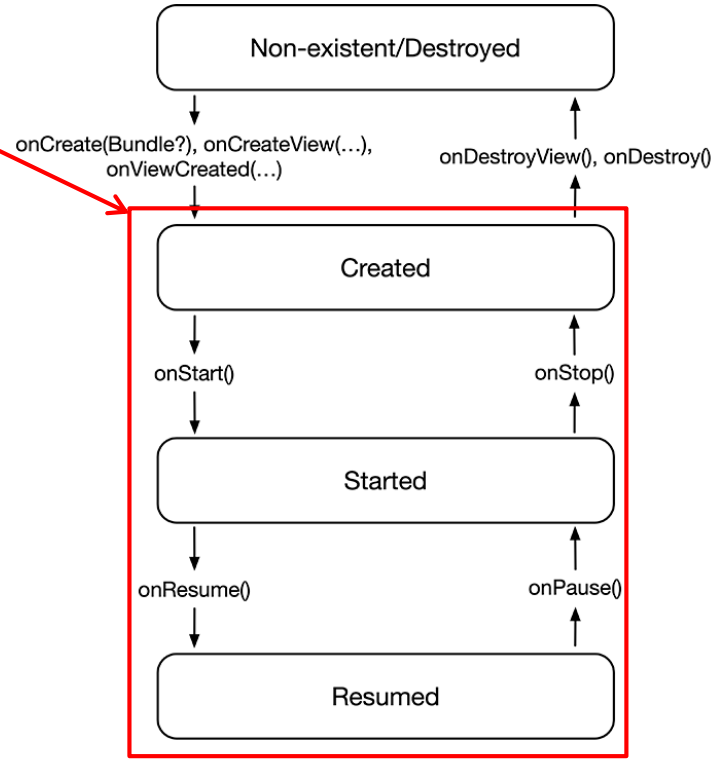
Fragment's Life Cycle

Ref: Android Nerd Ranch (5th edition), Chapter 9



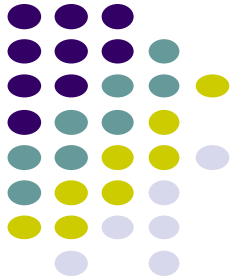
- Fragment's lifecycle similar to activity lifecycle
  - Has states **running**, **paused** and **stopped**
  - Also has some similar activity lifecycle methods (e.g. **onPause()**, **onStop( )**, etc)

- **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,** called by parent Activity



CrimeDetailFragment.kt kotlin File

Ref: Android Nerd Ranch (5th edition), Chapter 9



- Use Jetpack version of Fragment class (**androidx.fragment.app.Fragment**)
- In **CrimeDetailFragment** Override **onCreateView( )** function

Derive CrimeDetailFragment From Android's Fragment class

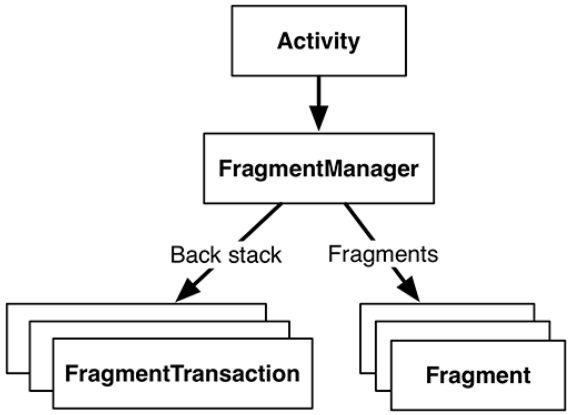
```
class CrimeDetailFragment : Fragment() {

    private lateinit var binding: FragmentCrimeDetailBinding

    private lateinit var crime: Crime

    override fun onCreate(savedInstanceState: Bundle?) {
        ...
    }

    override fun onCreateView(
        inflater: LayoutInflater,
        container: ViewGroup?,
        savedInstanceState: Bundle?
    ): View? {
        binding =
            FragmentCrimeDetailBinding.inflate(layoutInflater, container, false)
        return binding.root
    }
}
```

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

Adding UI Fragment to FragmentManager

Ref: Android Nerd Ranch (5th edition), Chapter 9

- 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

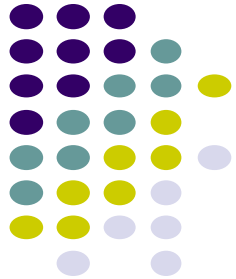


- **FragmentContainerView** interacts with **FragmentManager** to display **CrimeDetailFragment**
- **FragmentManager** interations uses transactions
- **E.g.** If a fragment is swapped out, would want replacement with a new fragment in one transaction.

Creates and returns instance of FragmentTransaction

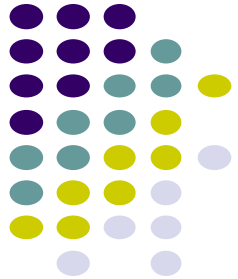
Add recently created CrimeDetailFragment

```
val fragment = CrimeDetailFragment()
supportFragmentManager
    .beginTransaction()
    .add(R.id.fragment_container, fragment)
    .commit()
```



Android Nerd Ranch CriminalIntent Chapters Skipped

Chapter 8: Android SDK versions and Compatibility



- Skipped several details and **UI chapters**
- Also minimum SDK choice, etc.

Marketing name	Version number	Version code	API level
Android Nougat	7.0	N	24
Android Nougat	7.1 – 7.1.2	N_MR1	25
Android Oreo	8.0	O	26
Android Oreo	8.1.0	O_MR1	27
Android Pie	9	P	28
Android 10	10	Q	29
Android 11	11	R	30
Android 12	12	S	31
Android 12L	12	Sv2	32

Chapter 10: Displaying Lists with RecyclerView

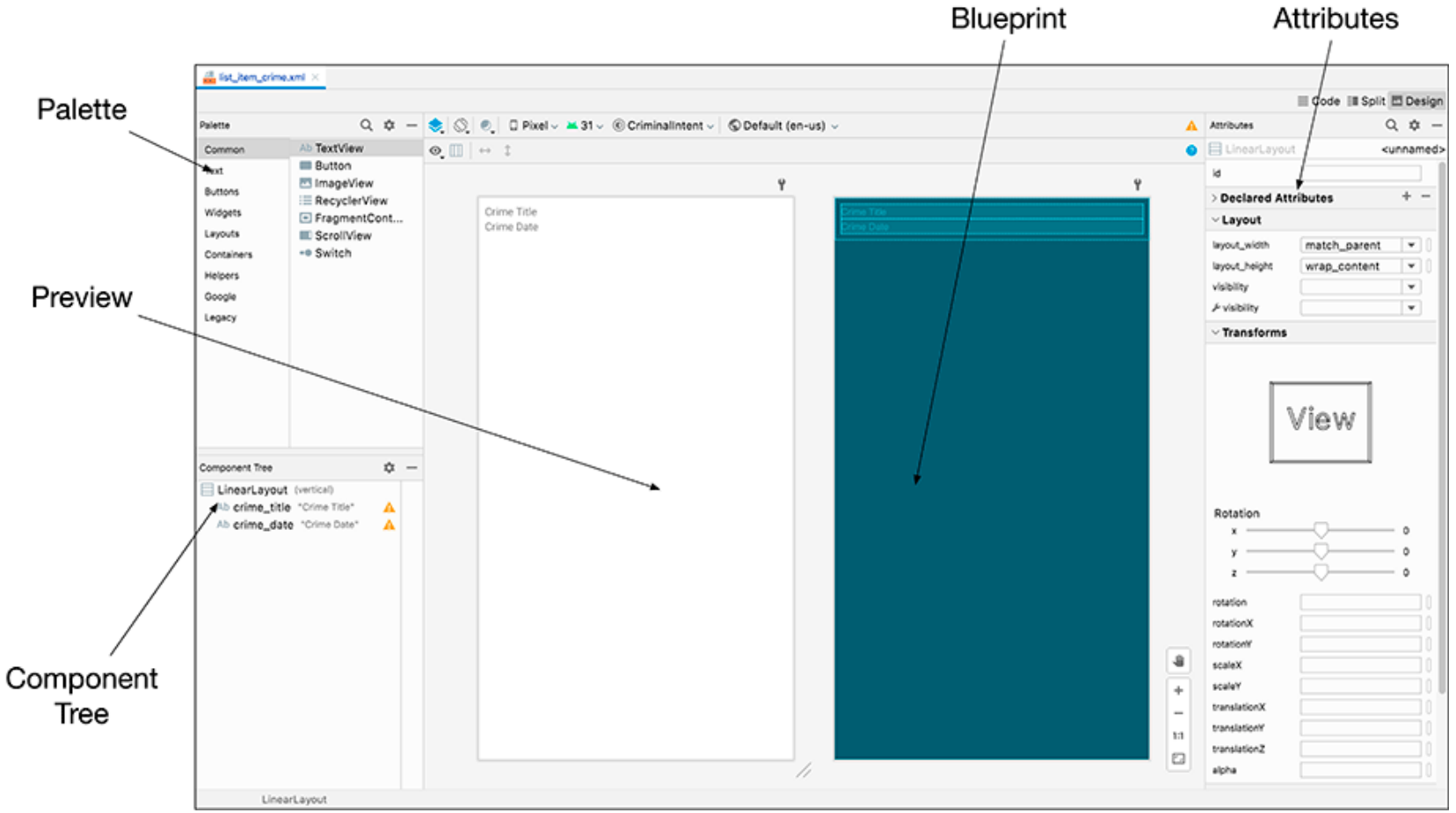
- Skipped some **UI chapters**

- RecyclerView facilitates view of large dataset

- E.g. Allows list of crimes (title, date) in **CriminalIntent**

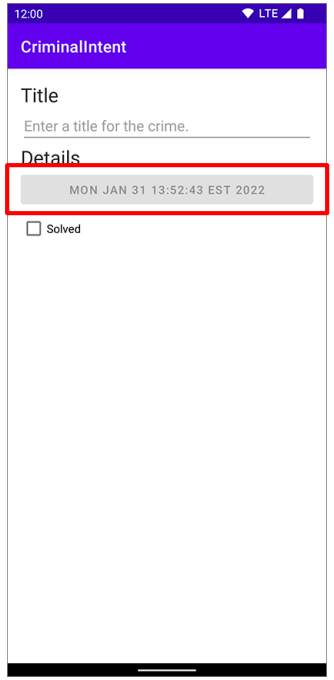
Chapter 11: Creating UI with Layouts and Views

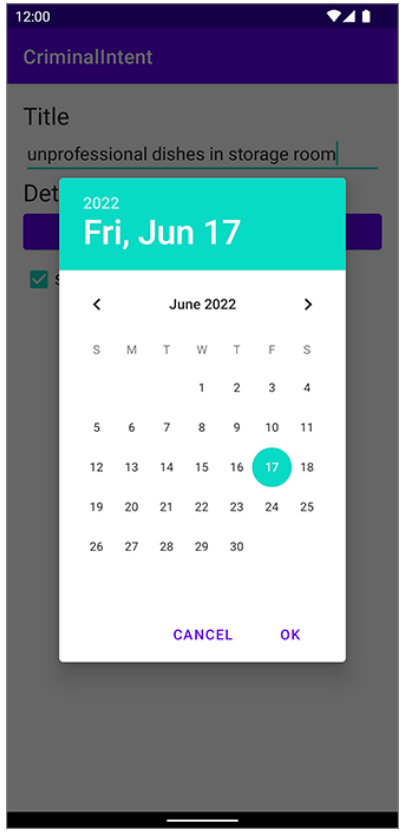
- Mostly already covered (ImageView, etc)
- Describes Contraint Layout (specify widget positions using constraints)



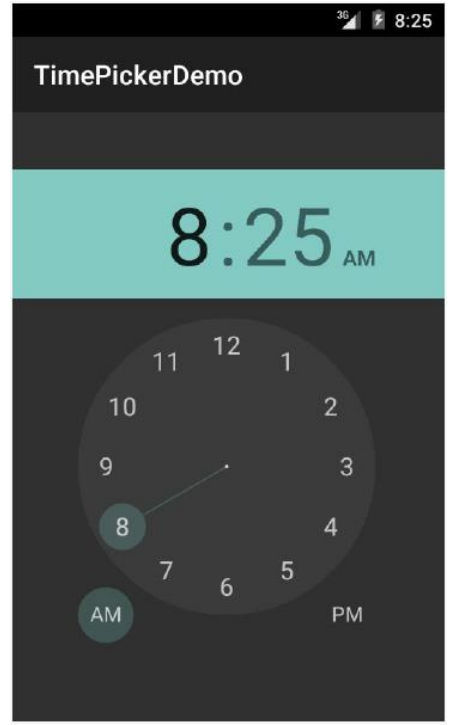
Chapter 14: Dialogs

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





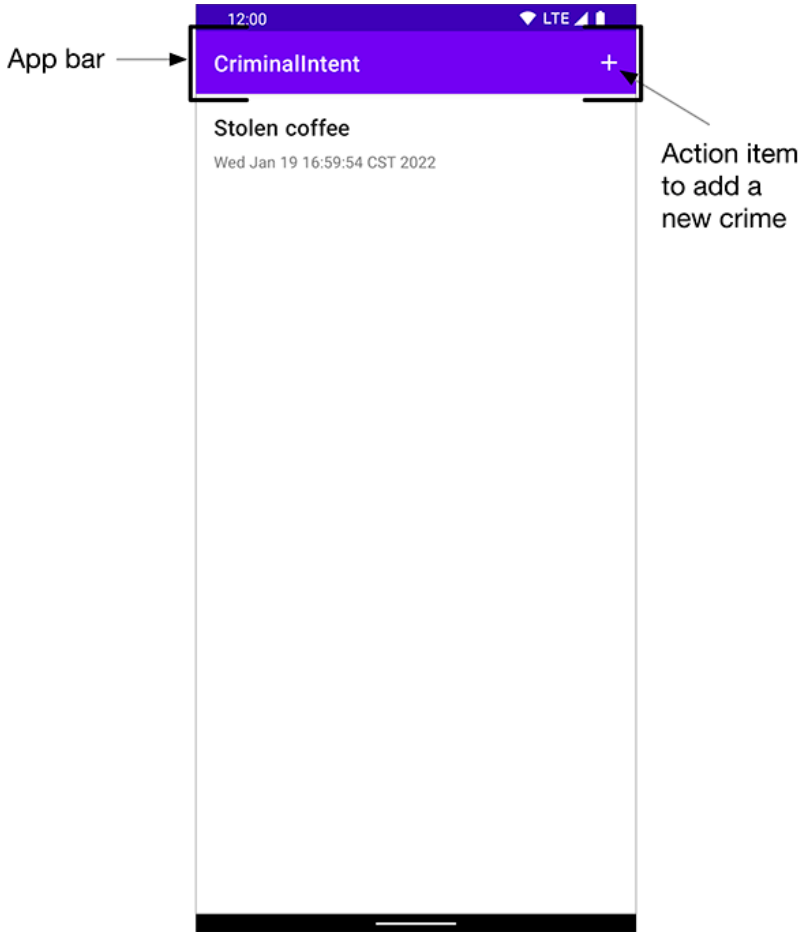
DatePicker



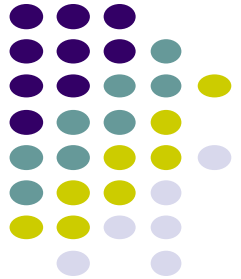
TimePicker also exists

Chapter 15: The Toolbar

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



References



- Android Nerd Ranch (5th edition)