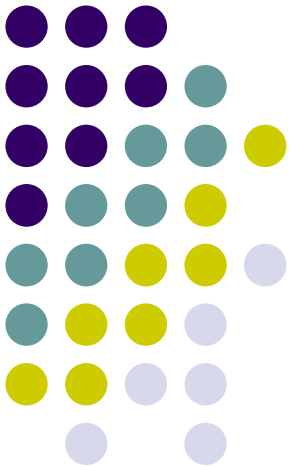
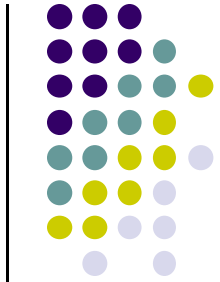
CS 403X Mobile and Ubiquitous Computing

Lecture 7: AdapterViews, Intents

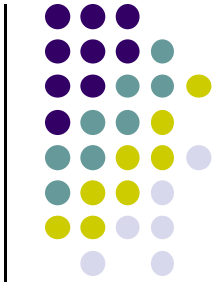
Emmanuel Agu





Data-Driven Layouts

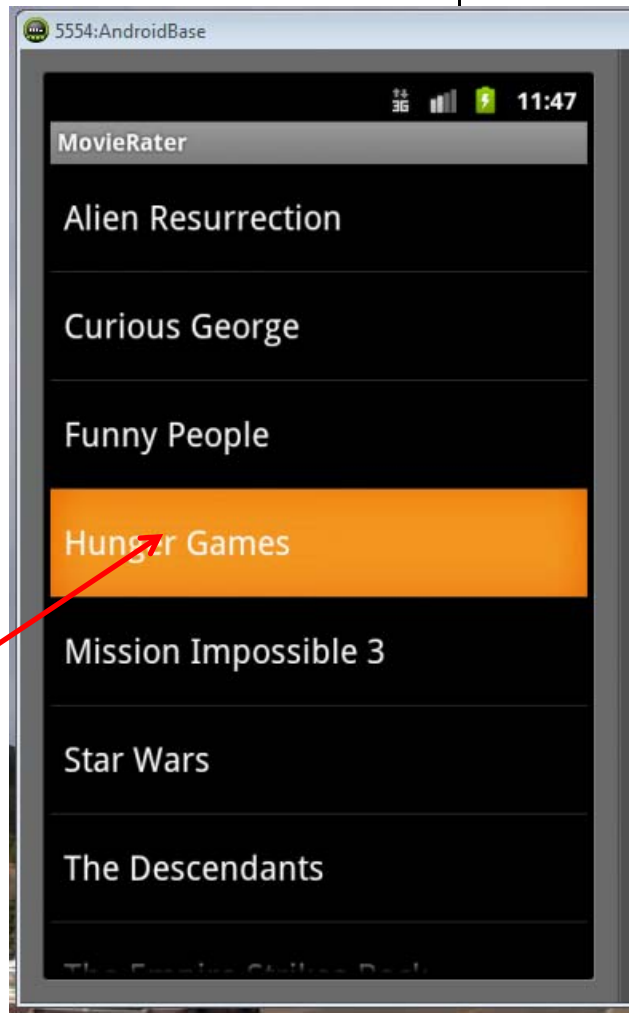
Data-Driven Layouts



- LinearLayout, RelativeLayout, TableLayout, GridLayout useful for positioning UI elements
  - Data is literally hard coded
- Other layouts dynamically composed from data
  - ListView, GridView, GalleryView
  - Tabs with TabHost, TabControl

Data Driven Layouts

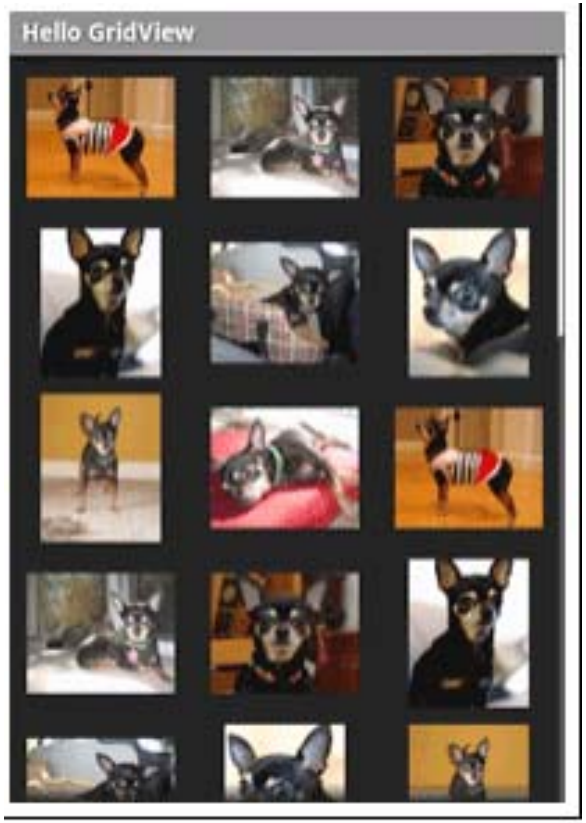
- May want to populate views from a data source (XML file or database)
- Layouts that display repetitive child Views from data
  - ListView
  - GridView
  - GalleryView
- ListView
  - vertical scroll, horizontal row entries, pick item



Data Driven Containers

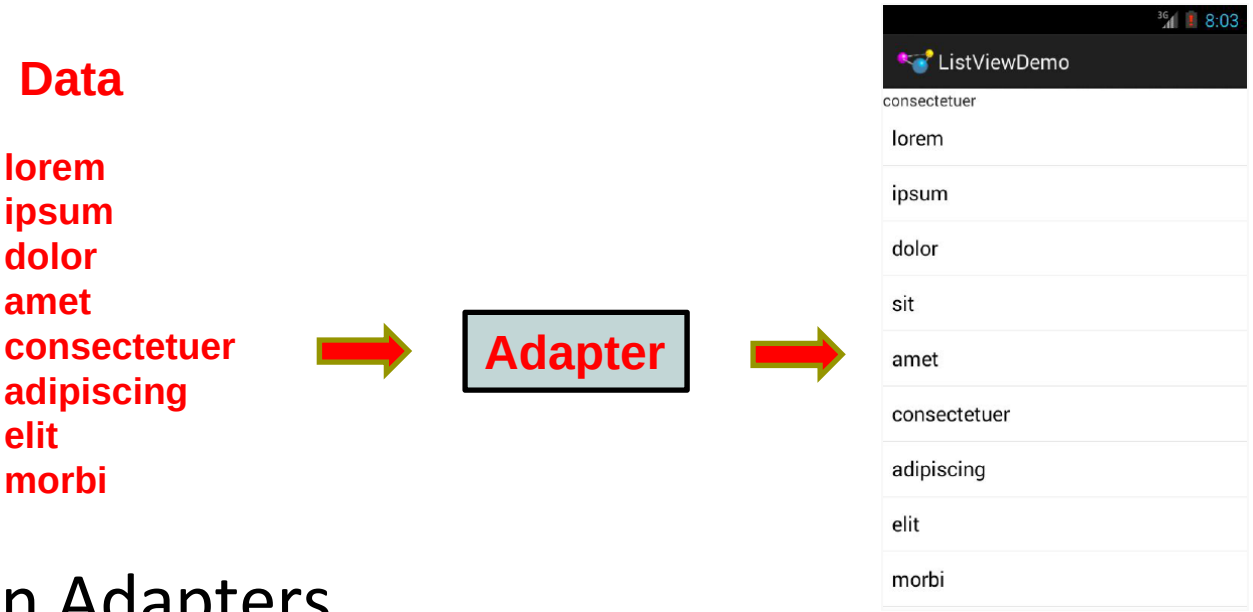
- GridView
  - specified number of rows and columns

- GalleryView
  - horizontal scrolling list, typically images



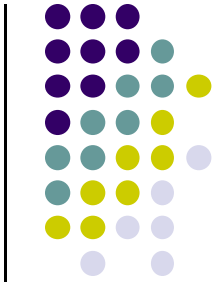
AdapterView

- ListView, GridView, and GalleryView are sub classes of AdapterView
- **Adapter:** generates widgets from a data source, populates layout
  - E.g. Data is adapted into cells of GridView



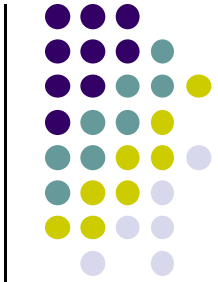
- Most common Adapters
  - **CursorAdapter:** read from database
  - **ArrayAdapter:** read from resource (e.g. XML file)

Adapters



- When using Adapter, a layout is defined for each child element (View)
- The adapter
  - Creates Views (widgets) using layout for each element in data source
  - Fills the containing View (List, Grid, Gallery) with the created Views
- Child Views can be as simple as a TextView or more complex layouts / controls
  - simple views can be declared in android.R.layout

Using ArrayAdapter



- Wraps adapter around a Java array of menu items or **java.util.List** instance

```
String[] items={"this", "is", "a", "really", "silly", "list"};
new ArrayAdapter<String>(this,
                         android.R.layout.simple_list_item_1,
                         items);
```

Context to use.
(e.g app's activity)

Array of items to display

Resource ID of View to use

- E.g. **android.R.layout.simple_list_item_1** turns strings into textView objects

Example: Creating ListView using AdapterArray

- **Task:** Create listView (on right) from strings below

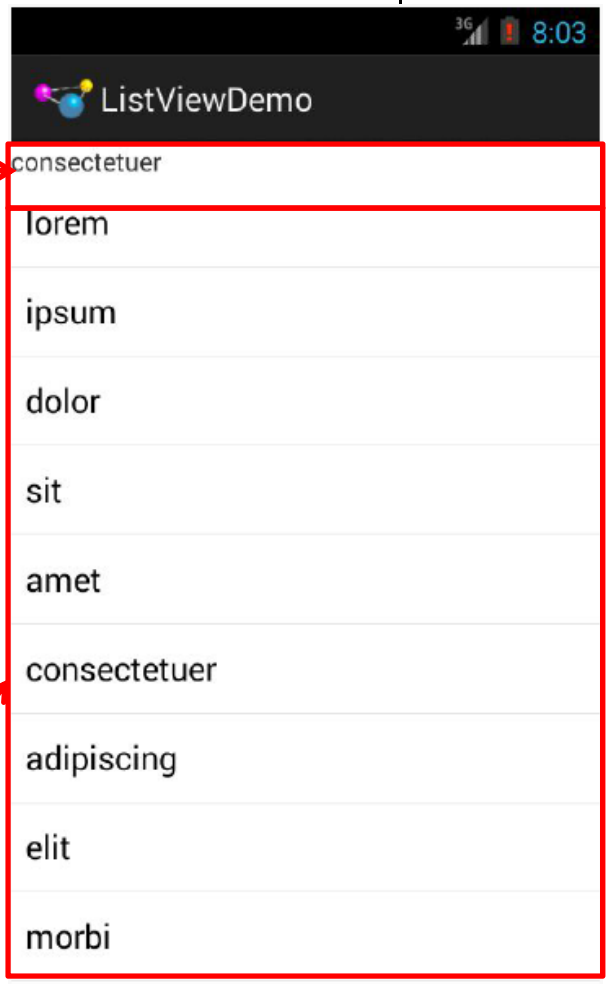
```
private static final String[] items={"lorem", "ipsum", "dolor",
        "sit", "amet",
        "consectetuer", "adipiscing", "elit", "morbi", "vel",
        "ligula", "vitae", "arcu", "aliquet", "mollis",
        "etiam", "vel", "erat", "placerat", "ante",
        "porttitor", "sodales", "pellentesque", "augue", "purus"};
```

Example: Creating ListView using AdapterArray

- First create LinearLayout

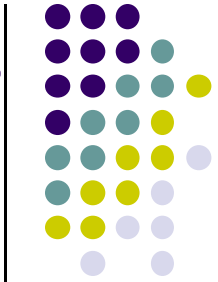
```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
  xmlns:android="http://schemas.android.co
  android:orientation="vertical"
  android:layout_width="match_parent"
  android:layout_height="match_parent">
  <TextView
    android:id="@+id/selection"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"/>
  <ListView
    android:id="@android:id/list"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    />
</LinearLayout>
```

TextView Widget for selected list item



Widget for main list of activity

Example: Creating ListView using AdapterArray

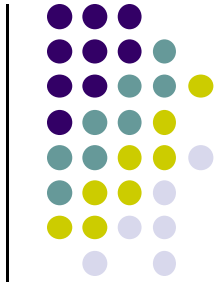


```
package com.commonsware.android.list;

import android.app.ListActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.ArrayAdapter;
import android.widget.ListView;
import android.widget.TextView;
public class ListViewDemo extends ListActivity {
  private TextView selection;
  private static final String[] items={"lorem", "ipsum", "dolor",
          "sit", "amet",
          "consectetuer", "adipiscing", "elit", "morbi", "vel",
          "ligula", "vitae", "arcu", "aliquet", "mollis",
          "etiam", "vel", "erat", "placerat", "ante",
          "porttitor", "sodales", "pellentesque", "augue", "purus"};

  @Override
  public void onCreate(Bundle icicle) {
    super.onCreate(icicle);
    setContentView(R.layout.main);
    setListAdapter(new ArrayAdapter<String>(this,
                        android.R.layout.simple_list_item_1,
                        items));
    selection=(TextView)findViewById(R.id.selection);
  }

  @Override
  public void onListItemClick(ListView parent, View v, int position,
                              long id) {
    selection.setText(items[position]);
  }
}
```

Set list adapter (Bridge Data source and views)

Get handle to TextView of Selected item

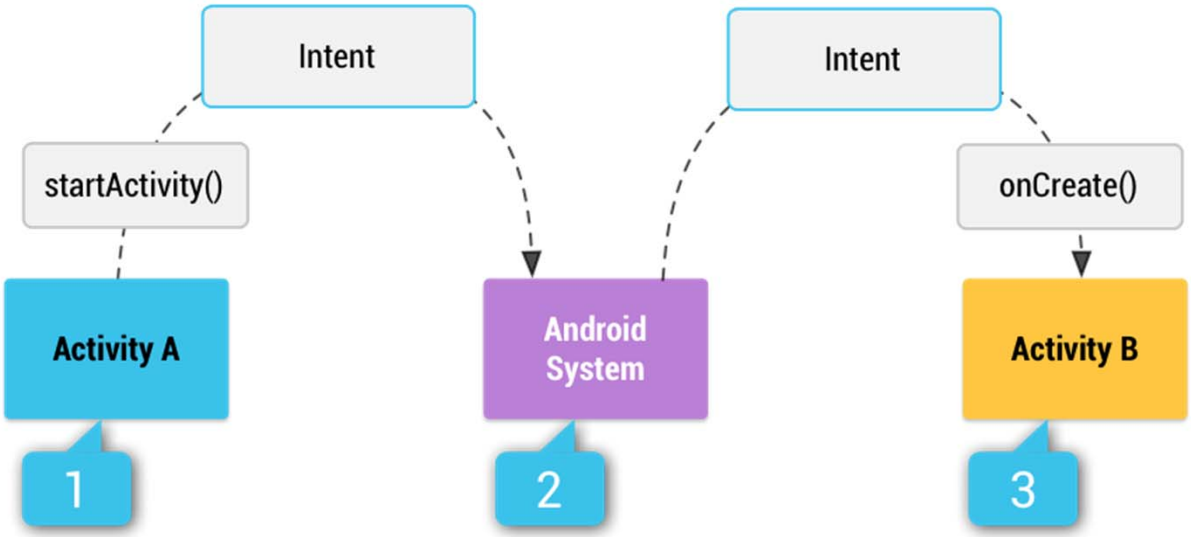
Change Text at top to that of selected view hen user clicks on selection



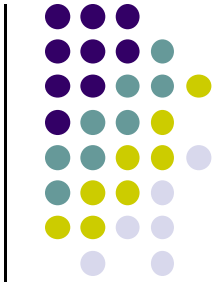
Intents

Intent

- **Intent:** a messaging object used by a component to request action from another app 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, carries any necessary data



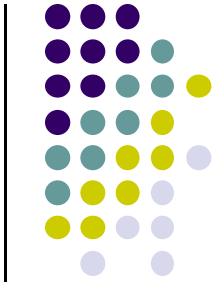
Intent: Result Received Back



- **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, carries any necessary data

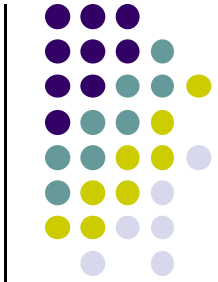
Implicit Vs Explicit Intents



- **Explicit Intent:** If components sending and receiving Intent are in same app
  - E.g. Activity A starts Activity B in same app

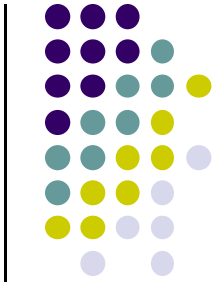
```
Intent intent = new Intent(MainLayoutActivity.this, LinearLayoutActivity.class);
startActivity(intent);
```

- **Implicit Intent:** If components sending and receiving Intent are in **different** apps

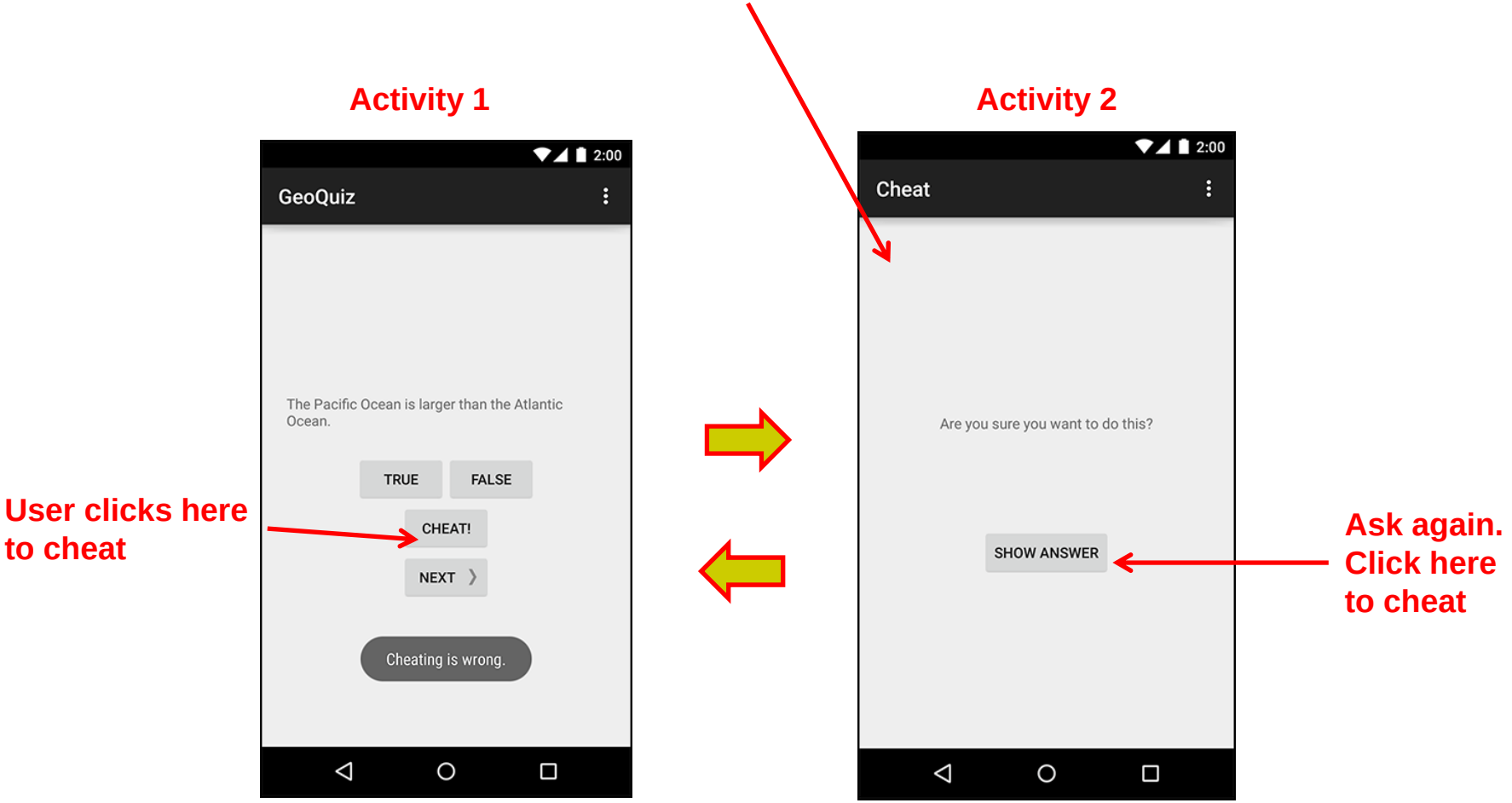


Intent Example: Starting Activity 2 from Activity 1

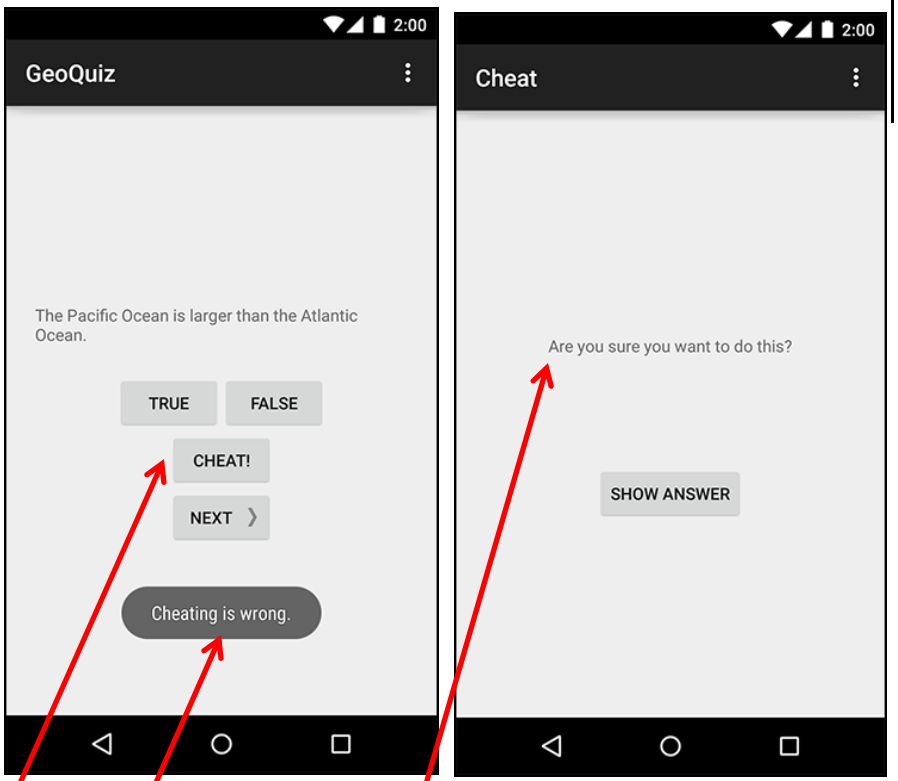
Allowing User to Cheat
Ref: Android Nerd Ranch (2nd edition) pg 87



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

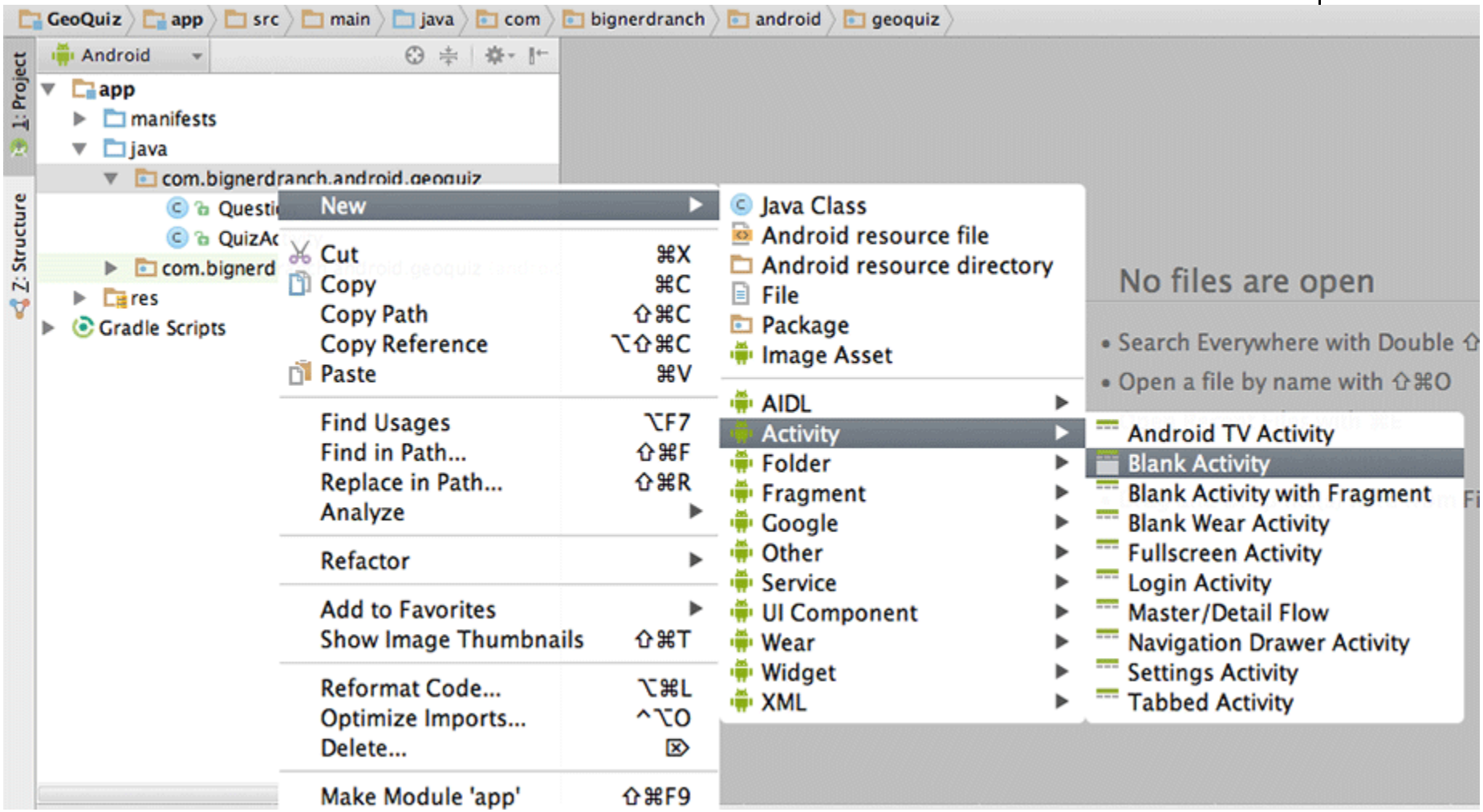


Add Strings for Activity 1 and Activity 2 to strings.xml

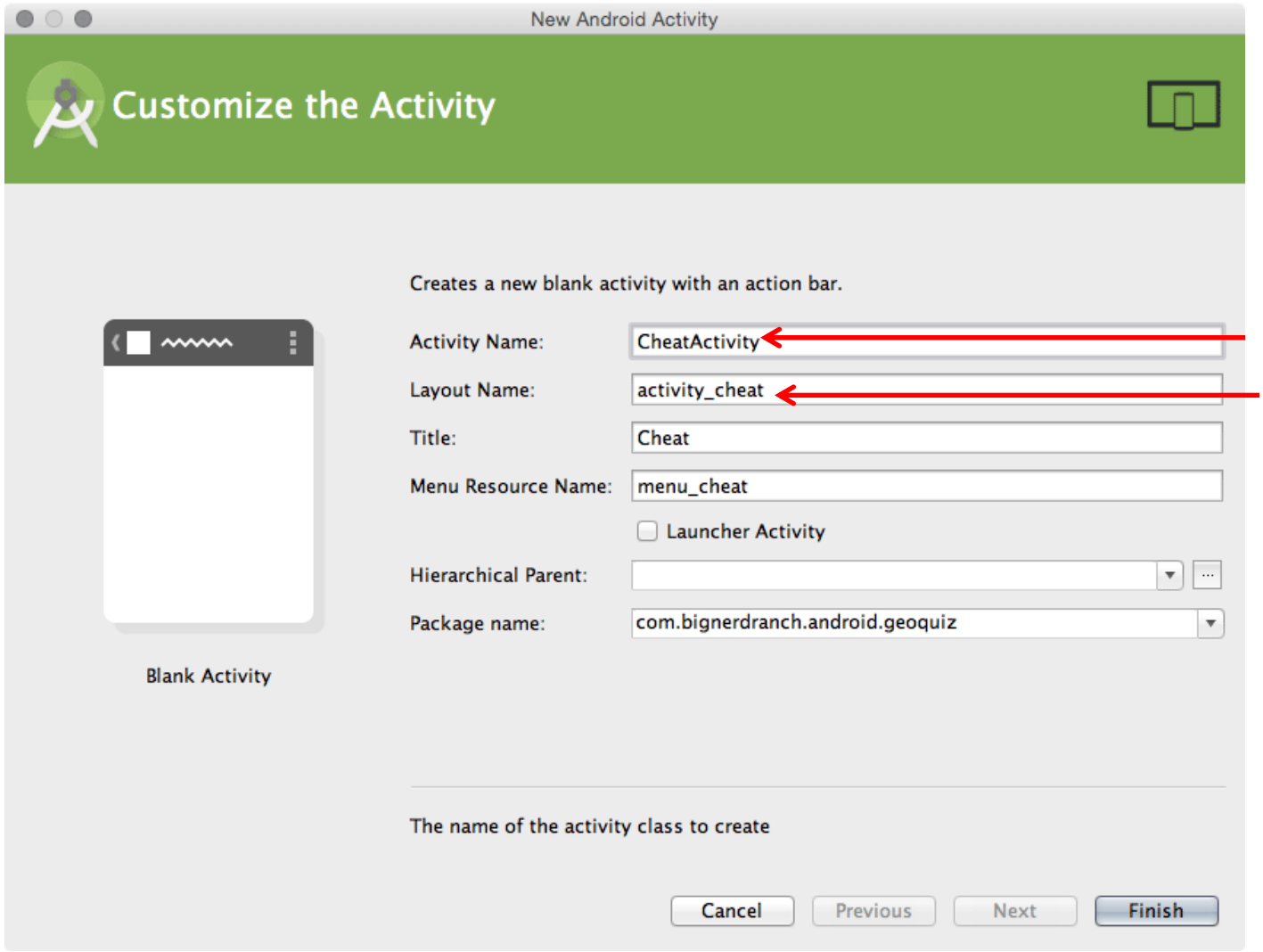



```
<?xml version="1.0" encoding="utf-8"?>
<resources>

    ...
    <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 Blank Activity (for Activity 2) in Android Studio



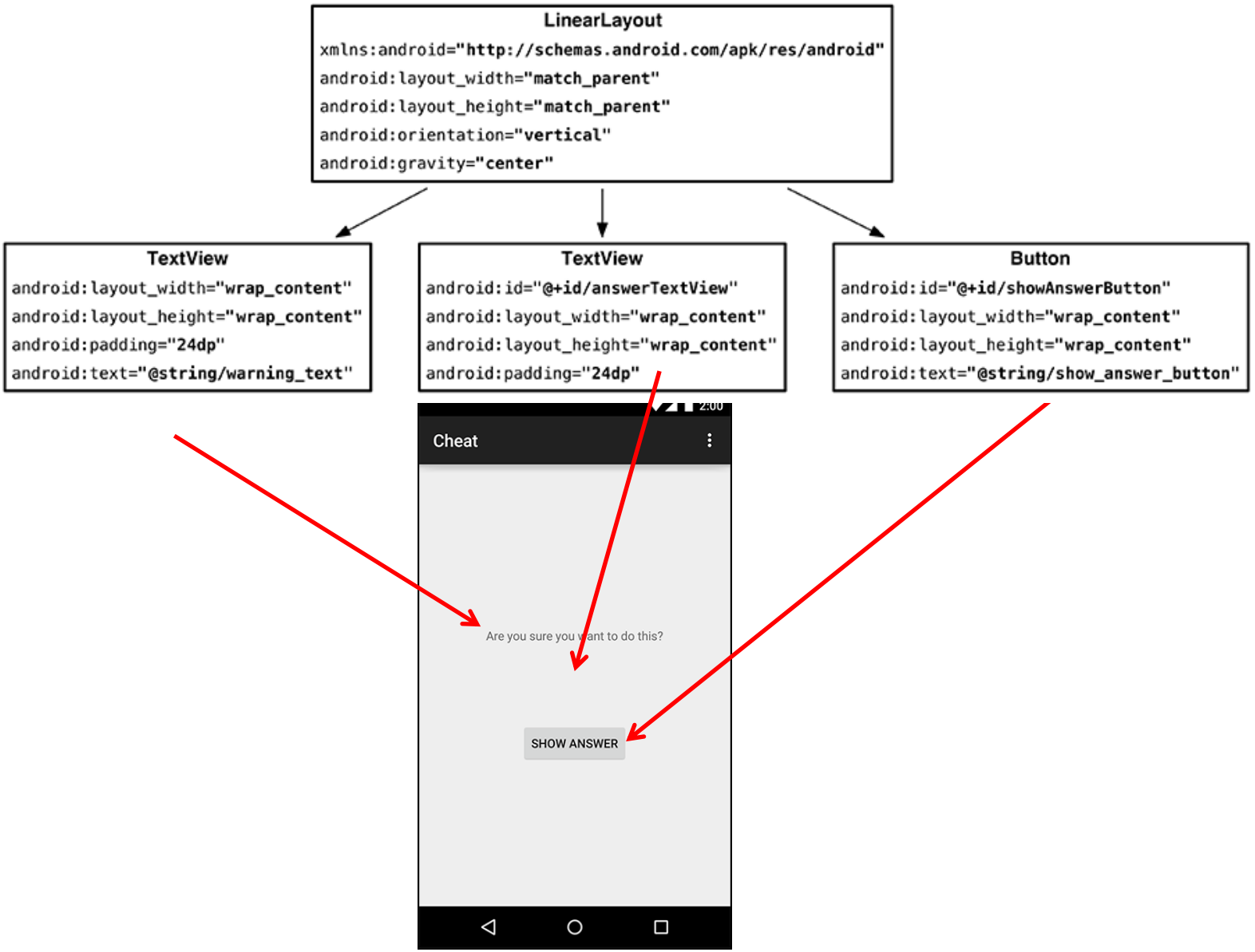
Specify Name and XML file for Activity 2



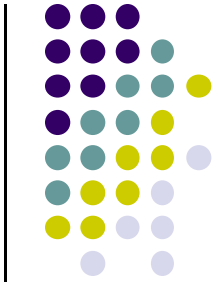
Screen 2
Code in CheatActivity.java

Uses activity_cheat.xml

Design Layout for Screen 2



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: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>
```

Activity 2

Declare New Activity in AndroidManifest.xml

- Create new activity (CheatActivity) in Android Studio

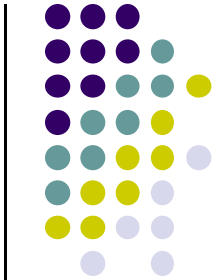
```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.bignerdranch.android.geoquiz" >

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:theme="@style/AppTheme" >
        <activity
            android:name=".QuizActivity"
            android:label="@string/app_name" >
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

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

</manifest>
```

Activity 1

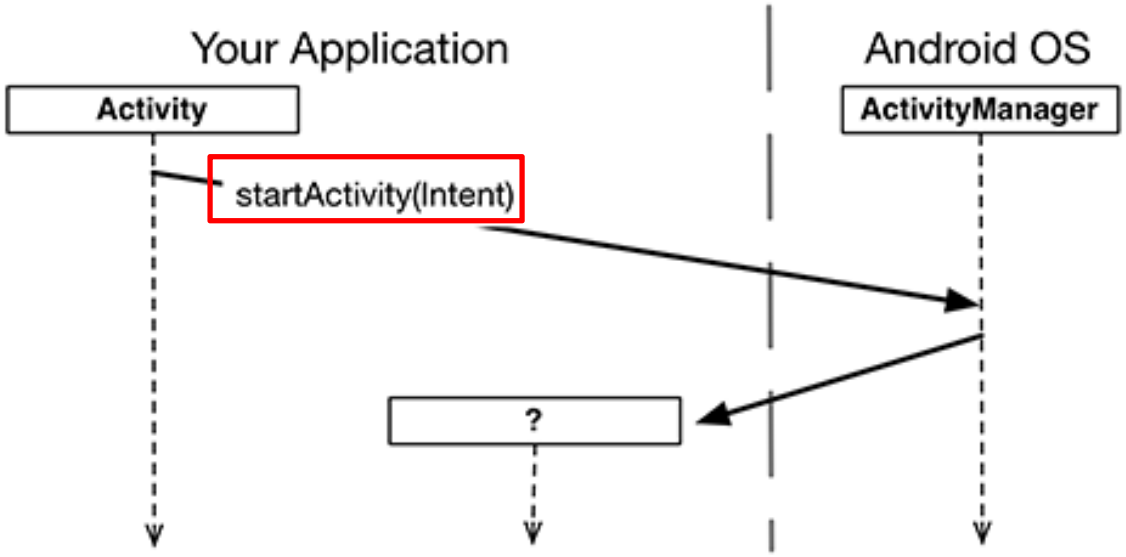
Activity 2 (CheatActivity)

Activity 2 (CheatActivity)

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)

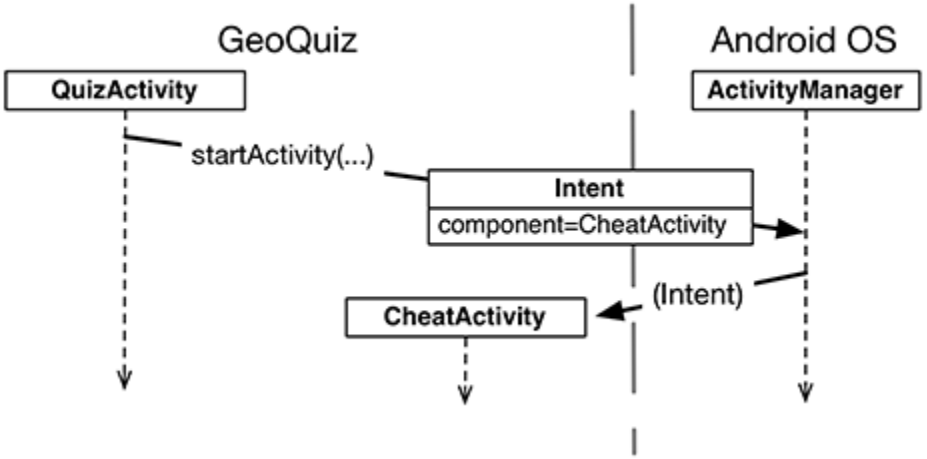


- Intent specifies which (target) Activity Android ActivityManager should start

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 i = new Intent(QuizActivity.this, CheatActivity.class);
        startActivity(i);
    }
});

...
```

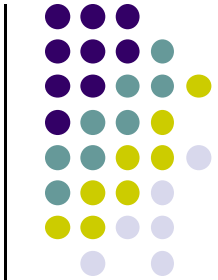
Build Intent

Use Intent to Start new Activity

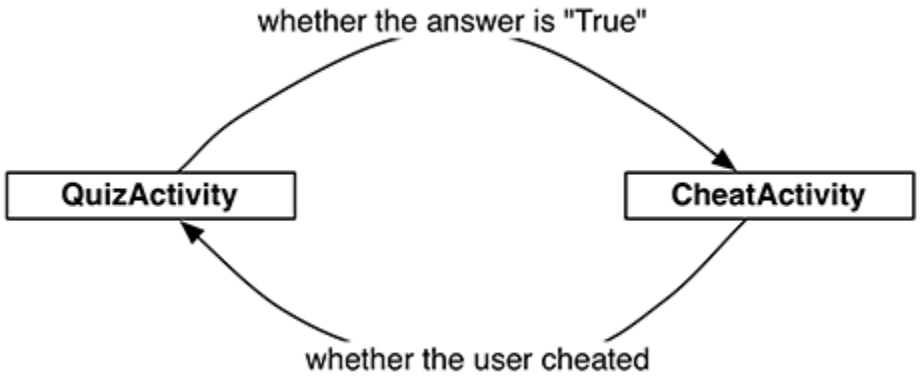
Parent Activity

New Activity 2

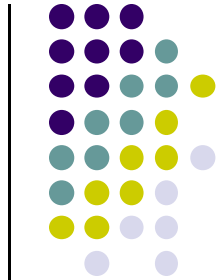
Implicit vs Explicit Intents



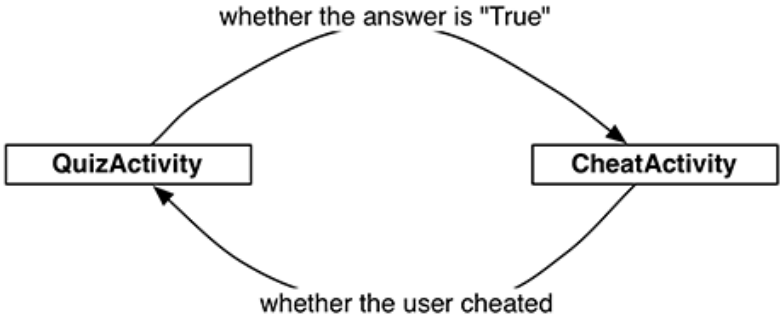
- Previous example is called an **explicit intent**
  - Activity 1 and activity 2 are in same app
- If Activity 2 were in another app, an **implicit intent** would have to be created instead
- Can also pass data between Activities 1 or 2
  - E.g. New Activity 1 can tell activity 1 correct answer (True/False)



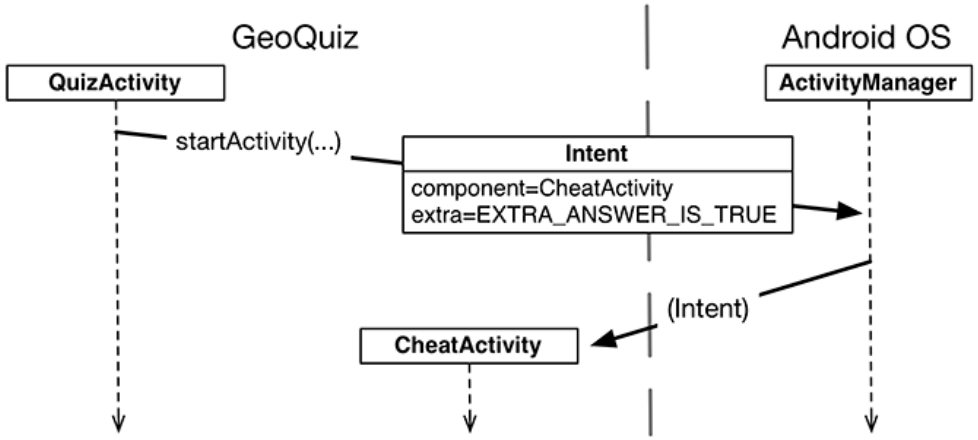
Passing Data Between Activities



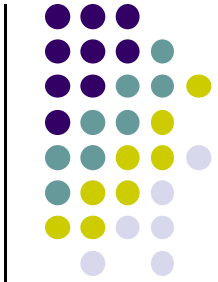
- Need to pass answer (True/False from QuizActivity to CheatActivity)



- 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

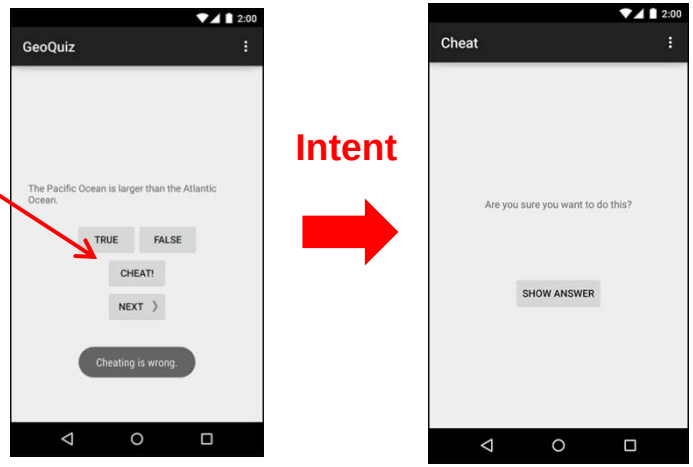


- To add **extra** to Intent, use **putExtra( )** command
- Encapsulate Intent creation into a method **newIntent( )**

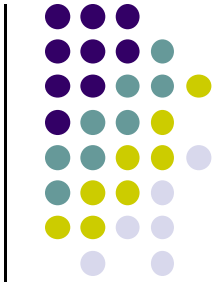
```
public class CheatActivity extends AppCompatActivity {

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

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

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

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

updateQuestion();
}
```

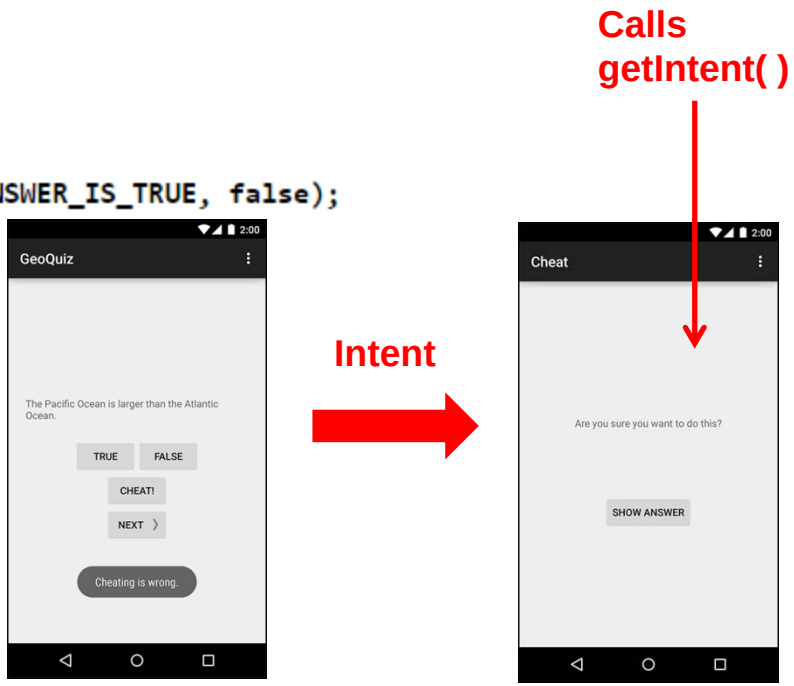


Passing Answer (True/False) as Intent Extra



- 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);
    }

    ...
}
```



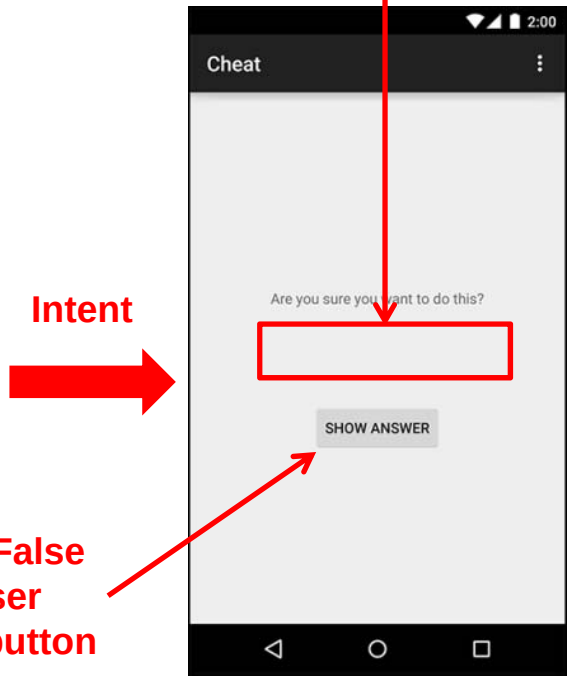
Passing Answer (True/False) as Intent Extra

- Finally, use True/False answer in Intent to display right answer if user clicks button

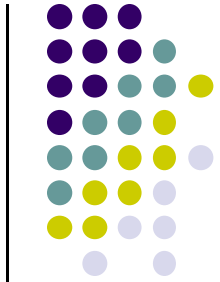
```
public class CheatActivity extends AppCompatActivity {

    ...

    private boolean mAnswerIsTrue;

    private TextView mAnswerTextView;
    private Button mShowAnswer;

    ...

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

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

        mAnswerTextView = (TextView) findViewById(R.id.answer_text_view);

        mShowAnswer = (Button) findViewById(R.id.show_answer_button);
        mShowAnswer.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                if (mAnswerIsTrue) {
                    mAnswerTextView.setText(R.string.true_button);
                } else {
                    mAnswerTextView.setText(R.string.false_button);
                }
            }
        });
    }

}
```

Display cheat answer in this TextView

Intent

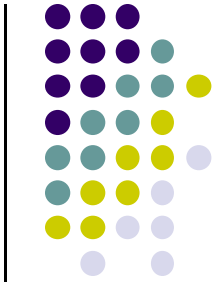


Show True/False answer if user clicks this button



More on Intents

Information Contained by Intent



- **Explicit Intent:** Contains name of component to start

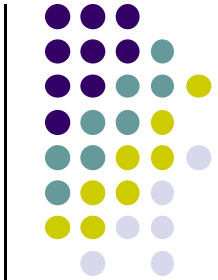
- Recall:

```
...

mCheatButton = (Button)findViewById(R.id.cheat_button);
mCheatButton.setOnClickListener(new View.OnClickListener() {

    @Override
    public void onClick(View v) {
        // Start CheatActivity
        Intent i = new Intent(QuizActivity.this, CheatActivity.class);
        startActivity(i);
    }
});

...
```

Parent Activity

Name of Activity to start

Intents



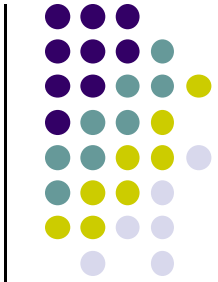
- **Implicit Intent:** Does not name component to start.
- Specifies
  - **Action** (what to do, example visit a web page)
  - **Data** (to perform operation on, example web page url)
- System decides component to receive intent based on **action**, **data, category**
- Example Implicit Intent to share data

```
// 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

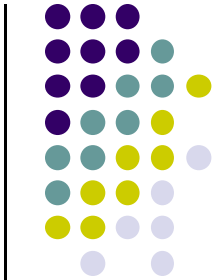
Data type

Intent Action



Constant	Target component	Action
ACTION_CALL	activity	Initiate a phone call.
ACTION_EDIT	activity	Display data for the user to edit.
ACTION_MAIN	activity	Start up as the initial activity of a task, with no data input and no returned output
ACTION_SYNC	activity	Synchronize data on a server with data on the mobile device.
ACTION_BATTERY_LOW	broadcast receiver	A warning that the battery is low.
ACTION_HEADSET_PLUG	broadcast receiver	A headset has been plugged into the device, or unplugged from it.
ACTION_SCREEN_ON	broadcast receiver	The screen has been turned on.
ACTION_TIMEZONE_CHANGED	broadcast receiver	The setting for the time zone has changed.

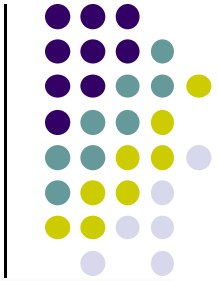
Intent Info - *Category*



- Some Intents also have categories
- String describing what kind of component should handle Intent

Constant	Meaning
`CATEGORY_BROWSABLE`	The target activity can be safely invoked by the browser to display data referenced by a link – for example, an image or an e-mail message.
`CATEGORY_GADGET`	The activity can be embedded inside of another activity that hosts gadgets.
`CATEGORY_HOME`	The activity displays the home screen, the first screen the user sees when the device is turned on or when the *Home* button is pressed.
`CATEGORY_LAUNCHER`	The activity can be the initial activity of a task and is listed in the top-level application launcher.
`CATEGORY_PREFERENCE`	The target activity is a preference panel.

Recall: Inside AndroidManifest.xml: Launcher Intent



Your package name

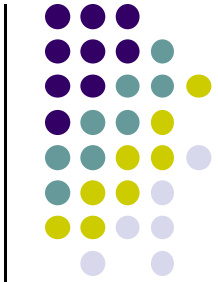
Android version

List of activities (screens) in your app

Action of intent

```
<?xml version="1.0"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
  package="com.commonsware.android.skeleton"
  android:versionCode="1"
  android:versionName="1.0">

  <application>
    <activity
      android:name="Now"
      android:label="Now">
      <intent-filter>
        <action android:name="android.intent.action.MAIN" />

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

</manifest>
```

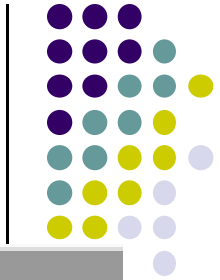
Intent Info - *Data*



- **Data:** URI (uniform resource identifier) or MIME type of data to be acted on

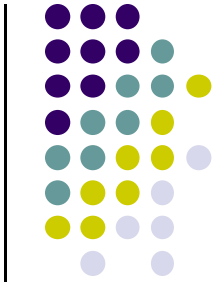
- MIME (Multipurpose Internet Mail Extension): describes type of information
  - `image/png` or `audio/mpeg`

Intent Constructors



Public Constructors
Intent () Create an empty intent.
Intent (Intent o) Copy constructor.
Intent (String action) Create an intent with a given action.
Intent (String action, Uri uri) Create an intent with a given action and for a given data url.
Intent (Context packageContext, Class<?> cls) Create an intent for a specific component.
Intent (String action, Uri uri, Context packageContext, Class<?> cls) Create an intent for a specific component with a specified action and data.

References



- Android Nerd Ranch (2nd edition)
- 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