CS 4518 Mobile and Ubiquitous Computing

Lecture 5: Rotating Device, Saving Data, Intents and Fragments

Emmanuel Agu

Administrivia

- Moved back deadlines for projects 2, 3 and final project
  - See updated schedule on class website
- Project 2 email out tonight, can be done on own computer
  - Submit source code + video of your app
  - Zoolab submission issues.
    - E.g. Projects done on Mac generated errors in zoolab
- Project teams: list of teams will be email out tonight
- Final project specs/ground rules out on Monday

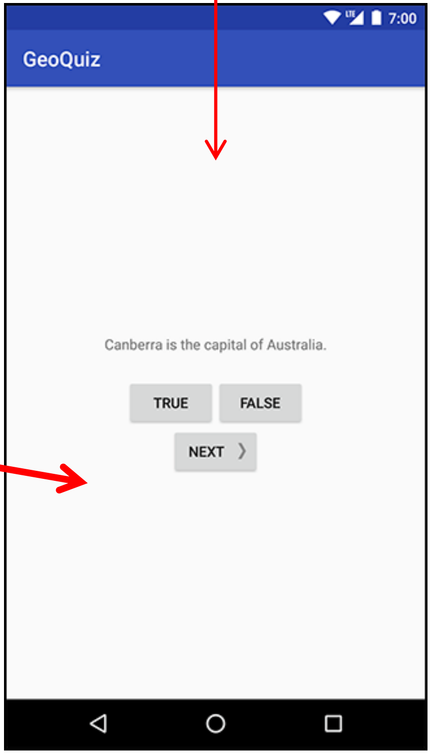
Rotating Device

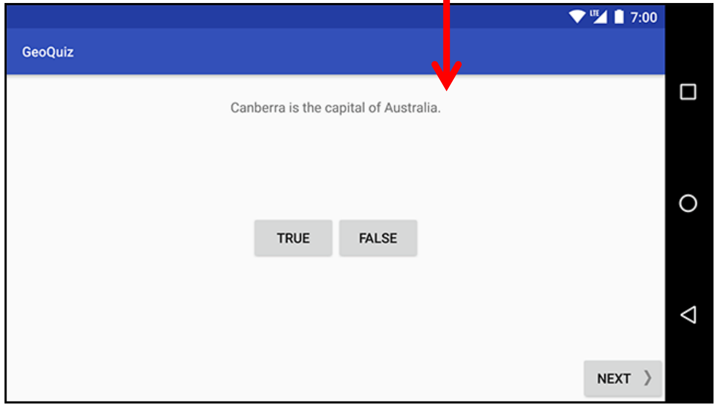
Rotating Device: Using Different Layouts

- Rotating device (e.g. portrait to landscape) kills current activity and creates new activity in landscape mode
- Rotation changes **device configuration**
- **Device configuration**: screen orientation/density/size, keyboard type, dock mode, language, etc.
- Apps can specify different resources (e.g. XML layout files, images) to use for different device configurations

- E.g. use different app layouts for portrait vs landscape screen orientation

Use portrait XML file



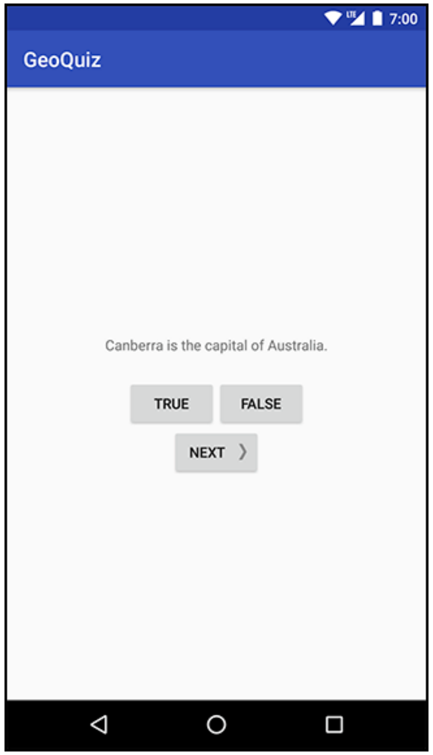


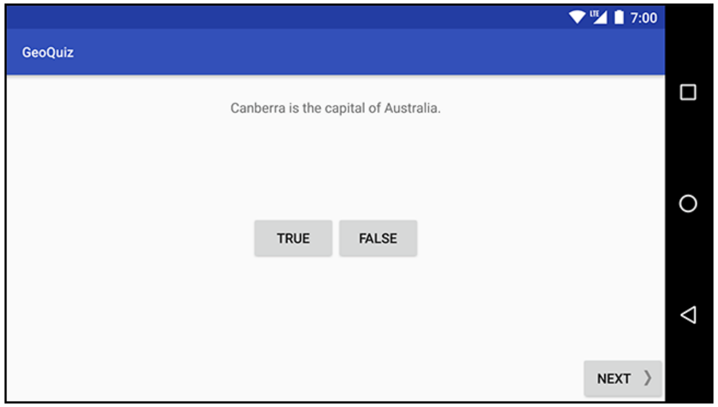
Use landscape XML file

Rotating Device: Using Different Layouts

- Portrait device: use XML layout file in **res/layout**
- Landscape device: use XML layout file in **res/layout-land/**
- Copy XML layout file (activity_quiz.xml) from **res/layout** to **res/layout-land/** and tailor it
- If configuration changes, current activity destroyed, **onCreate -> setContentView (R.layout.activity_quiz)** called again

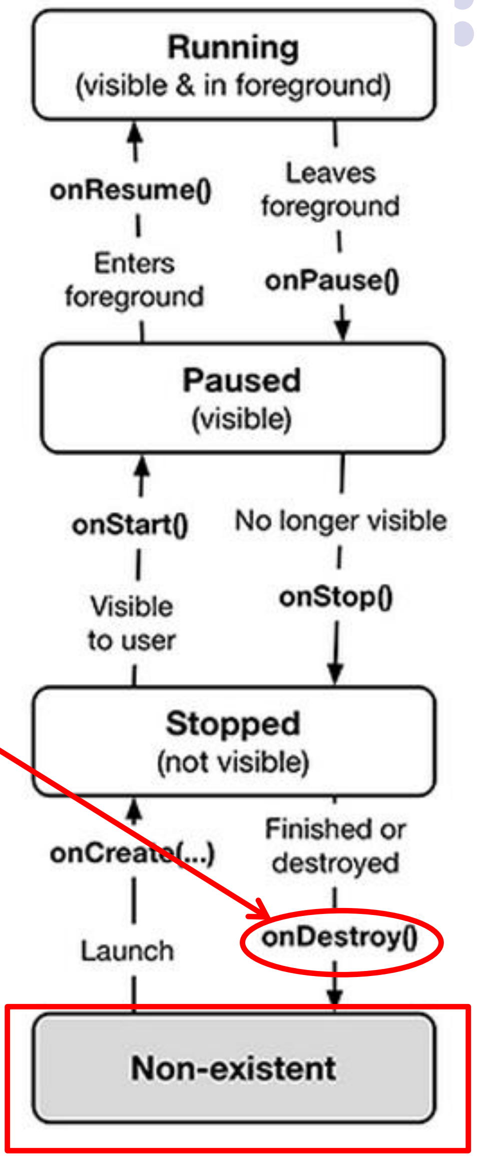
onCreate called whenever user switches between portrait and landscape





Dead or Destroyed Activity

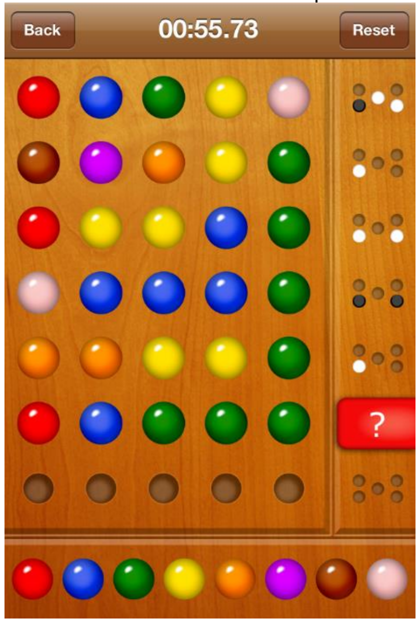
- onDestroy( ) called to destroy a stopped app

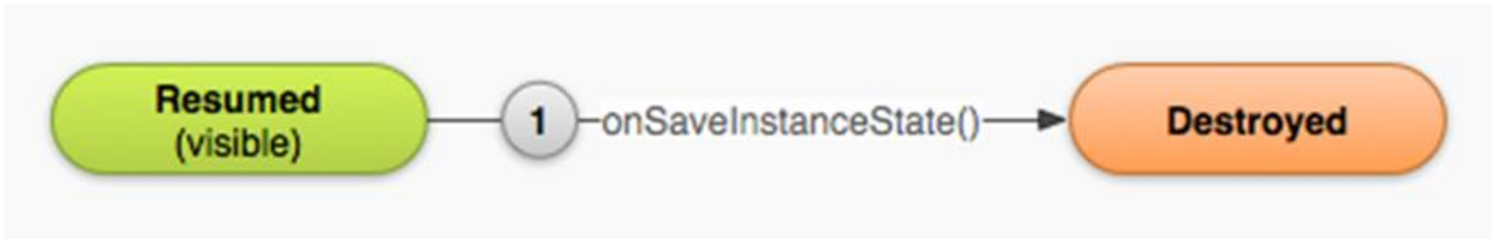


Saving State Data

Activity Destruction

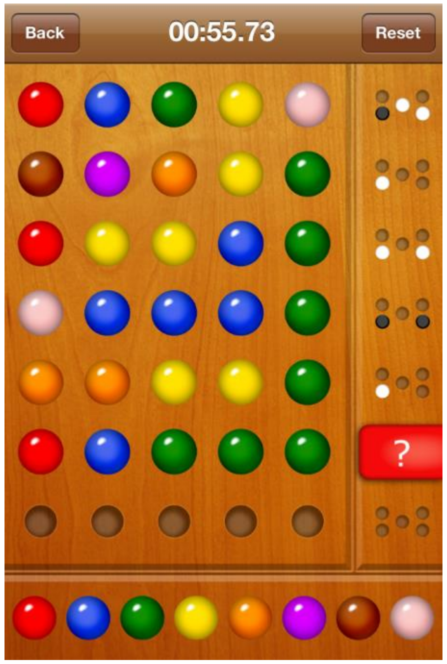
- App may be destroyed
  - On its own by calling finish
  - If user presses **back button**
- Before Activity destroyed, system calls **onSaveInstanceState**
- Saves state required to recreate Activity later
  - E.g. Save current positions of game pieces





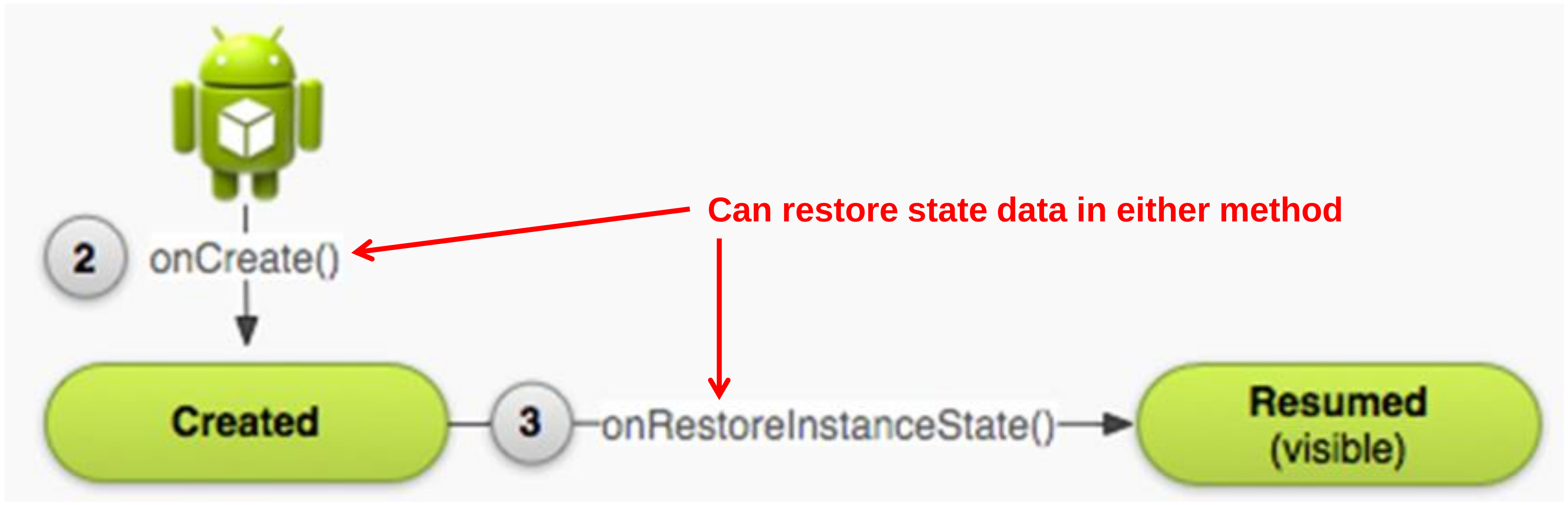
onSaveInstanceState: Saving App State

- Systems write info about views to Bundle
- Programmer must save other app-specific information using **onSaveInstanceState( )**
  - E.g. board state in a board game such as mastermind



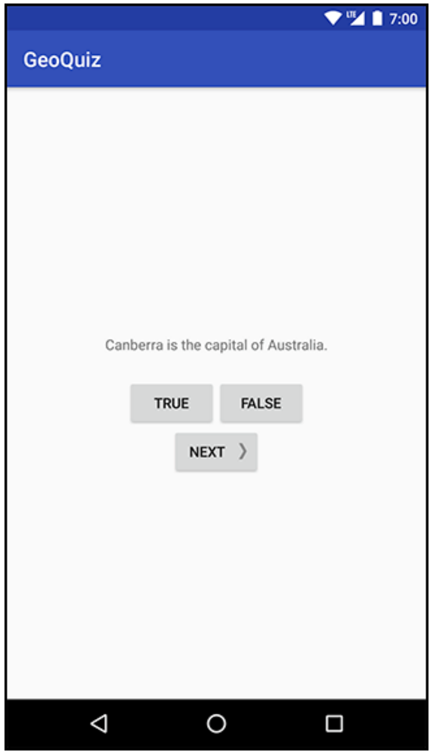
onRestoreInstanceState( ): Restoring State Data

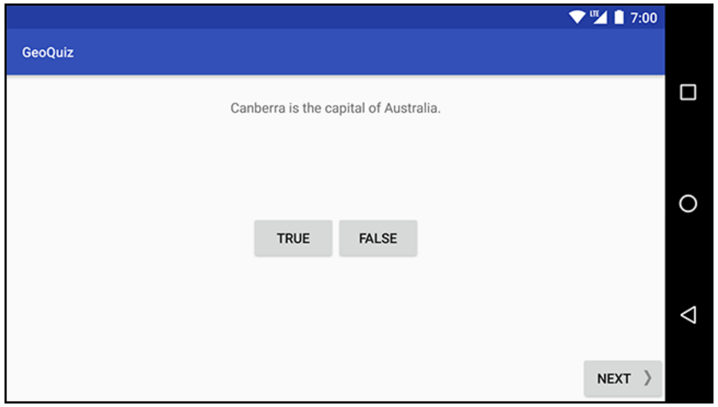
- When an Activity recreated Bundle sent to **onCreate** and **onRestoreInstanceState()**
- Can use either method to restore app state data



Saving Data Across Device Rotation

- Since rotation causes activity to be destroyed and new one created, values of variables lost or reset
- To avoid losing or resetting values, save them using **onSaveInstanceState** before activity is destroyed
    - E.g. called before portrait layout is destroyed
- System calls **onSaveInstanceState** before **onPause( )**, **onStop( )** and **onDestroy( )**





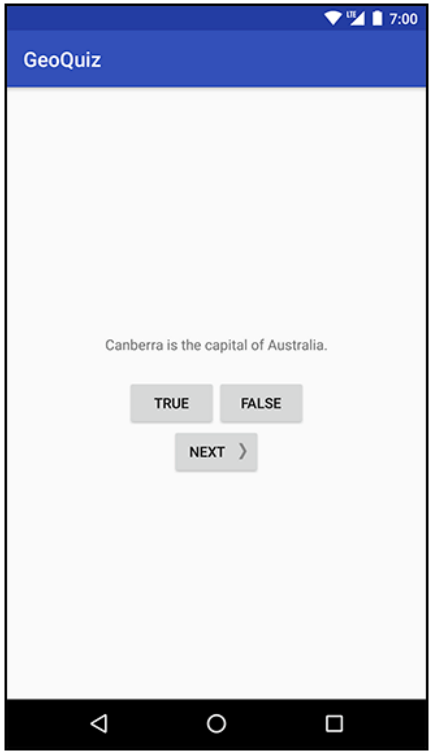
Saving Data Across Device Rotation

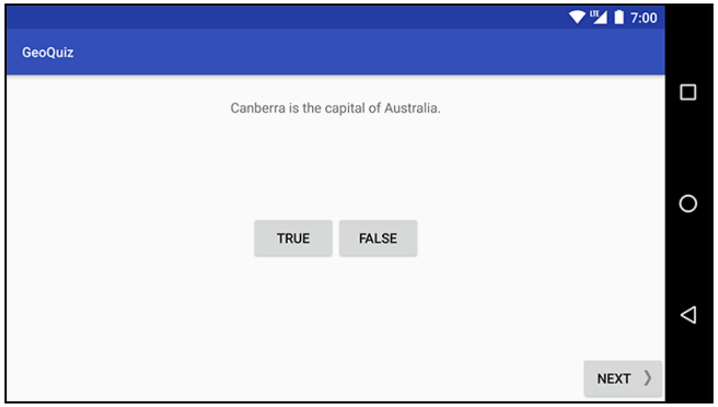
- For example, to save the value of a variable **mCurrentIndex** during rotation
- First, create a constant KEY_INDEX as a key for storing data in the bundle

```
private static final String KEY_INDEX = "index";
```

- Then override **onSaveInstanceState** method

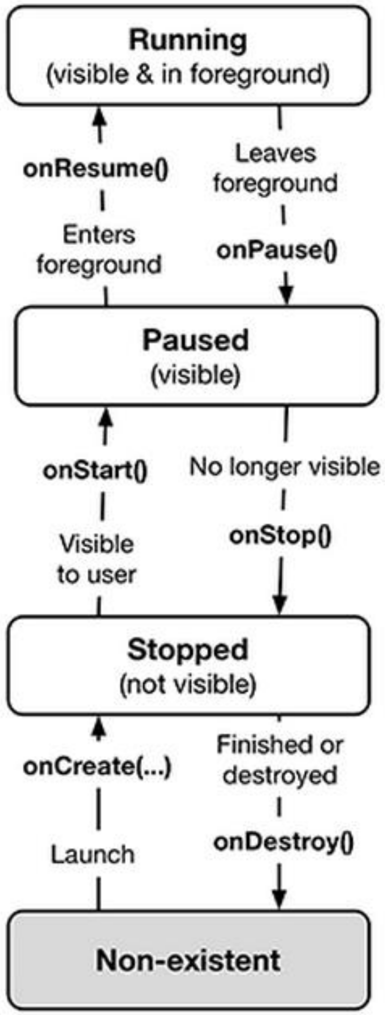
```
@Override
public void onSaveInstanceState(Bundle savedInstanceState) {
    super.onSaveInstanceState(savedInstanceState);
    Log.i(TAG, "onSaveInstanceState");
    savedInstanceState.putInt(KEY_INDEX, mCurrentIndex);
}
```





Question

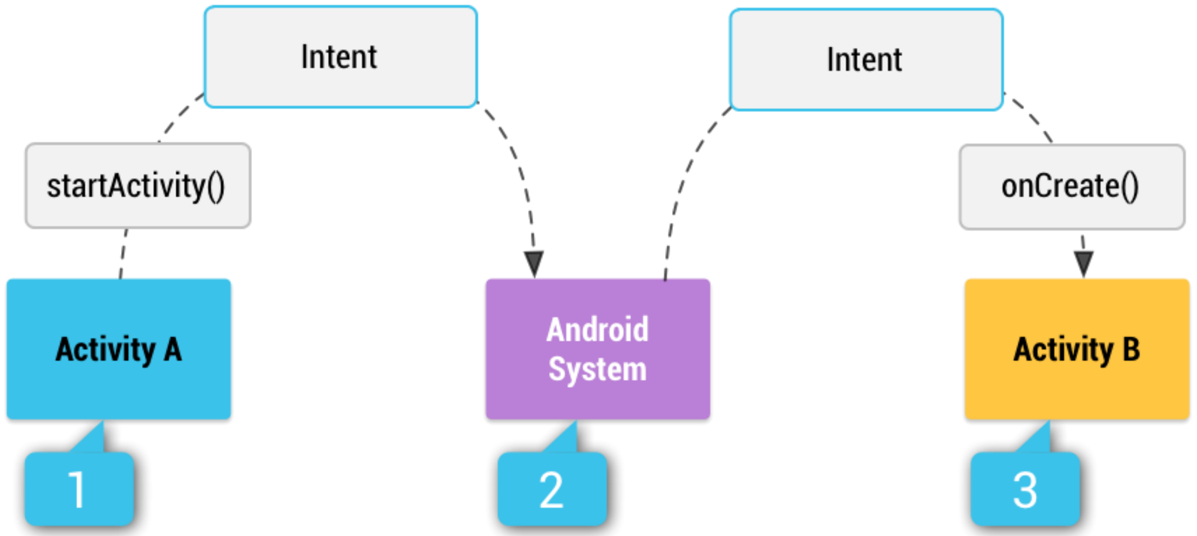
- Whenever I watch YouTube video on my phone, if I receive a phone call and video stops at 2:31, after call, when app resumes, it should restart at 2:31.
- How do you think this is implemented?
  - In which Android methods should code be put into?
  - How?



Intents

Intent

- **Intent:** a messaging object used by a component to request action from another app or component
- 3 main use cases for Intents
- **Case 1 (Activity A starts Activity B, no result back):**
  - Call **startActivity( )**, pass an Intent
  - Intent describes Activity to start, plus any necessary data



Intent: Result Received Back

- **Case 2 (Activity A starts Activity B, gets result back):**
  - Call **startActivityForResult( )**, pass an Intent
  - Separate Intent received in Activity A's **onActivityResult( )** callback

- **Case 3 (Activity A starts a Service):**
  - E.g. Activity A starts service to download big file in the background
  - Activity A calls **StartService( )**, passes an Intent
  - Intent describes Service to start, plus any necessary data

Implicit Vs Explicit Intents

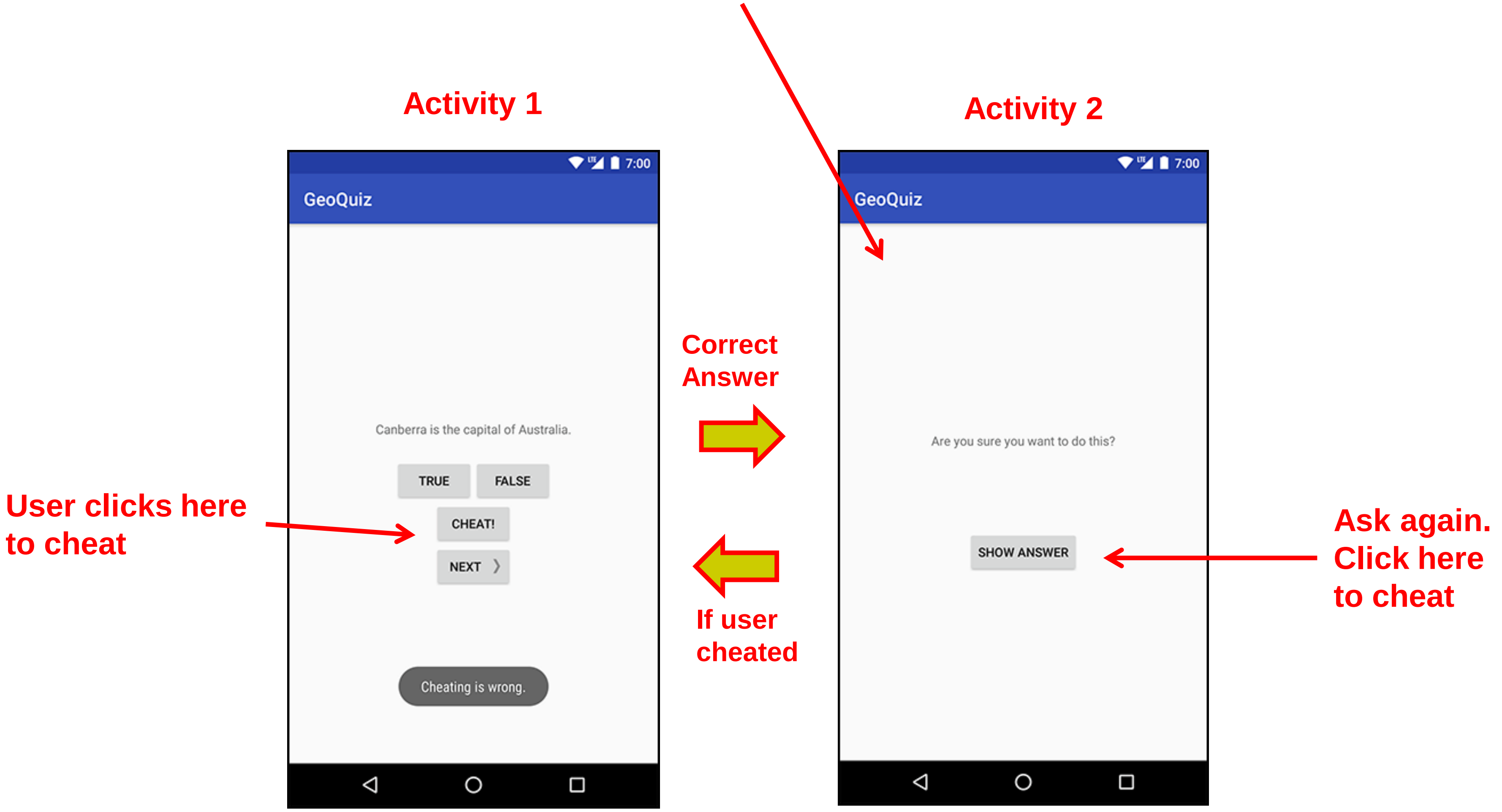
- **Explicit Intent:** If components sending and receiving Intent are in same app
  - E.g. Activity A starts Activity B in same app
  - Activity A explicitly says what Activity (B) that should be started

- **Implicit Intent:** If components sending and receiving Intent are in **different apps**
  - Activity B specifies what ACTION it needs done, doesn't specify Activity to do it
  - Example of Action: take a picture, any camera app can handle this

Intent Example: Starting Activity 2 from Activity 1

Allowing User to Cheat
Ref: Android Nerd Ranch (3rd edition) pg 91

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

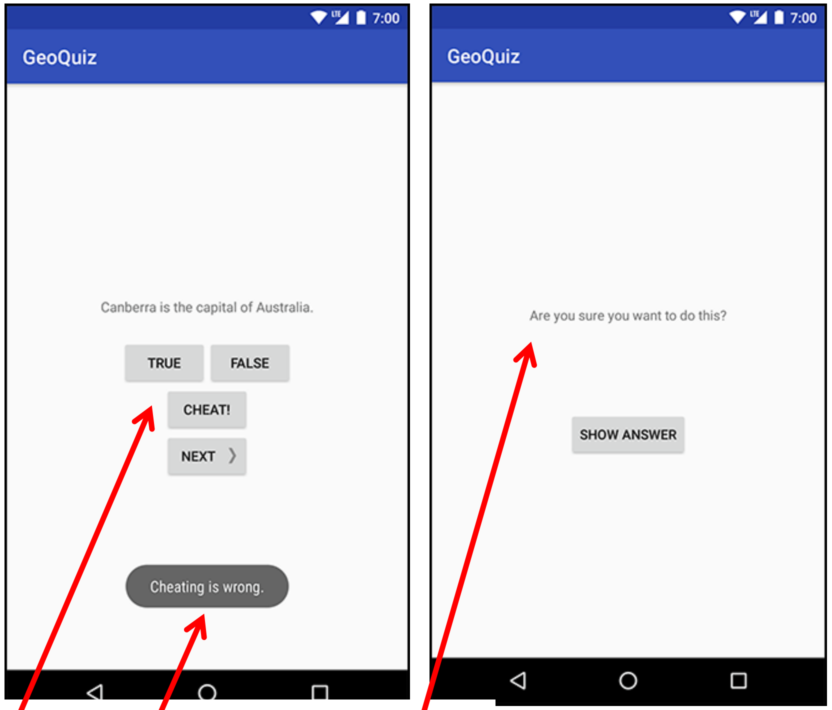


Canberra is the capital of Australia.

Cheating is wrong.

Are you sure you want to do this?

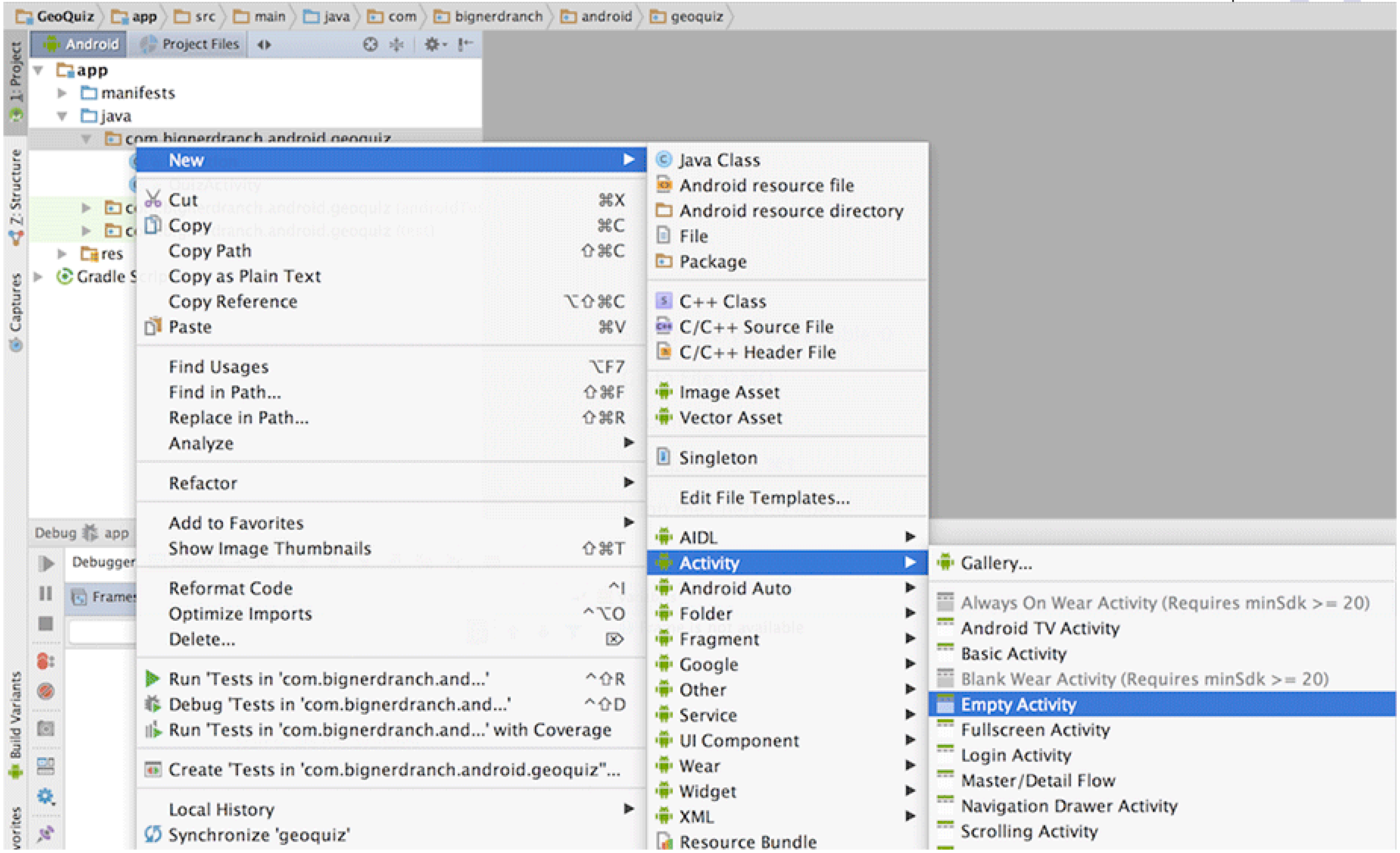
Add Strings for Activity 1 and Activity 2 to strings.xml



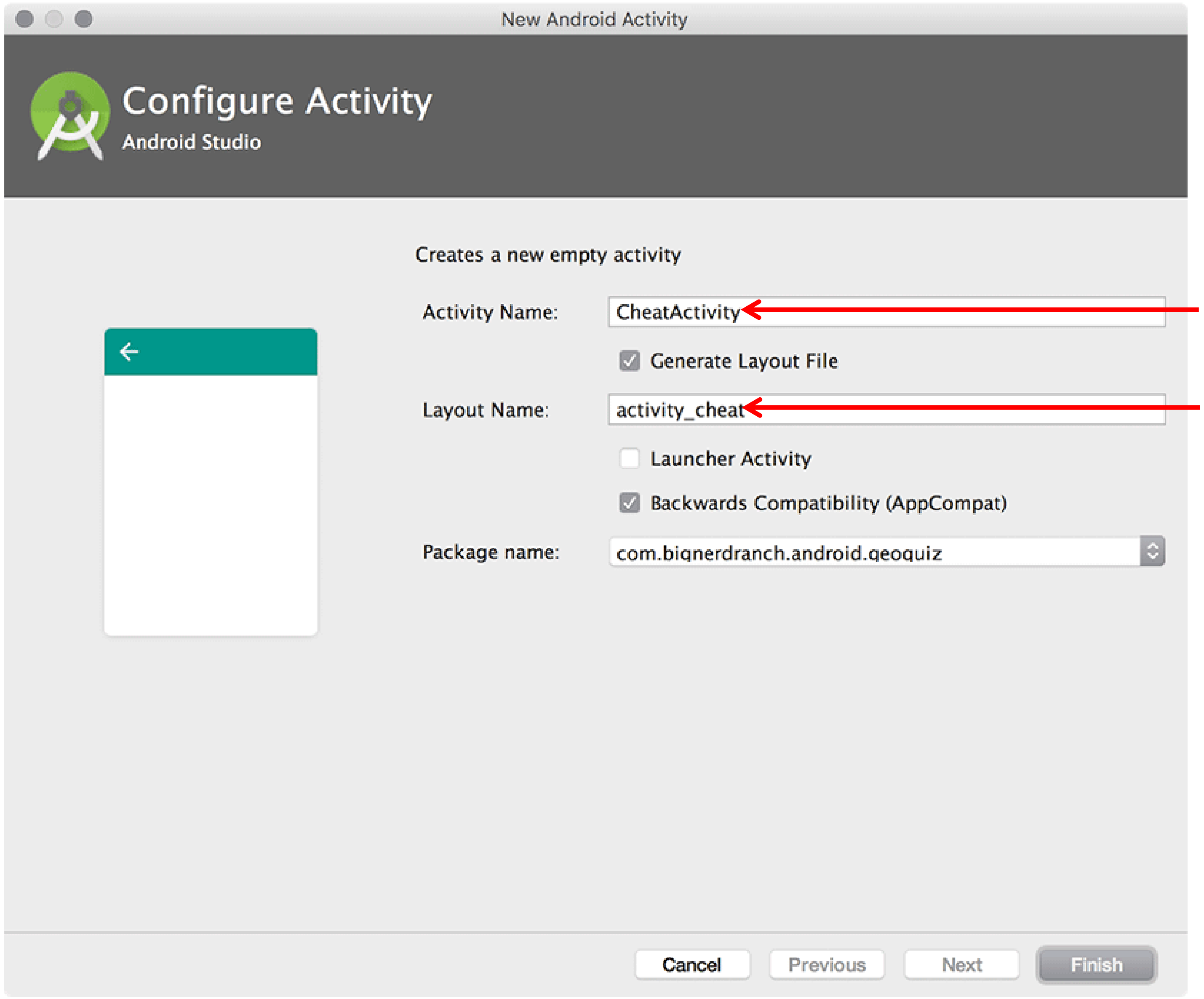
GeoQuiz
Canberra is the capital of Australia.
TRUE
FALSE
CHEAT!
NEXT
Cheating is wrong.
GeoQuiz
Are you sure you want to do this?
SHOW ANSWER

```
<?xml version="1.0" encoding="utf-8"?>
<resources>

    ...
    <string name="question_asia">Lake Baikal is the world\'s oldest and
deepest
      freshwater lake.</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) in Android Studio



Specify Name and XML file for Activity 2



New Android Activity

Configure Activity
Android Studio

Creates a new empty activity

Activity Name: CheatActivity

[✓] Generate Layout File

Layout Name: activity_cheat

[ ] Launcher Activity

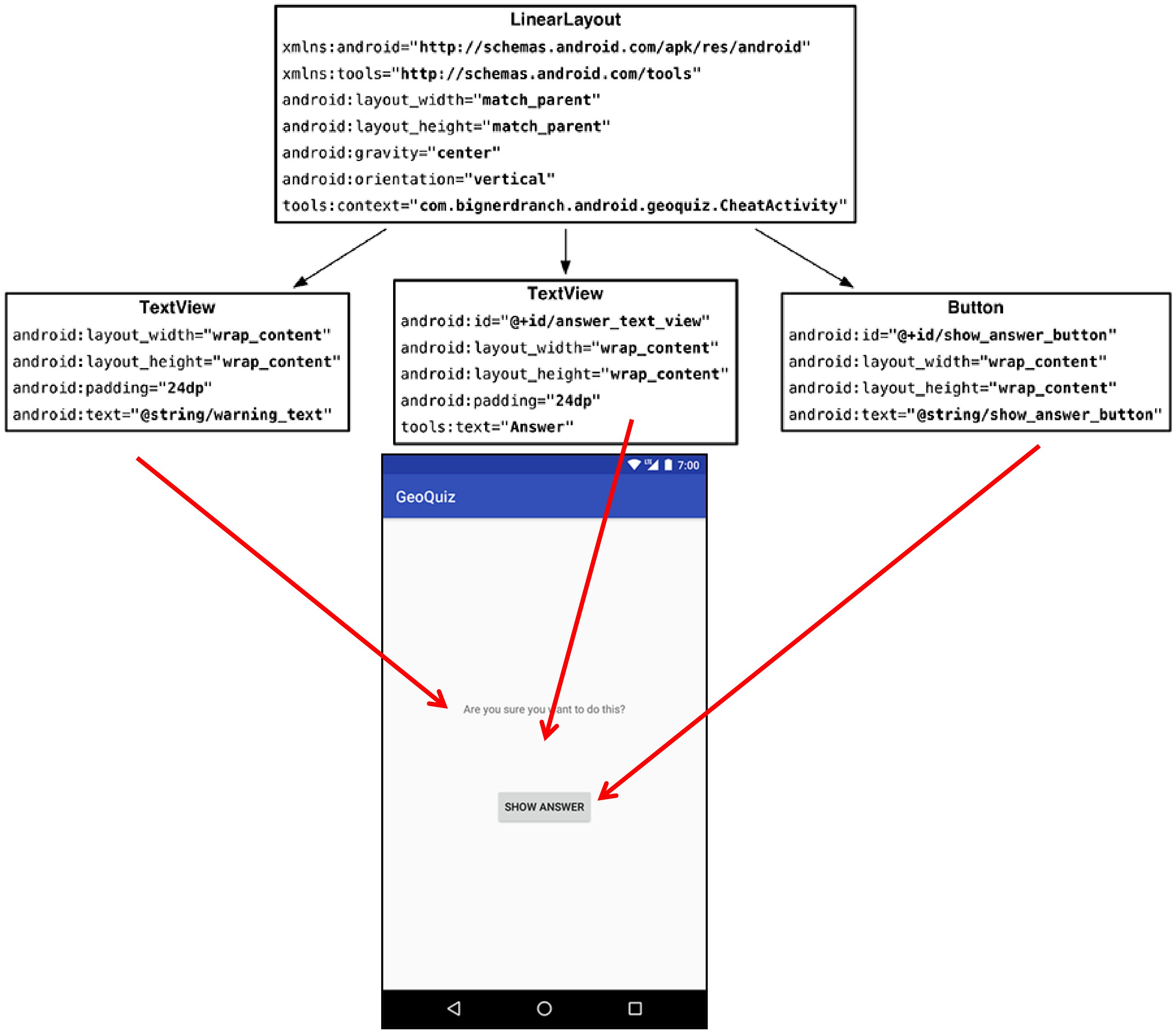
[✓] Backwards Compatibility (AppCompat)

Package name: com.bignerdranch.android.geoquiz

Screen 2 Java code in CheatActivity.java

Layout uses activity_cheat.xml

Design Layout for Screen 2



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"
```

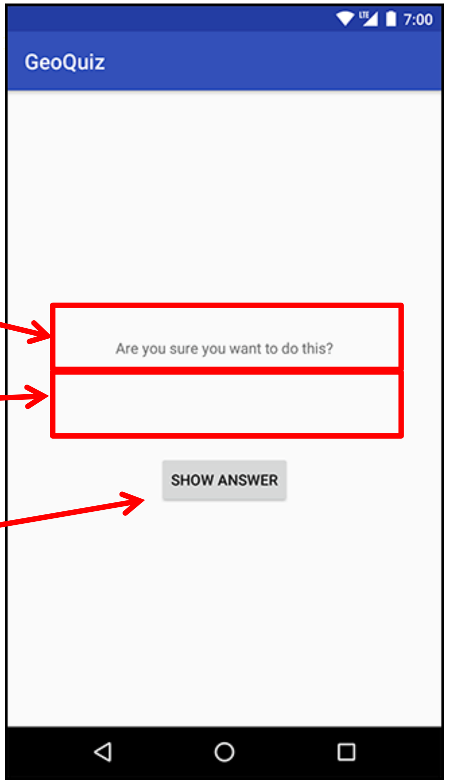
TextView
```
android:id="@+id/answer_text_view"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:padding="24dp"
tools:text="Answer"
```

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

Write XML Layout Code for Screen 2

```
<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:orientation="vertical"
              android:gravity="center"
              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>
```

Activity 2



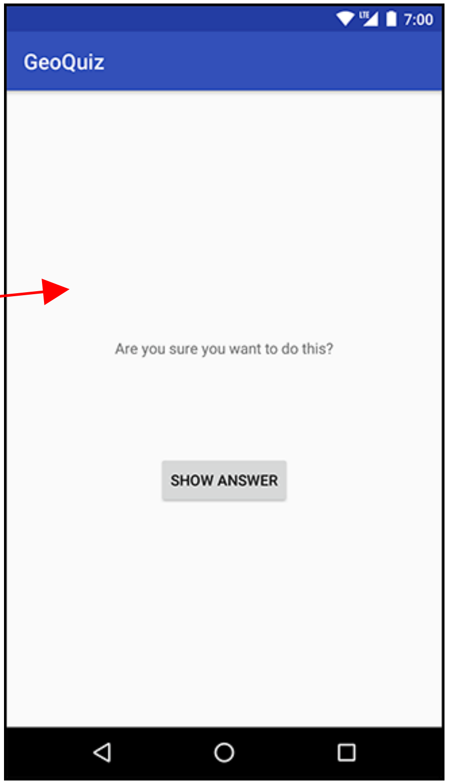
Declare New Activity (CheatActivity) in AndroidManifest.xml

```
<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:supportsRtl="true"
        android:theme="@style/AppTheme">

        <activity android:name=".QuizActivity">
            <intent-filter>
                <action android:name="android.intent.action.MAIN"/>

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

</manifest>
```

Activity 1

Activity 2 (CheatActivity)

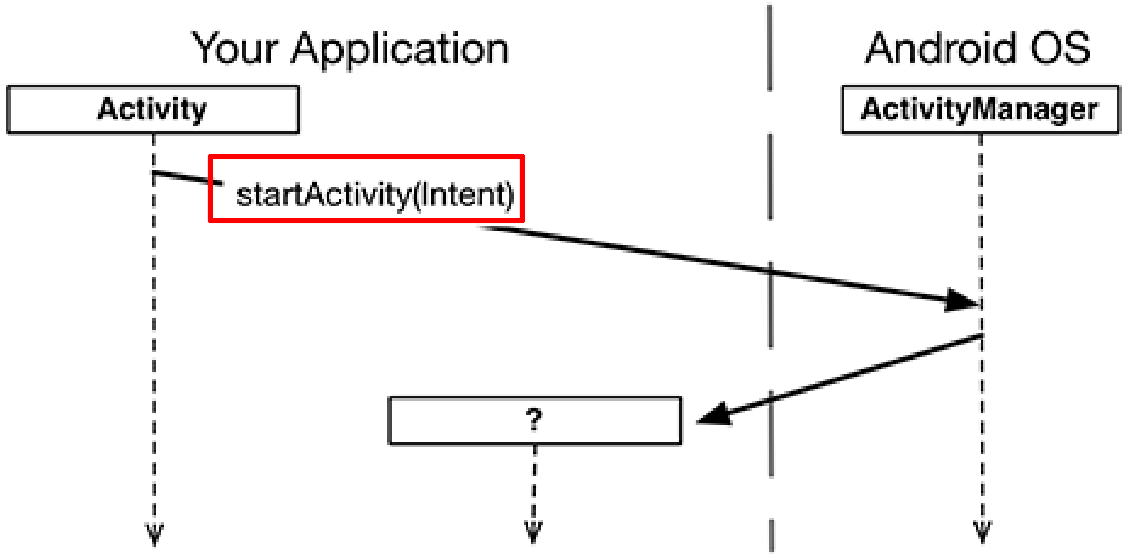
Activity 2 (CheatActivity)



Are you sure you want to do this?

Starting Activity 2 from Activity 1

- Activity 1 starts activity 2
  - **through** the Android OS
  - by calling **startActivity(Intent)**
- Passes Intent (object for communicating with Android OS)

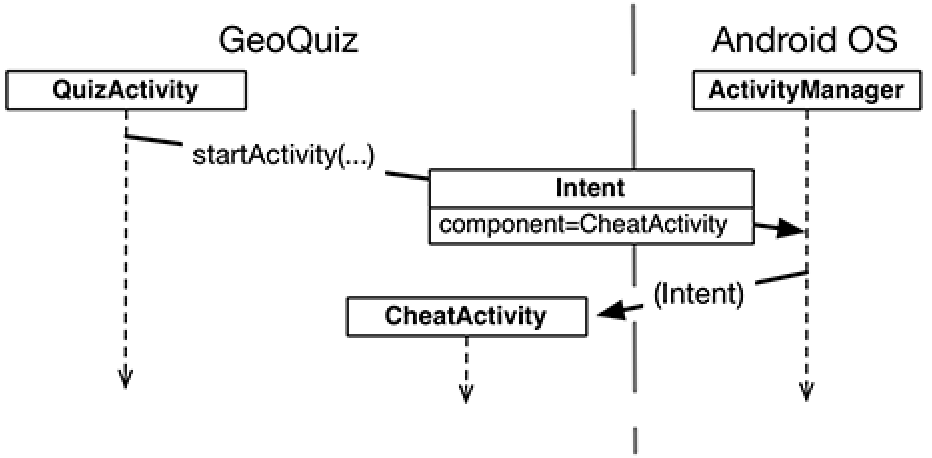


- Intent specifies which (target) Activity Android ActivityManager should start

Starting Activity 2 from Activity 1

- Intents have many different constructors. We will use form:

```
public Intent(Context packageContext, Class<?> cls)
```



- Actual code looks like this

```
mCheatButton = (Button)findViewById(R.id.cheat_button);
mCheatButton.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        // Start CheatActivity
        Intent intent = new Intent(QuizActivity.this, CheatActivity.class);
        startActivity(intent);
    }
});
```

Build Intent

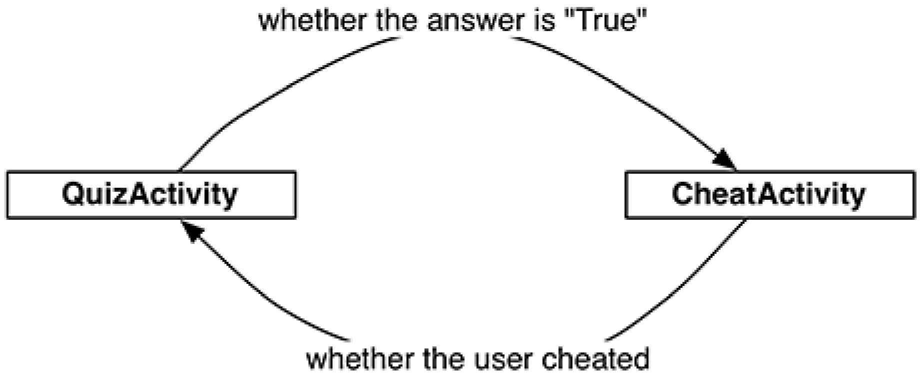
Use Intent to Start new Activity

Parent Activity

New Activity 2

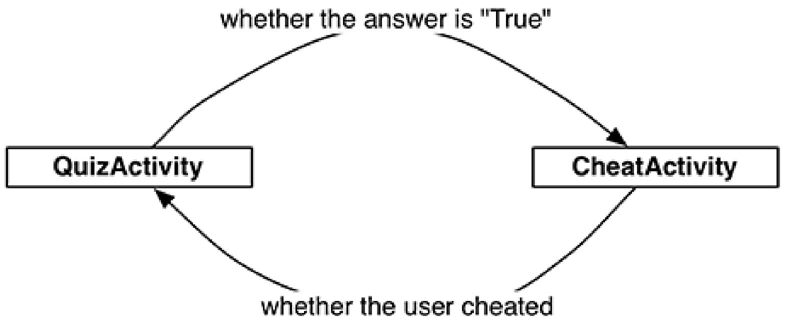
Implicit vs Explicit Intents

- Previous example is called an **explicit intent**
  - Activity 1 and activity 2 are in same app
- If Activity 2 were in another app, an **implicit intent** would have to be created instead
- Can also pass data between Activities 1 and 2
  - E.g. Activity 1 can tell Activity 2 correct answer (True/False)

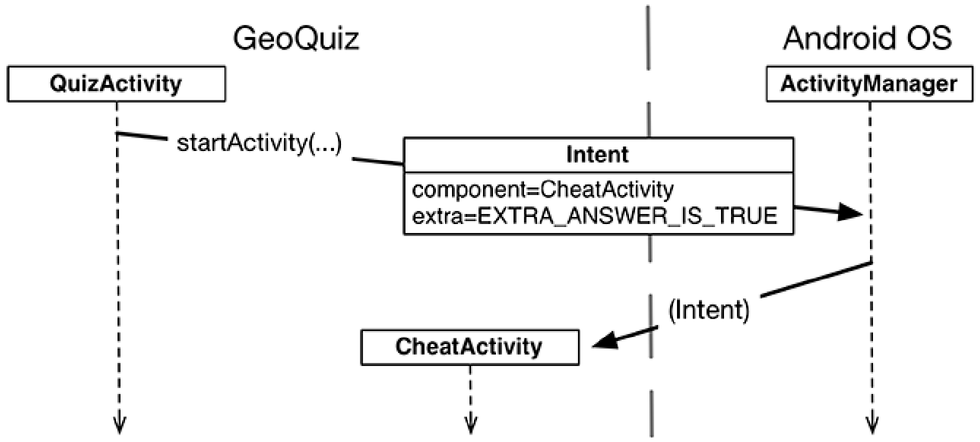


Passing Data Between Activities

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



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



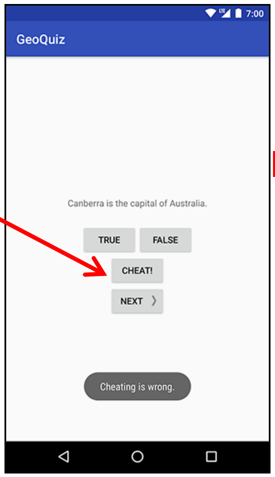
Passing Answer (True/False) as Intent Extra

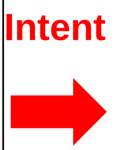
- To add **extra** to Intent, use **putExtra( )** command
- Encapsulate Intent creation into a method **newIntent( )**

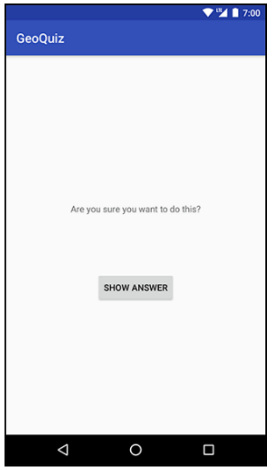
```
public class CheatActivity extends AppCompatActivity {

    private static final String EXTRA_ANSWER_IS_TRUE =
            "com.bignerdranch.android.geoquiz.answer_is_true";

    public static Intent newIntent(Context packageContext, boolean answerIsTrue) {
        Intent intent = new Intent(packageContext, CheatActivity.class);
        intent.putExtra(EXTRA_ANSWER_IS_TRUE, answerIsTrue);
        return intent;
    }
    ...
```

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

```
mCheatButton.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        // Start CheatActivity
        ~~Intent intent = new Intent(QuizActivity.this, CheatActivity.class);~~
        boolean answerIsTrue = mQuestionBank[mCurrentIndex].isAnswerTrue();
        Intent intent = CheatActivity.newIntent(QuizActivity.this, answerIsTrue);
        startActivity(intent);
    }
});
```







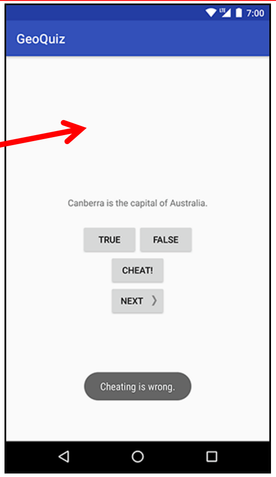
Passing Answer (True/False) as Intent Extra

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

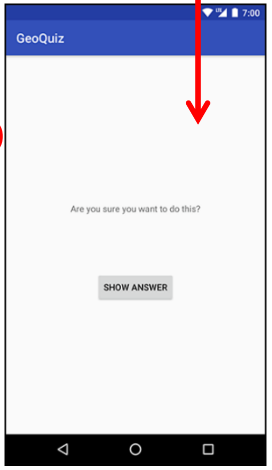
```
public class CheatActivity extends AppCompatActivity {

    private static final String EXTRA_ANSWER_IS_TRUE =
            "com.bignerdranch.android.geoquiz.answer_is_true";

    private boolean mAnswerIsTrue;
    ...
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_cheat);

        mAnswerIsTrue = getIntent().getBooleanExtra(EXTRA_ANSWER_IS_TRUE, false);
    }
    ...
}
```

Calls getIntent( )

Calls startActivity(Intent)



Intent (Answer = Extra)



Important: Read Android Nerd Ranch (3rd edition) pg 91

Implicit Intents

- **Implicit Intent:** Does not name component to start.
- Specifies
  - **Action** (what to do, example visit a web page)
  - **Data** (to perform operation on, e.g. web page url)
- Typically, many components (apps) can take a given action
  - E.g. Many phones have installed multiple apps that can view images
- System decides component to receive intent based on **action**, **data, category**
- Example Implicit Intent to share data

```java
// Create the text message with a string
Intent sendIntent = new Intent();
sendIntent.setAction(Intent.ACTION_SEND);
sendIntent.putExtra(Intent.EXTRA_TEXT, textMessage);
sendIntent.setType("text/plain");
```

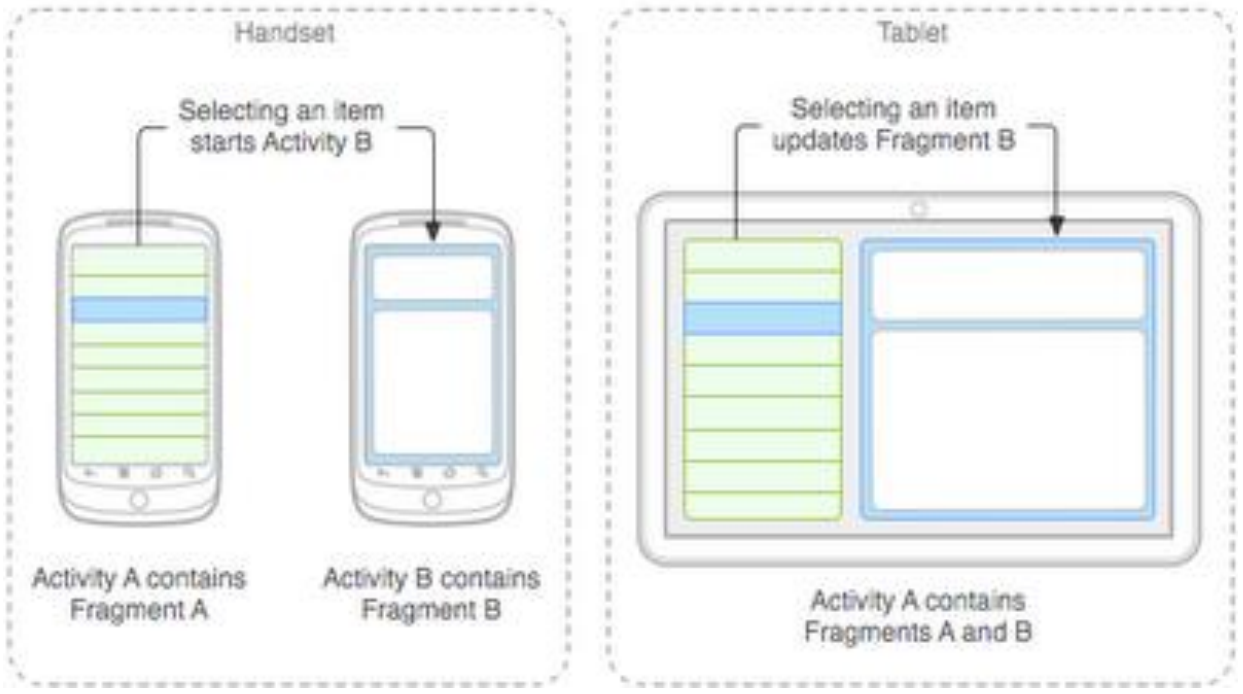
ACTION (No receiving Activity specified)

Data type

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.

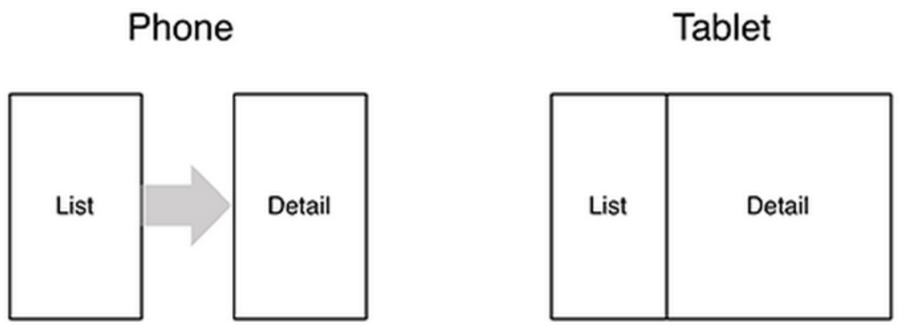


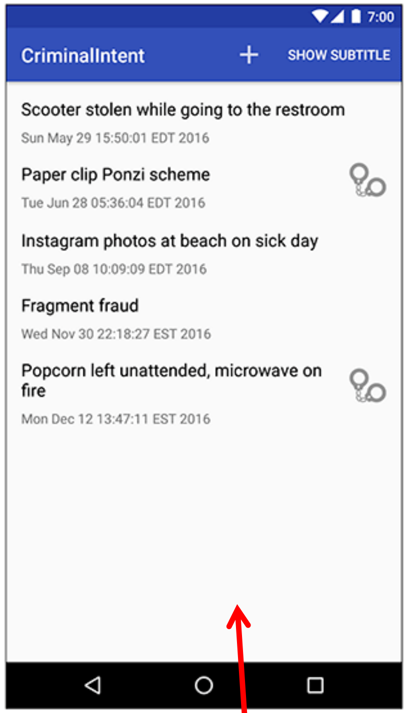
Handset
Selecting an item starts Activity B
Activity A contains Fragment A
Activity B contains Fragment B
Tablet
Selecting an item updates Fragment B
Activity A contains Fragments A and B

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





CriminalIntent + SHOW SUBTITLE

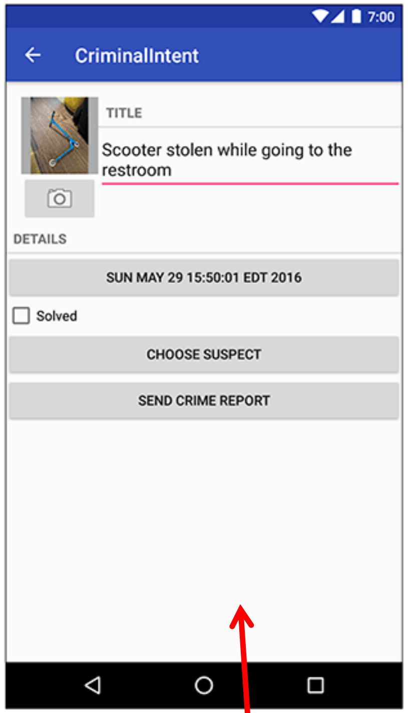
Scooter stolen while going to the restroom
Sun May 29 15:50:01 EDT 2016

Paper clip Ponzi scheme
Tue Jun 28 05:36:04 EDT 2016

Instagram photos at beach on sick day
Thu Sep 08 10:09:09 EDT 2016

Fragment fraud
Wed Nov 30 22:18:27 EST 2016

Popcorn left unattended, microwave on fire
Mon Dec 12 13:47:11 EST 2016



CriminalIntent

TITLE
Scooter stolen while going to the restroom

DETAILS
SUN MAY 29 15:50:01 EDT 2016
Solved
CHOOSE SUSPECT
SEND CRIME REPORT

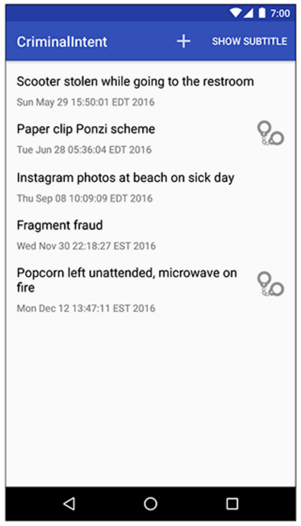
- **On tablet:** show list + detail
- **On phone:** swipe to show next crime

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



7:00

CriminalIntent + SHOW SUBTITLE

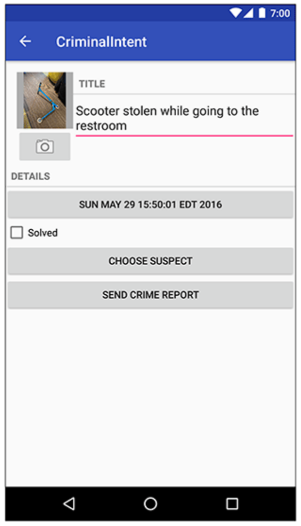
Scooter stolen while going to the restroom
Sun May 29 15:50:01 EDT 2016

Paper clip Ponzi scheme
Tue Jun 28 05:36:04 EDT 2016

Instagram photos at beach on sick day
Thu Sep 08 10:09:09 EDT 2016

Fragment fraud
Wed Nov 30 22:18:27 EST 2016

Popcorn left unattended, microwave on fire
Mon Dec 12 13:47:11 EST 2016



7:00

CriminalIntent

TITLE
Scooter stolen while going to the restroom

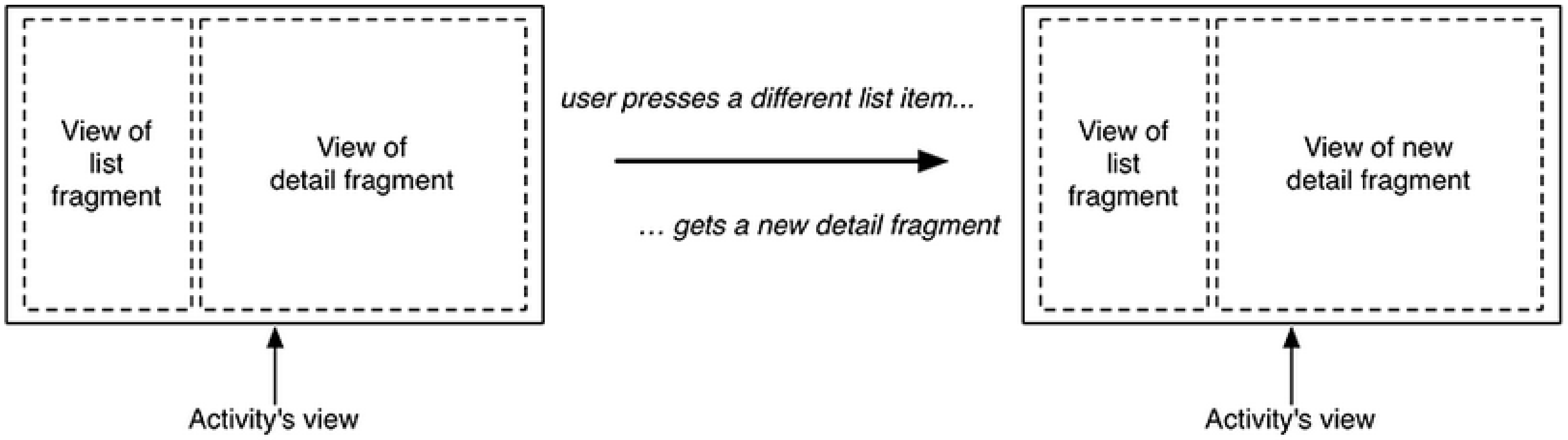
DETAILS

SUN MAY 29 15:50:01 EDT 2016

Solved

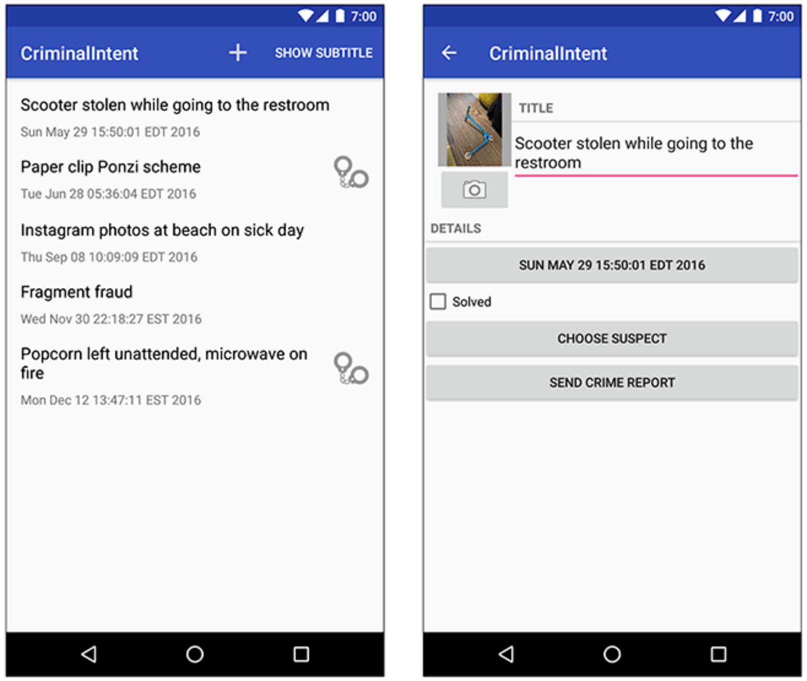
CHOOSE SUSPECT

SEND CRIME REPORT



Starting Criminal Intent

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



7:00

CriminalIntent + SHOW SUBTITLE

Scooter stolen while going to the restroom
Sun May 29 15:50:01 EDT 2016

Paper clip Ponzi scheme
Tue Jun 28 05:36:04 EDT 2016

Instagram photos at beach on sick day
Thu Sep 08 10:09:09 EDT 2016

Fragment fraud
Wed Nov 30 22:18:27 EST 2016

Popcorn left unattended, microwave on fire
Mon Dec 12 13:47:11 EST 2016

7:00

CriminalIntent

TITLE
Scooter stolen while going to the restroom

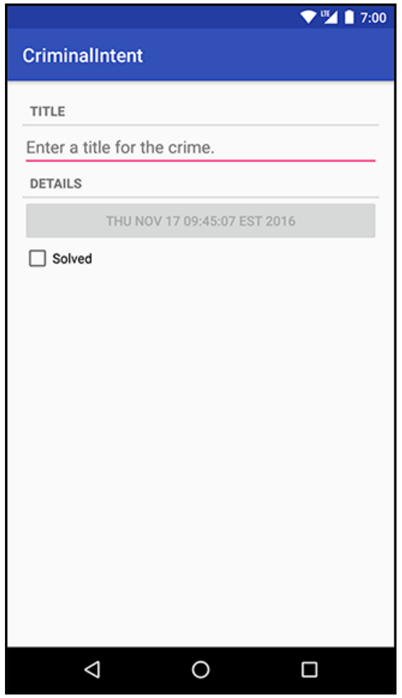
DETAILS
SUN MAY 29 15:50:01 EDT 2016

☐ Solved

CHOOSE SUSPECT

SEND CRIME REPORT

Final Look of CriminalIntent



7:00

CriminalIntent

TITLE
Enter a title for the crime.

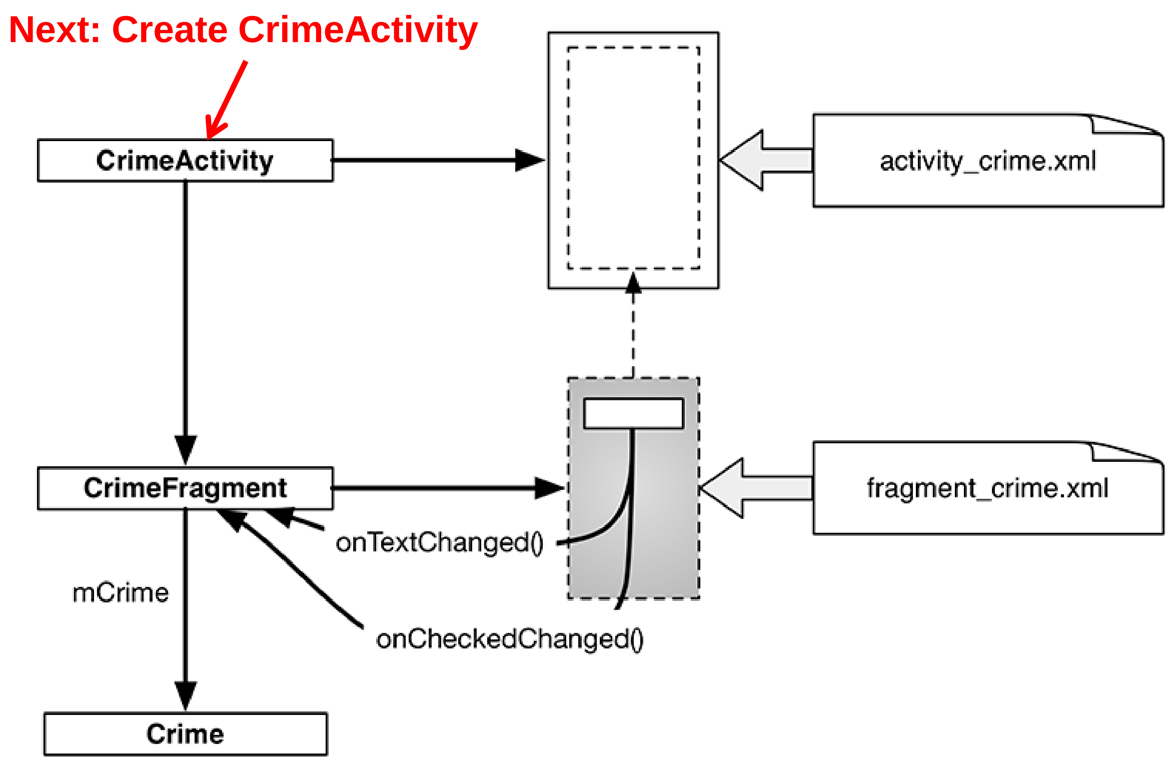
DETAILS
THU NOV 17 09:45:07 EST 2016

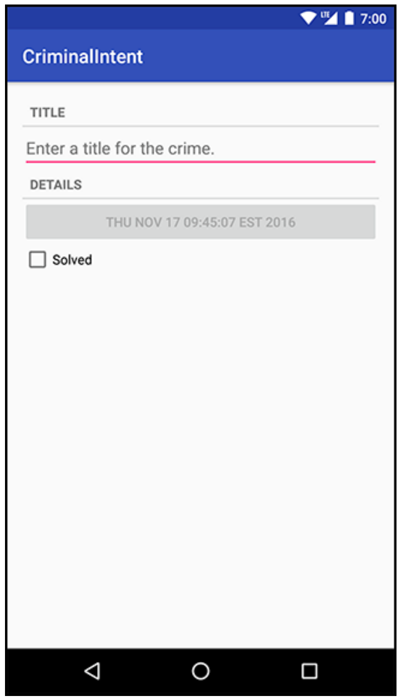
☐ Solved

Start small
Develop detail view using Fragments

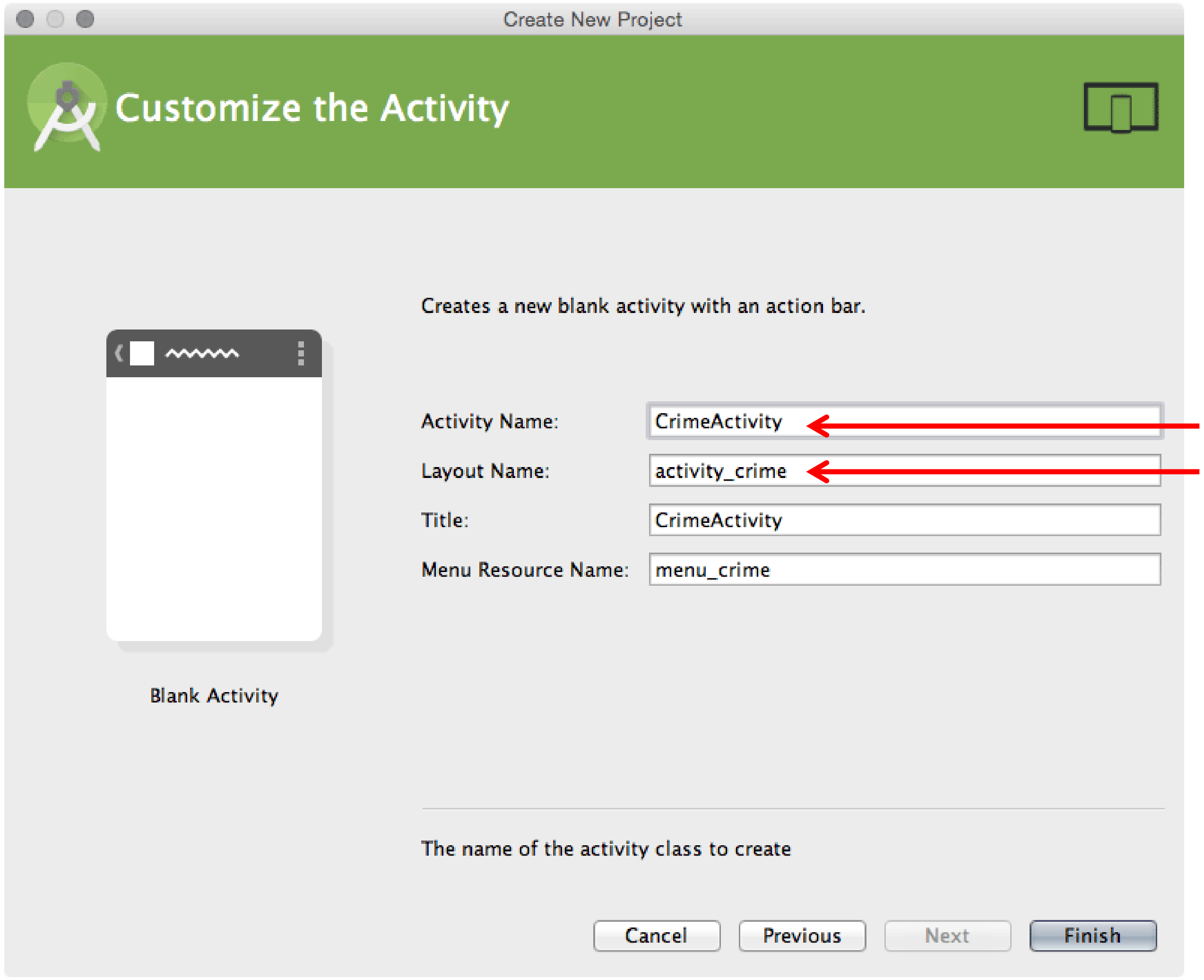
Starting Criminal Intent

- **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



Create New Project

Customize the Activity

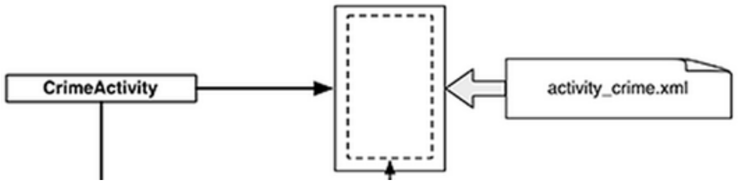
Creates a new blank activity with an action bar.

Activity Name:	CrimeActivity
Layout Name:	activity_crime
Title:	CrimeActivity
Menu Resource Name:	menu_crime

The name of the activity class to create

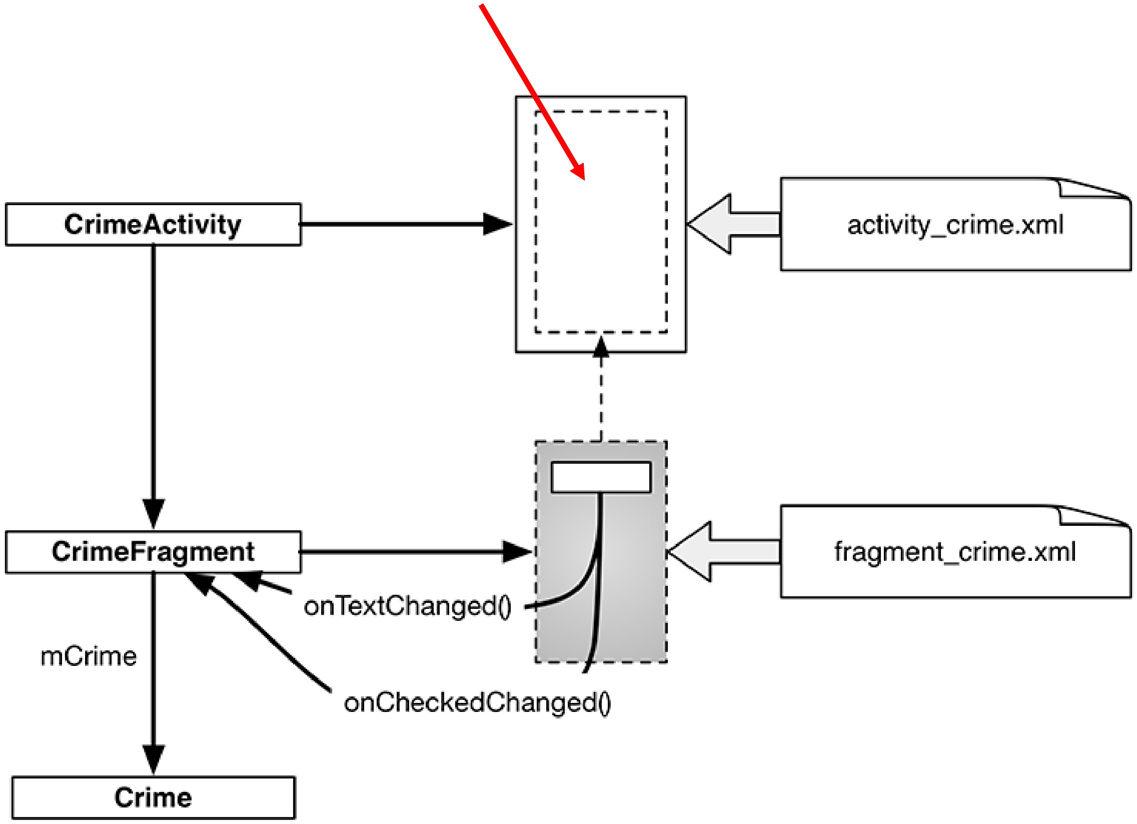
Creates CrimeActivity.java

Formatted using activity_crime.xml



Fragment Hosted by an Activity

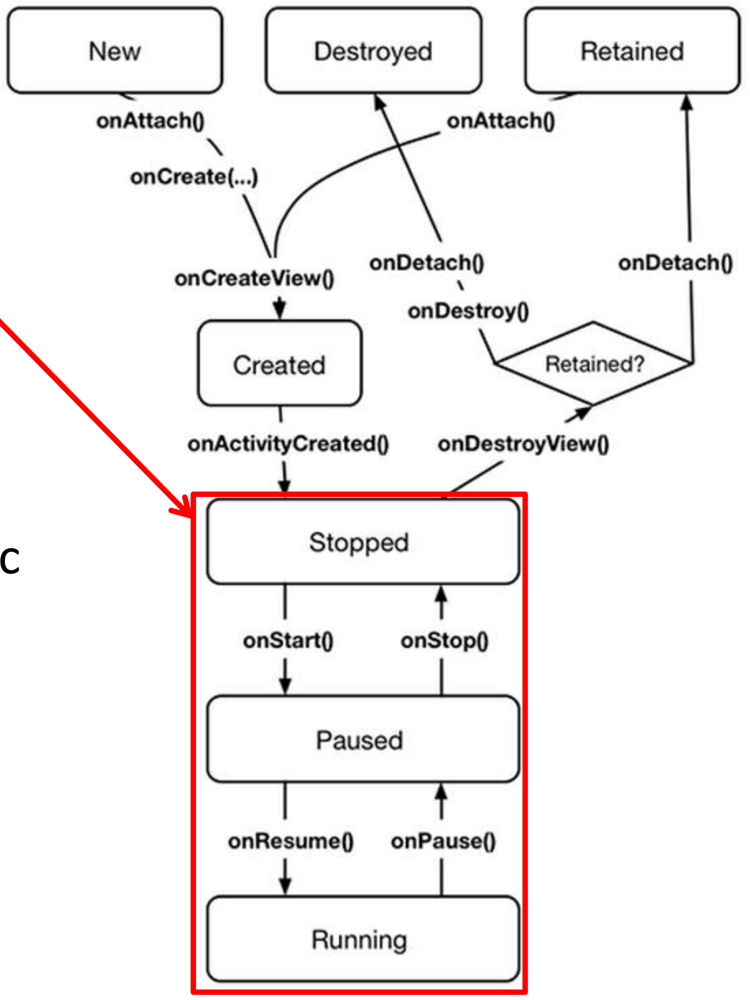
- Each fragment must be hosted by an Activity
- To host a UI fragment, an activity must
  - Define a spot in its layout for the fragment
  - Manage the lifecycle of the fragment instance (next)
- E.g.: **CrimeActivity** defines "spot" for **CrimeFragment**



Fragment's Life Cycle

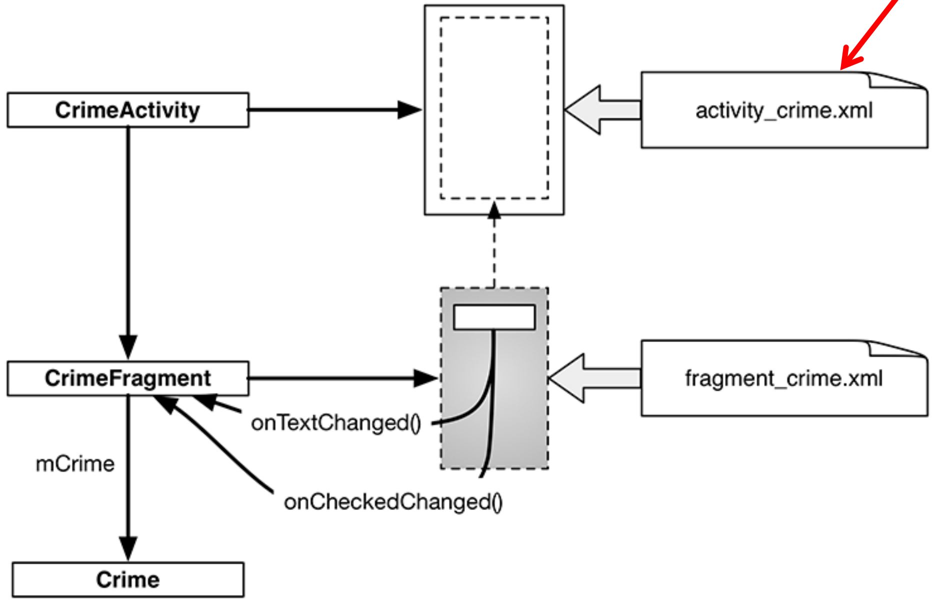
- Fragment's lifecycle similar to activity lifecycle
  - Has states **running**, **paused** and **stopped**
  - Also has some similar activity lifecycle methods (e.g. **onPause()**, **onStop( )**, etc)

- **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**



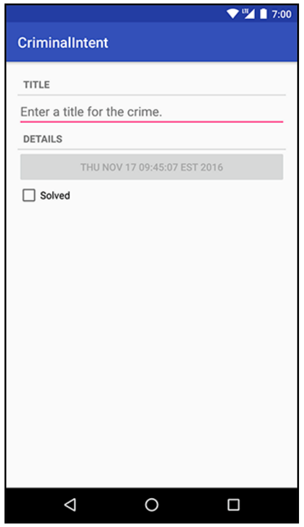
Hosting UI Fragment in an Activity

- 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



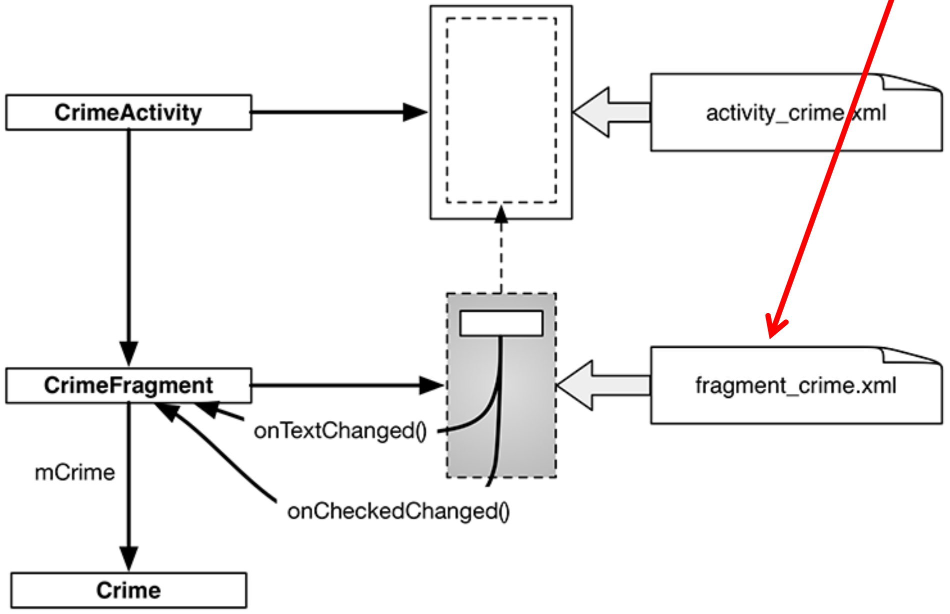
FrameLayout

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

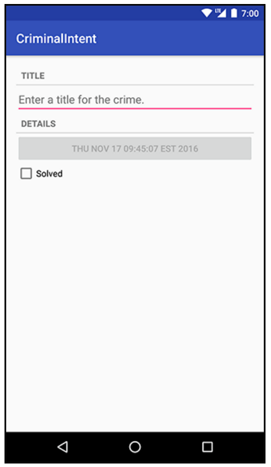


Creating a UI Fragment

- 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

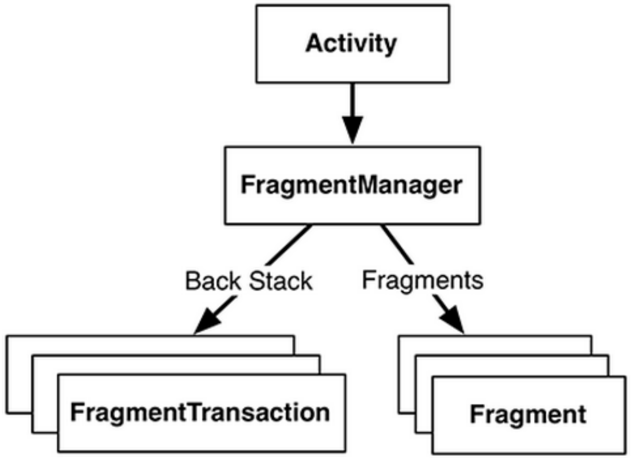
```
public class CrimeFragment extends Fragment {
    private Crime mCrime;

    @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



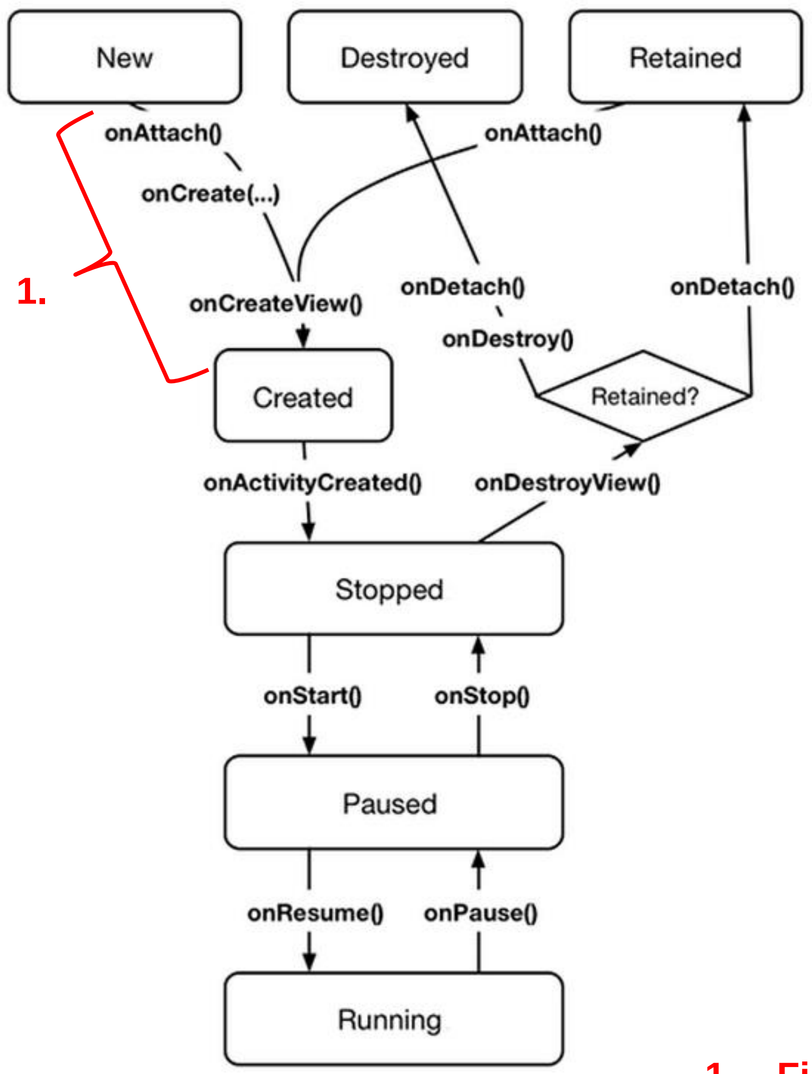
```
public class CrimeActivity extends AppCompatActivity {

    @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

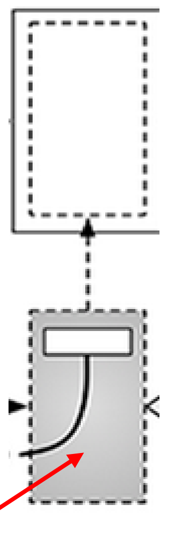
Add Fragment to activity's view

Examining Fragment's Lifecycle



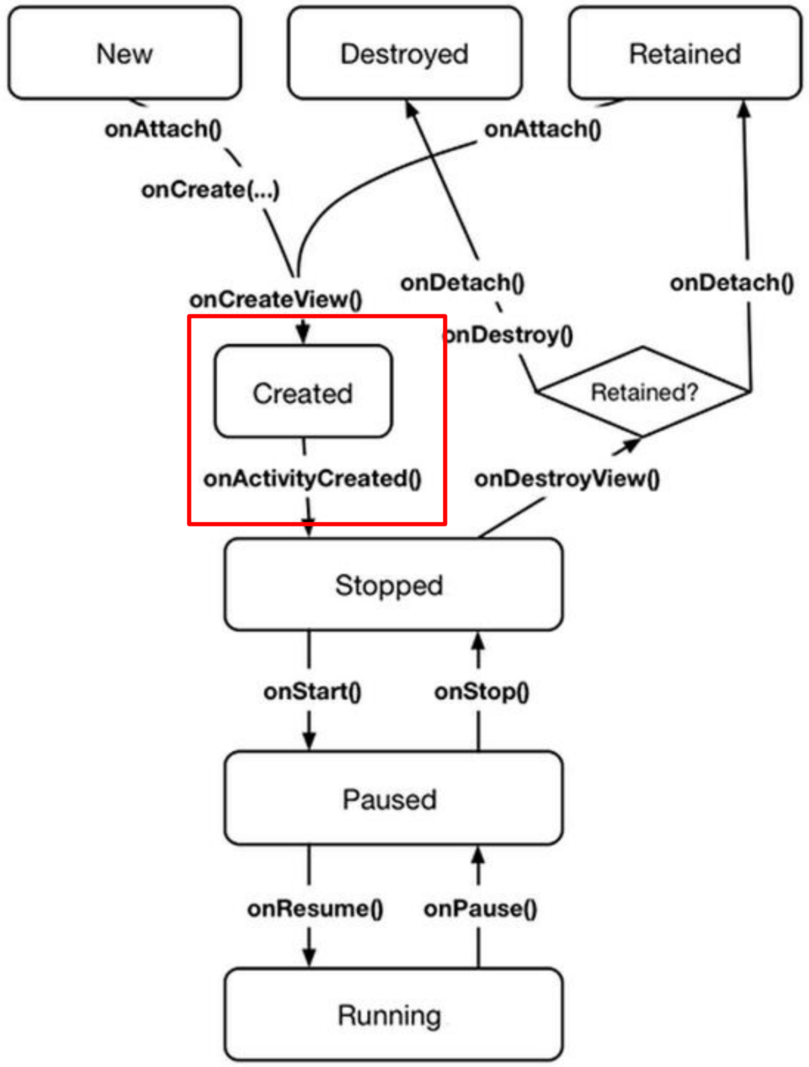
- **FragmentManager** calls fragment lifecycle methods

- **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

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