MULTIPLE ACTIVITIES

Mobile Application Development CS-347

Instructor: Philip Garofalo, M.S.

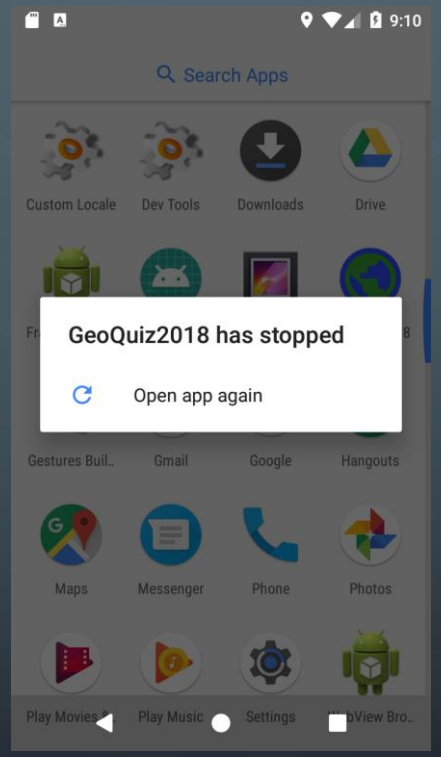
NEIU

DEBUGGING ANDROID APPS

- Comment out the following line in QuizActivity.onCreate():

```
@Override
protected void onCreate(Bundle savedInstanceState) {
  super.onCreate(savedInstanceState);
  Log.d(TAG, "onCreate: ");
  setContentView(R.layout.activity_quiz);

  if (savedInstanceState != null) {
    mCurrentIndex = savedInstanceState.getInt(KEY_INDEX, 0);
  }

  //mQuestionTextView = (TextView)findViewById(R.id.question_text_view);
  ...
```

RUN THE APP NOW AND … OOPS!



LOGCAT

- Use the Logcat console to see error and log messages output by your app.
- Logcat content includes exceptions and stack traces.
- A stack trace is a list of all the calls to a particular line of code. Usually in reverse order.
- To explicitly generate an exception stack trace use
  - Log.d(String, String, Throwable).
  - For example:
  - Log.d(TAG, “Beginning of Foo function”, new Exception());
  - This will generate a stack trace in Logcat at that line.

BREAKPOINTS

- **Breakpoints** allow you to stop at a particular line in your program.
- Whereupon you can observe
  - register and variable values
  - the call stack
  - call arguments
- You can set **watchpoint** that will stop your program when a variable reaches a particular value.
- Once your app has stopped, you can then **step through your code** one line at a time.
- See all the breakpoints you've set by selecting **Run>View Breakpoints...**

EXCEPTION BREAKPOINTS

- You can stop your program when an exception occurs with **exception breakpoints**.
- The program will stop where the exception is thrown.
- Obviously, this will work with both caught and uncaught exceptions.

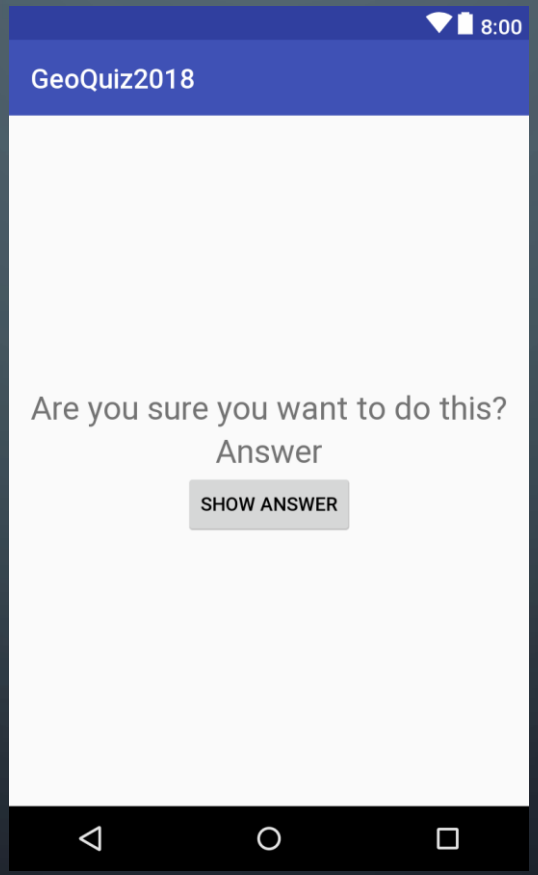
ANDROID LINT

- Lint is a **static analysis** tool. It examines your source code.
- A dynamic analysis tool observers your program while it is running.
- Lint give you a closer examination of code and find subtle errors and places for potential errors.
- It will report compatibility errors, initialization mistakes,

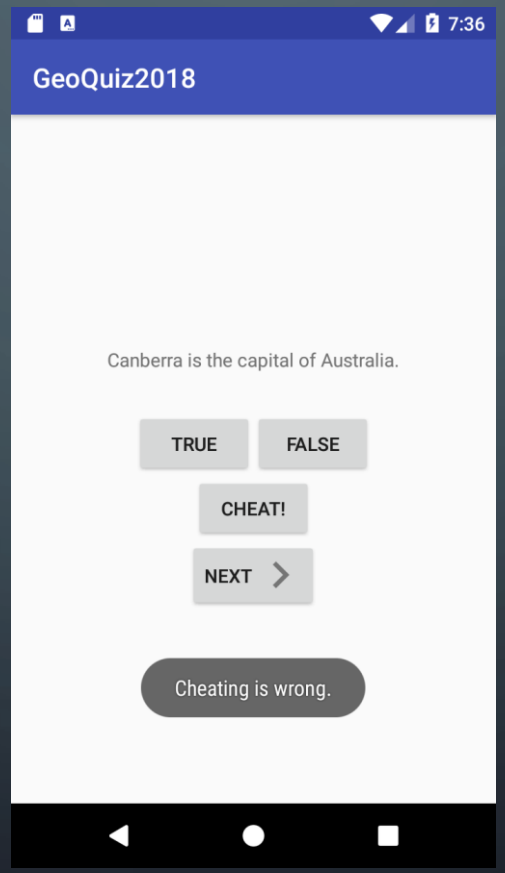
R CLASS DEBUGGING

- Mysterious problems with the R Class? Try the following:
  - Check your resource files for correct XML.
  - Clean your project and rebuild with **Build>Clean Project** or **Build>Rebuild Project**.
  - Sync your project with Gradle with **Tools>Android>Sync Project with Gradle Files**
  - Run Android Lint with **Analyze>Inspect Code...**

WE'RE GOING TO ADD A CHEAT SCREEN!



...AND A SCOLD SCREEN!



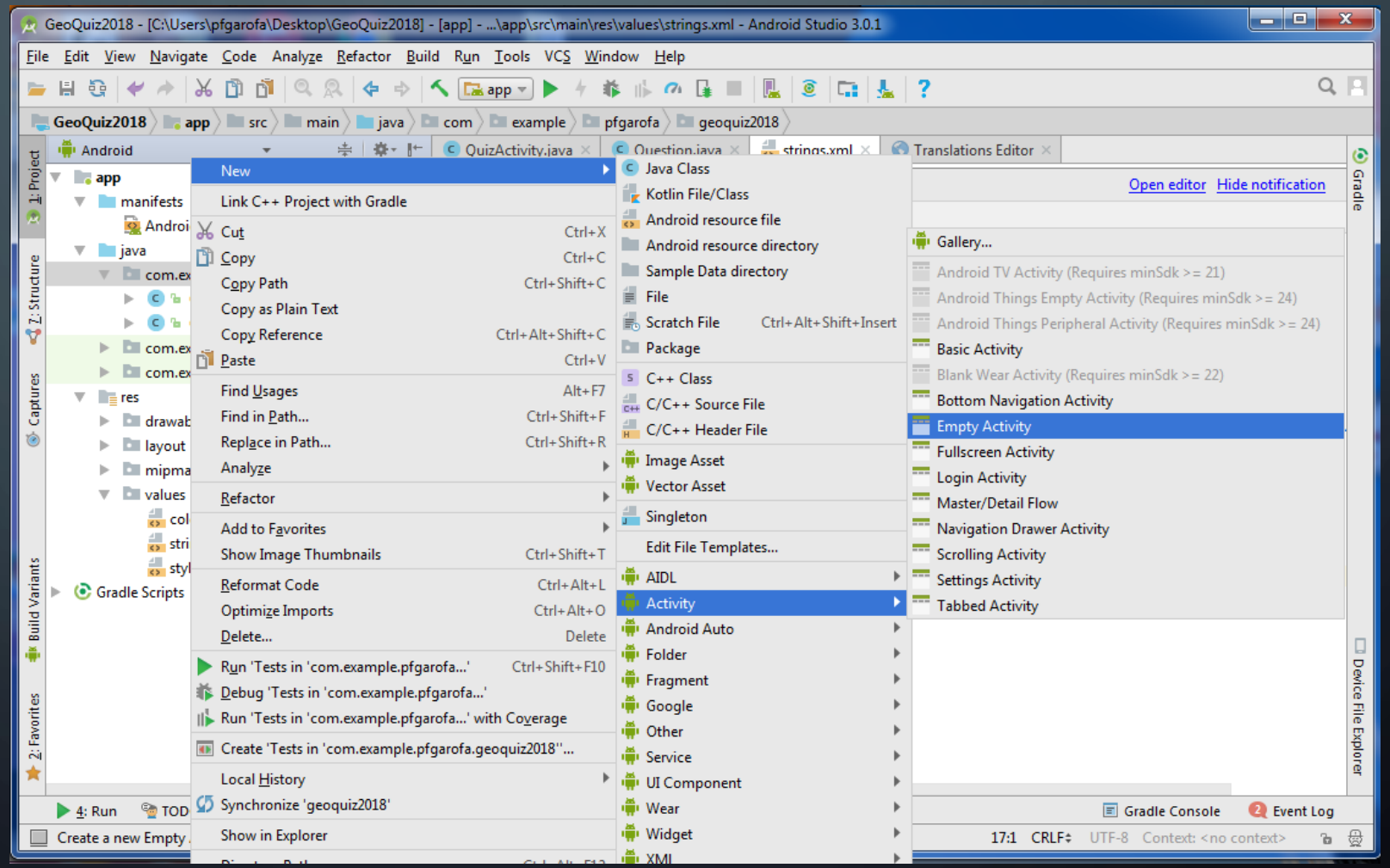
SETTING UP YOUR SECOND ACTIVITY

- For starters, let's add four new strings.

```
<resources>
    ...
    <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="judgement_toast">Cheating is wrong.</string>
</resources>
```

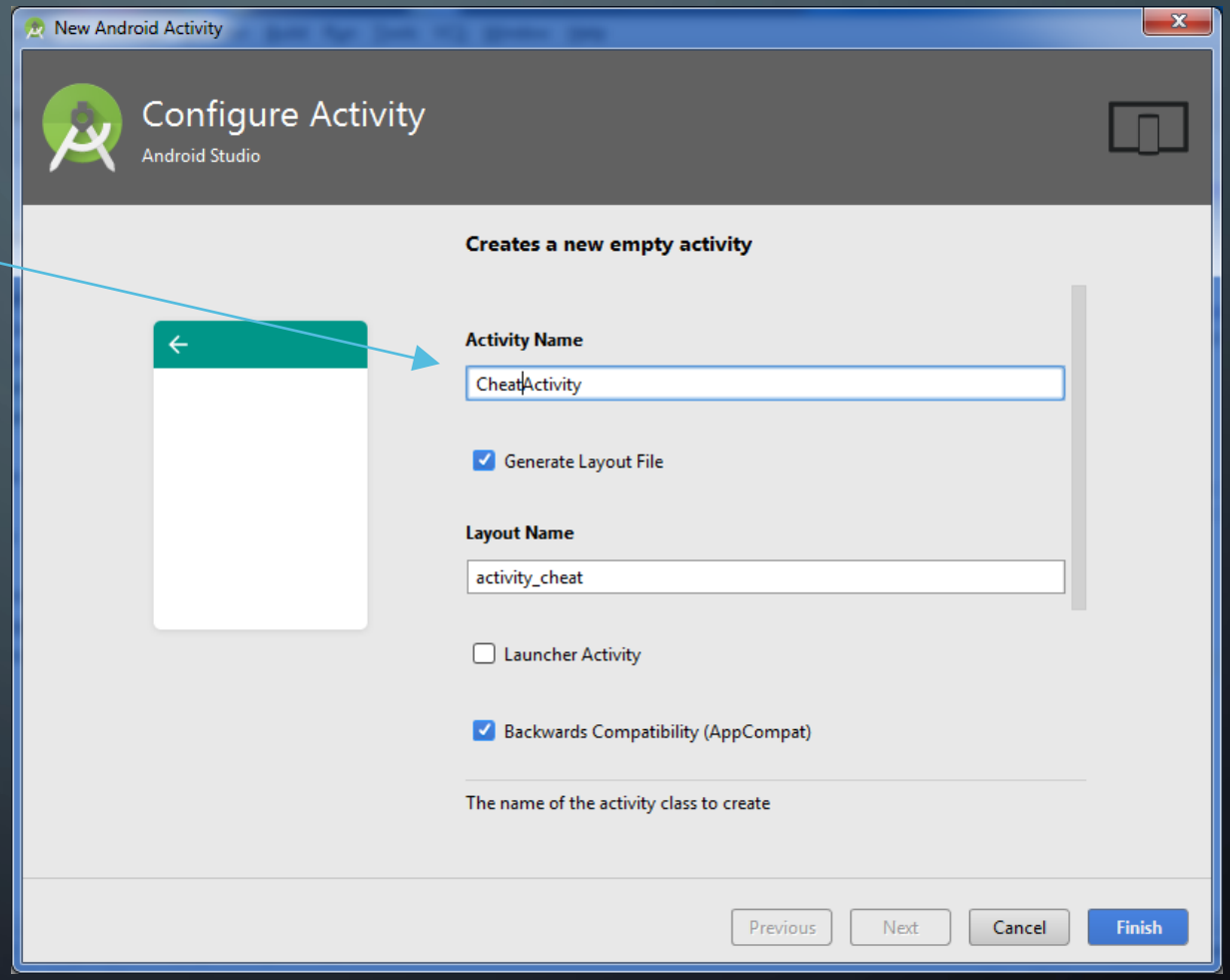
CREATING A NEW ACTIVITY

Create a new activity by right-clicking on the class package and selecting New>Activity>EmptyActivity



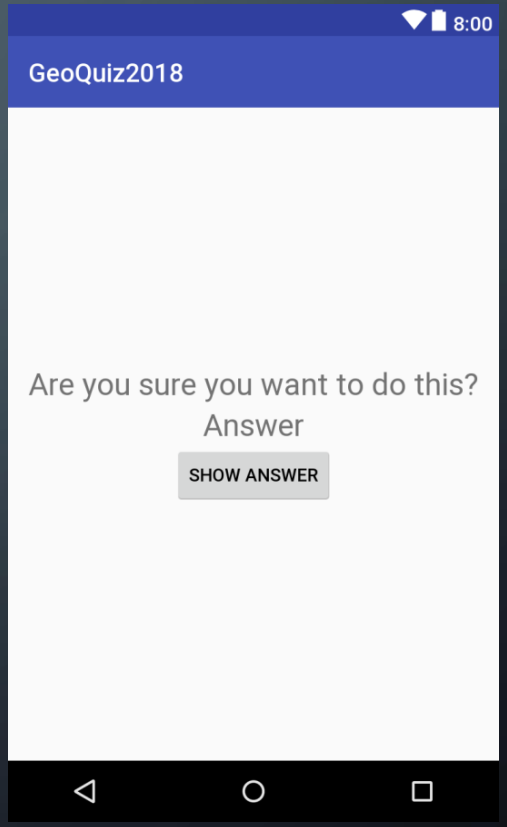
CREATING A NEW ACTIVITY

Name the activity CheatActivity.



DESIGN THE NEW ACTIVITY'S LAYOUT

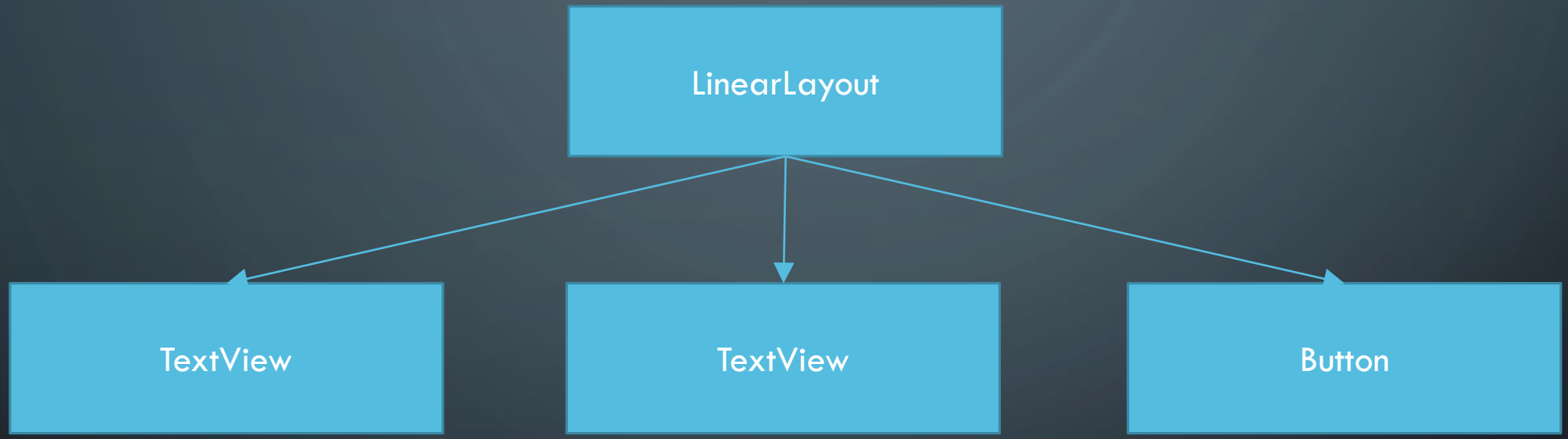
- Change the layout to LinearLayout.
- Set its orientation to vertical.
- Set its gravity to center.
- Add two TextView widgets.
  - Set their text size to 24sp
  - Set the widths and heights to wrap_content.
  - Set the first TextView's text to warning_text.
  - Set the second TextView's id to answer_text_view.
  - Set its **tool** text to "Answer".
- Add a Button Widget.
  - Set its width to wrap_content.
  - Set its id to show_answer_button.
  - It's text to show_answer_button.



WHAT'S THE DIFFERENCE BETWEEN SP AND DP?

- **dp** stands for *density-independent pixel*. Pronounced "dip". Used for margins, paddings, and box sizes. Anwhere you specify pixels.
- **sp** stands for *scale-independent pixel*. They're dps that also use font size as a factor. Used for text.
- **px** is for *pixels, individual screen dots*.
- **pt** is for points (72 per inch). Corresponds to a measure used in print publishing.

CHEAT ACTIVITY'S VIEW HIERARCHY



DESIGN THE NEW ACTIVITY'S LAYOUT

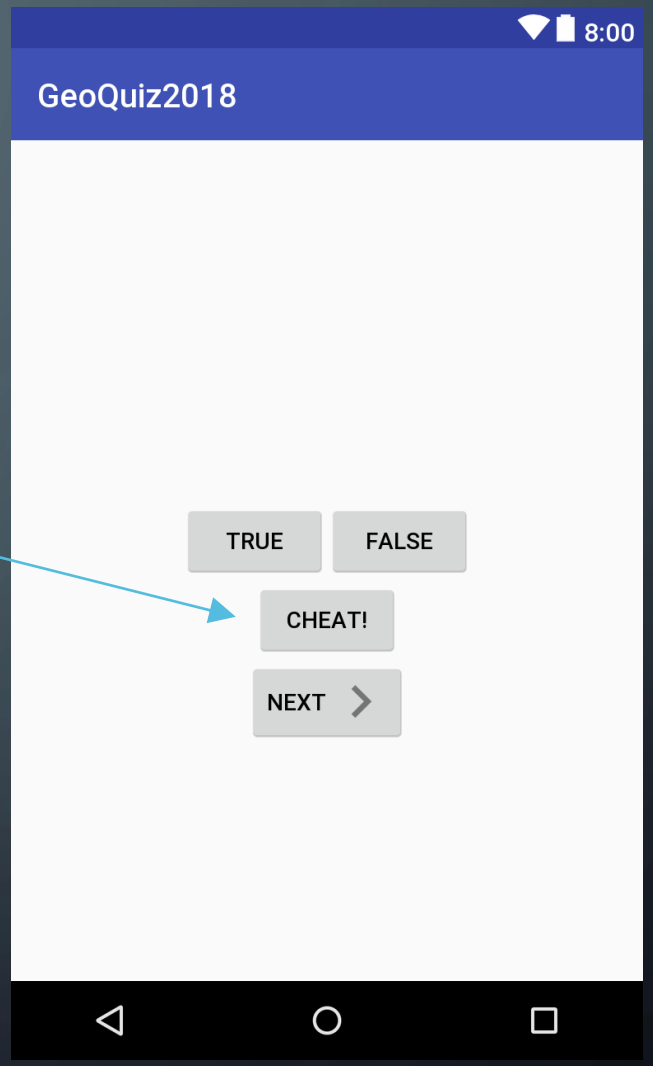
```
<LinearLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:gravity="center"
    android:orientation="vertical"

tools:context="com.example.pfgarofa.geoquiz2018.CheatActivity">

    <TextView
        android:id="@+id/textView"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/warning_text"
        android:textSize="24sp" />

    <TextView
        android:id="@+id/answer_text_view"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="TextView"
        android:textSize="24sp"
        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>
```

DECLARING AN ACTIVITY IN THE MANIFEST FILE

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.pfgarofa.geoquiz2018">

    <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/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>
```

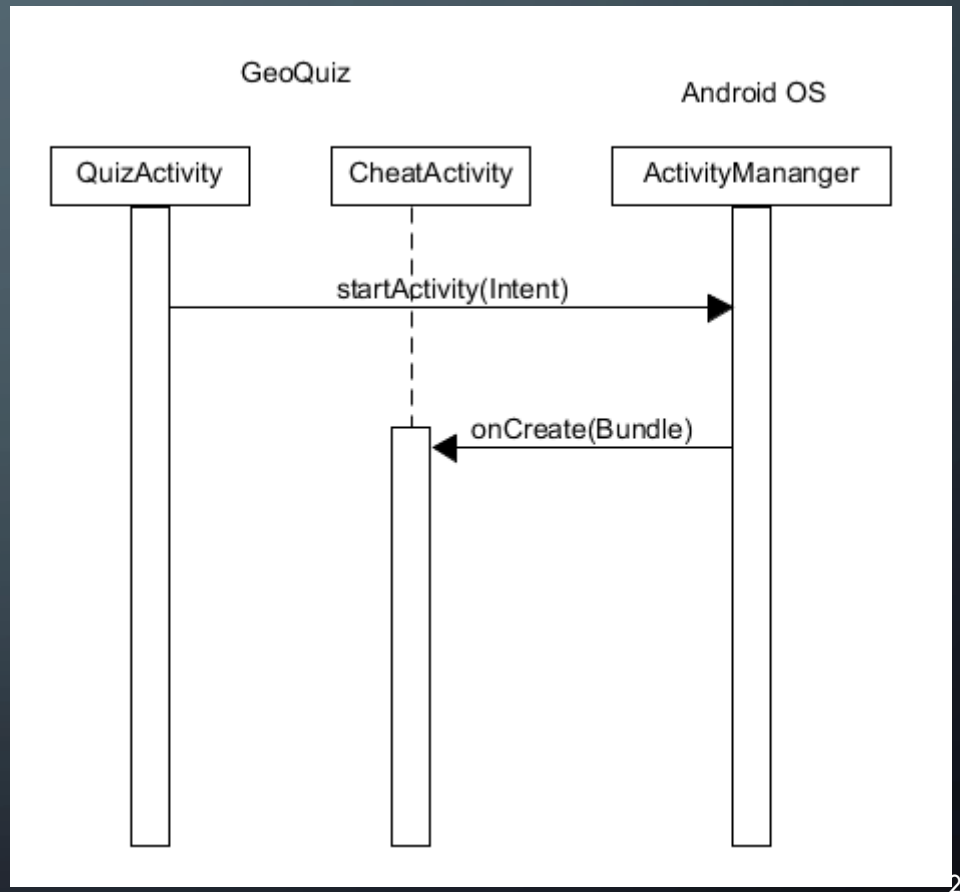
ADD A CHEAT BUTTON

- Add a “cheat” button to the main activity, activity_quiz.xml.
- Place it between the True and False buttons and the Next button.
- Change its width to wrap_content.
- Set its id to cheat_button.
- Set its text to cheat_button.



HOW TO START ANOTHER ACTIVITY

- You start another activity by calling startActivity(Intent intent).
- It takes an Intent object for an argument. The intent is passed to the OS.
- The new activity is started by Android's ActivityManager.
- We specify the name of the activity in the intent.



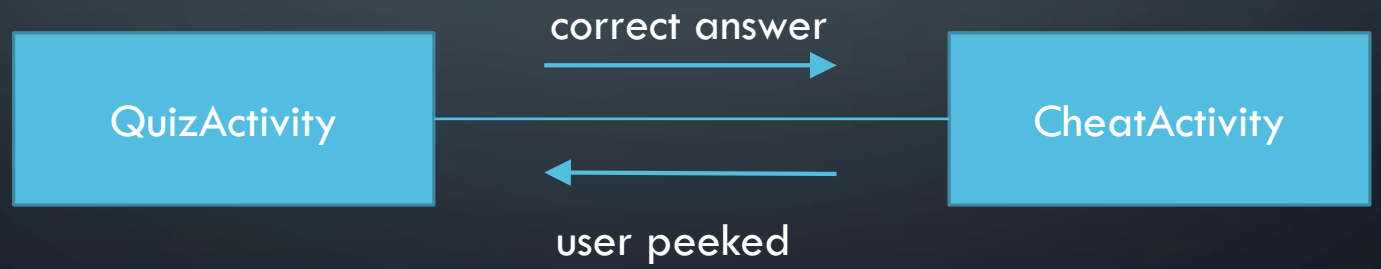
COMMUNICATING WITH INTENTS

- Wire up the next button and call startActivity() from the onClickListener.

```
public class QuizActivity extends AppCompatActivity {
    ...
    private Button mNextButton;
    private Button mCheatButton;
    ...
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        ...
        mCheatButton = (Button) findViewById(R.id.cheat_button);
        mCheatButton.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                Intent intent = new Intent(QuizActivity.this, CheatActivity.class);
                startActivity(intent);
            }
        });
        ...
```

PASSING DATA BETWEEN ACTIVITIES

- Here's how the app works. QuizActivity passes a Boolean indicating the correct answer to CheatAcitivity.
- CheatActivity returns an indication whether the user peeked at the correct answer.



USING INTENT EXTRAS

- We can add “extra” data to an intent, the data to pass.
- We’re going to pass the value of `mQuestionBank[mCurrentIndex].isAnswerTrue()`.
- Extra data is in a map or dictionary format like a Bundle: key-value pairs.
- To add an extra we call `Intent.putExtra(key, value)`.

IMPLEMENTING THE INTENT WITH EXTRA DATA

- We're going to put the creation of the intent in CheatActivity.

```
public class CheatActivity extends AppCompatActivity {

  private static final String EXTRA_CORRECT_ANSWER =
     "com.example.pfgarofa.geoquiz2018.correct_answer";

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

  ...
```

IMPLEMENTING THE INTENT WITH EXTRA DATA (2)

- Now change the Intent creation in QuizActivity.java.

```
mCheatButton = (Button) findViewById(R.id.cheat_button);
mCheatButton.setOnClickListener(new View.OnClickListener() {
  @Override
  public void onClick(View v) {
    // removed previous new Intent() call
    boolean correctAnswer = mQuestionBank[mCurrentIndex].isAnswerTrue();
    Intent intent = CheatActivity.newIntent(QuizActivity.this,
      correctAnswer);
    startActivity(intent);
  }
});
```

IMPLEMENTING THE INTENT WITH EXTRA DATA (3)

```
public class CheatActivity extends AppCompatActivity {
    ...
    private TextView mAnswerTextView;
    private Button mShowAnswerButton;

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

        mCorrectAnswer = getIntent().getBooleanExtra(EXTRA_CORRECT_ANSWER, false);

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

        mShowAnswerButton = (Button) findViewById(R.id.show_answer_button);
        mShowAnswerButton.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                mAnswerTextView.setText(mCorrectAnswer ? R.string.true_button : R.string.false_button);
            }
        });
    }
```

getIntent returns the intent the activity was called with.

Default value

RUN IT!

Now run the app and try out the cheat screen!

Use the back button to return to the question.

CHEATING IS EASY. AVOID EASY.

- Cheaters never win, so let's fix the app by letting QuizActivity know if the user peeked at the answer.
- We will modify QuizActivity to request a return result from CheatActivity with
  - `startActivityForResult(intent, request_code)`.
- We will also modify CheatActivity to send the return value with one of
  - `setResult(result_code)`
  - `setResult(result_code, intent)`

GETTING A CHILD ACTIVITY'S RESULT

```
public class QuizActivity extends AppCompatActivity {
    ...
    private static final String KEY_INDEX = "index";
    private static final int REQUEST_CODE_CHEAT = 0;

    ...
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        ...
        mCheatButton.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                boolean correctAnswer = mQuestionBank[mCurrentIndex].isAnswerTrue();
                Intent intent = CheatActivity.newIntent(QuizActivity.this, correctAnswer);
                startActivityForResult(intent, REQUEST_CODE_CHEAT);
            }
         });
```

SENDING BACK AN INTENT

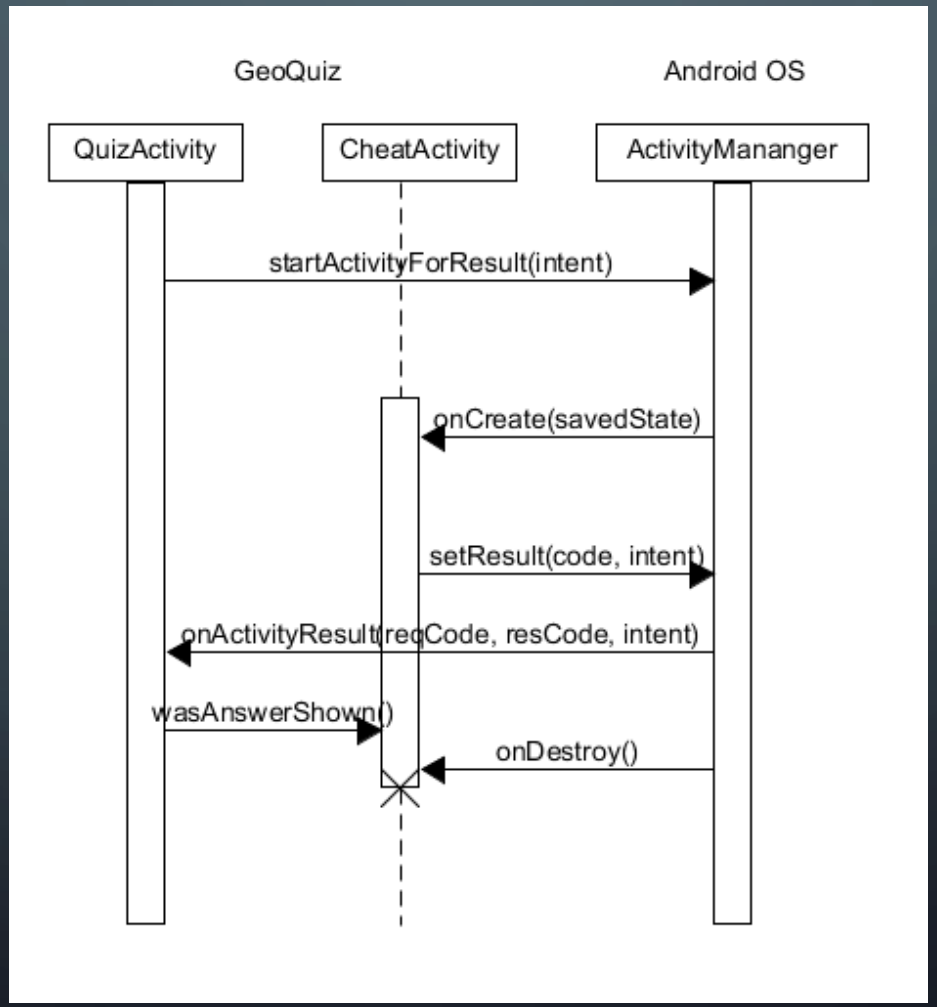
```
public class CheatActivity extends AppCompatActivity {
  ...
  private static final String EXTRA_ANSWER_SHOWN = "com.example.pfgarofa.geoquiz2018.answer_shown";
  ...
 @Override
    protected void onCreate(Bundle savedInstanceState) {
    ...
    mShowAnswerButton.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                mAnswerTextView.setText(mCorrectAnswer ? R.string.true_button : R.string.false_button);
                setAnswerShowResult(true);
            }
        });
    }

    private void setAnswerShowResult(boolean isAnswerShown) {
        Intent data = new Intent();
        data.putExtra(EXTRA_ANSWER_SHOWN, isAnswerShown);
        setResult(RESULT_OK, data);
    }
}
```

SENDING BACK AN INTENT (PART 2)

- One more addition to CheatActivity:

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

public static boolean wasAnswerShown(Intent result) {
  return result.getBooleanExtra(EXTRA_ANSWER_SHOWN, false);
}
```

RETURNING THE RESULT SEQUENCE



HANDLING THE RESULT

```
public class QuizActivity extends AppCompatActivity {
  ...
  private int mCurrentIndex = 0;
  private boolean mIsCheater;
  ...
  @Override
  protected void onActivityResult(int requestCode, int resultCode, Intent data) {
    if (resultCode != Activity.RESULT_OK) {
      return;
    }

    if (requestCode == REQUEST_CODE_CHEAT) {
      if (data == null) {
        return;
      }
      mIsCheater = CheatActivity.wasAnswerShown(data);
    }
  }
```

CHECK THE ANSWER

```
public class QuizActivity extends AppCompatActivity {
  ...
  @Override
    protected void onCreate(Bundle savedInstanceState) {
    ...
    mNextButton.setOnClickListener(new View.OnClickListener() {
      @Override
      public void onClick(View v) {
        mCurrentIndex = (mCurrentIndex + 1) % mQuestionBank.length;
        mIsCheater = false;
        updateQuestion();
      }
    });
  ...
```

CHECK THE ANSWER (PART 2)

- Add the following test to QuizActivity.checkAnswer():

```
private void checkAnswer(boolean userChoice) {
    boolean correctAnswer = mQuestionBank[mCurrentIndex].isAnswerTrue();

    int messageResId = 0;

    if (mIsCheater) {
        messageResId = R.string.judgement_toast;
    } else {
        if (userChoice == correctAnswer) {
            messageResId = R.string.correct_answer;
        } else {
            messageResId = R.string.incorrect_answer;
        }
    }

    Toast.makeText(this, messageResId, Toast.LENGTH_SHORT).show();
}
```

RUN THE APP AGAIN

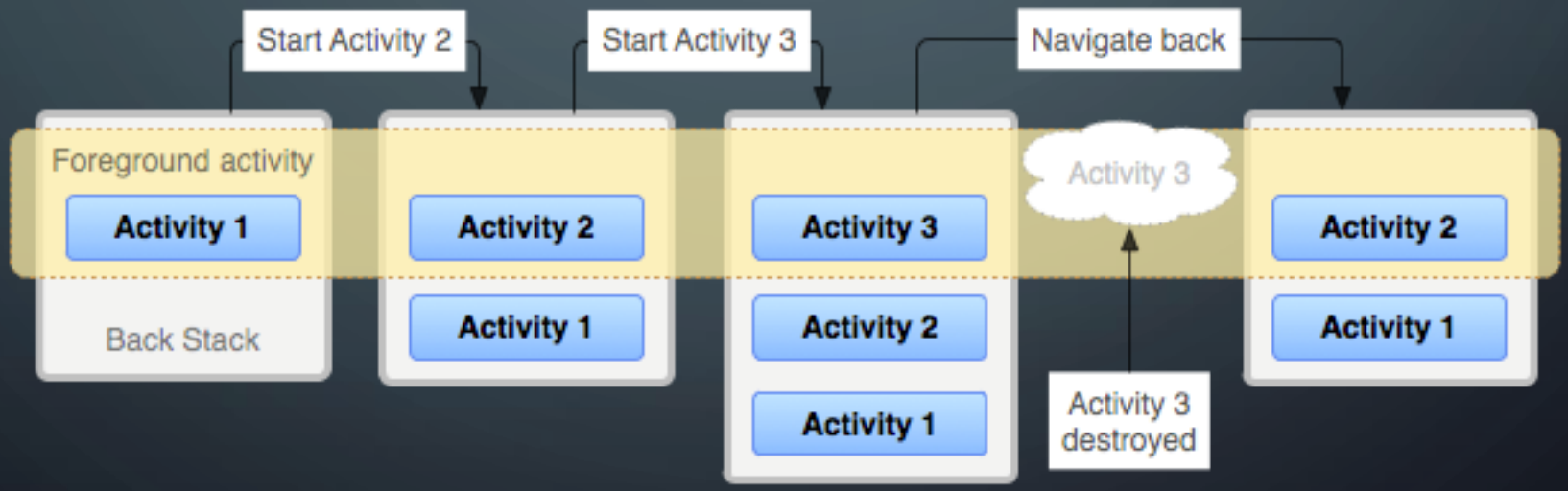
Test the cheat check!

ANDROID TASKS

- A **task** *is a collection of activities* that users interact with when performing a certain job.
- The *activities are arranged in a stack*—**the *back stack***—in the order in which each activity is opened.

BACK STACK

For example, an email app might have one activity to show a list of new messages. When the user selects a message, a new activity opens to view that message. This new activity is added to the back stack. If the user presses the Back button, that new activity is finished and popped off the stack.



ADDING CODE FROM LATER APIS

```
mShowAnswerButton.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        mAnswerTextView.setText(mCorrectAnswer ? R.string.true_button : R.string.false_button);
        setAnswerShowResult(true);

        int cx = mShowAnswerButton.getWidth() / 2;
        int cy = mShowAnswerButton.getHeight() / 2;
        float radius = mShowAnswerButton.getWidth();

        Animator anim = ViewAnimationUtils.createCircularReveal(mShowAnswerButton,
                cx, cy, radius, 0);
        anim.addListener(new AnimatorListenerAdapter() {
            @Override
            public void onAnimationEnd(Animator animation) {
                super.onAnimationEnd();
                mShowAnswerButton.setVisibility(View.INVISIBLE);
            }
        });
        anim.start();
    }
});
```

ADDING CODE FROM LATER APIS

```
mShowAnswerButton.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        mAnswerTextView.setText(mCorrectAnswer ? R.string.true_button : R.string.false_button);
        setAnswerShowResult(true);

        if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.LOLLIPOP) {
            int cx = mShowAnswerButton.getWidth() / 2;
            int cy = mShowAnswerButton.getHeight() / 2;
            float radius = mShowAnswerButton.getWidth();

            final Animator anim = ViewAnimationUtils.createCircularReveal(mShowAnswerButton,
                    cx, cy, radius, 0);
            anim.addListener(new AnimatorListenerAdapter() {
                @Override
                public void onAnimationEnd(Animator animation) {
                    super.onAnimationEnd(animation);
                    mShowAnswerButton.setVisibility(View.INVISIBLE);
                }
            });
            anim.start();
        } else {
            mShowAnswerButton.setVisibility(View.INVISIBLE);
        }
    }
});
```