Android Basics
Introduction to Android

Javed Hasan

BJIT Limited

Introduction to Android

Android App Basics

- Your First Android Application GeoQuiz
- Create Android Project with Eclipse IDE

UI Layout Basics

- View Hierarchy
- Widget Attributes
- Creating String Resources

Resources and Resource Ids

- Adding Ids to button resources
- Wiring up widgets in Android code
- Making Toast Messages

Running Android App on Emulator

Learning Android

As beginning Android programmer, you face a steep learning curve.

Android has a culture. That culture speaks Java, but knowing Java is not enough. Getting your head around Android requires learning many new ideas and techniques.

To be an Android programmer, you must:

- write Android applications
- understand what you are writing

Your First Android Application

The application we are going to create is called GeoQuiz.

GeoQuiz tests the user's knowledge of geography.

The user presses True or False to answer the question on screen, and GeoQuiz provides instant feedback.

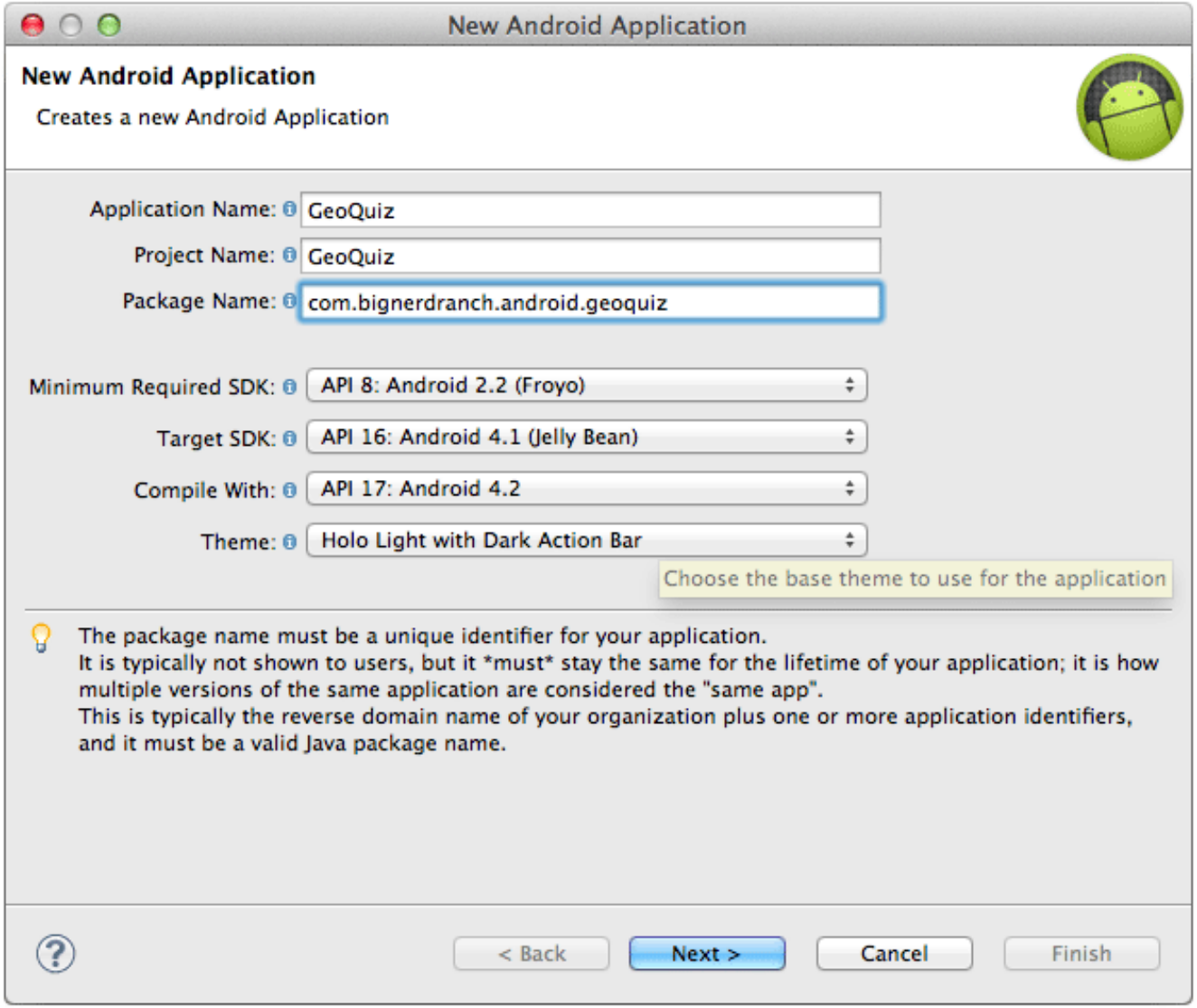
App Basics

GeoQuiz application will consist of an activity and a layout:

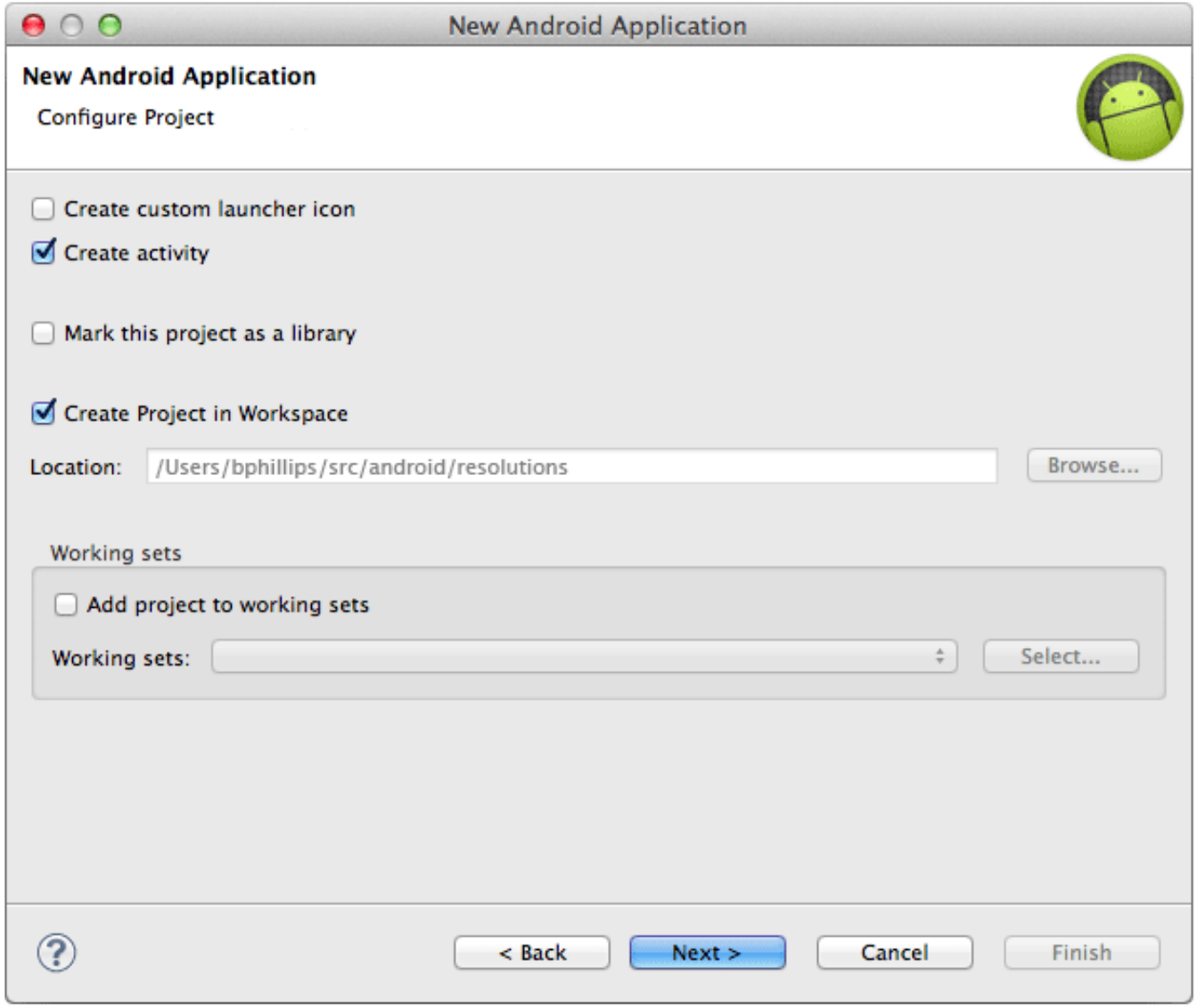
An **activity** is an instance of Activity, a class in the Android SDK. An activity is responsible for managing user interaction with a screen of information.

A **layout** defines a set of user interface objects and their position on the screen.

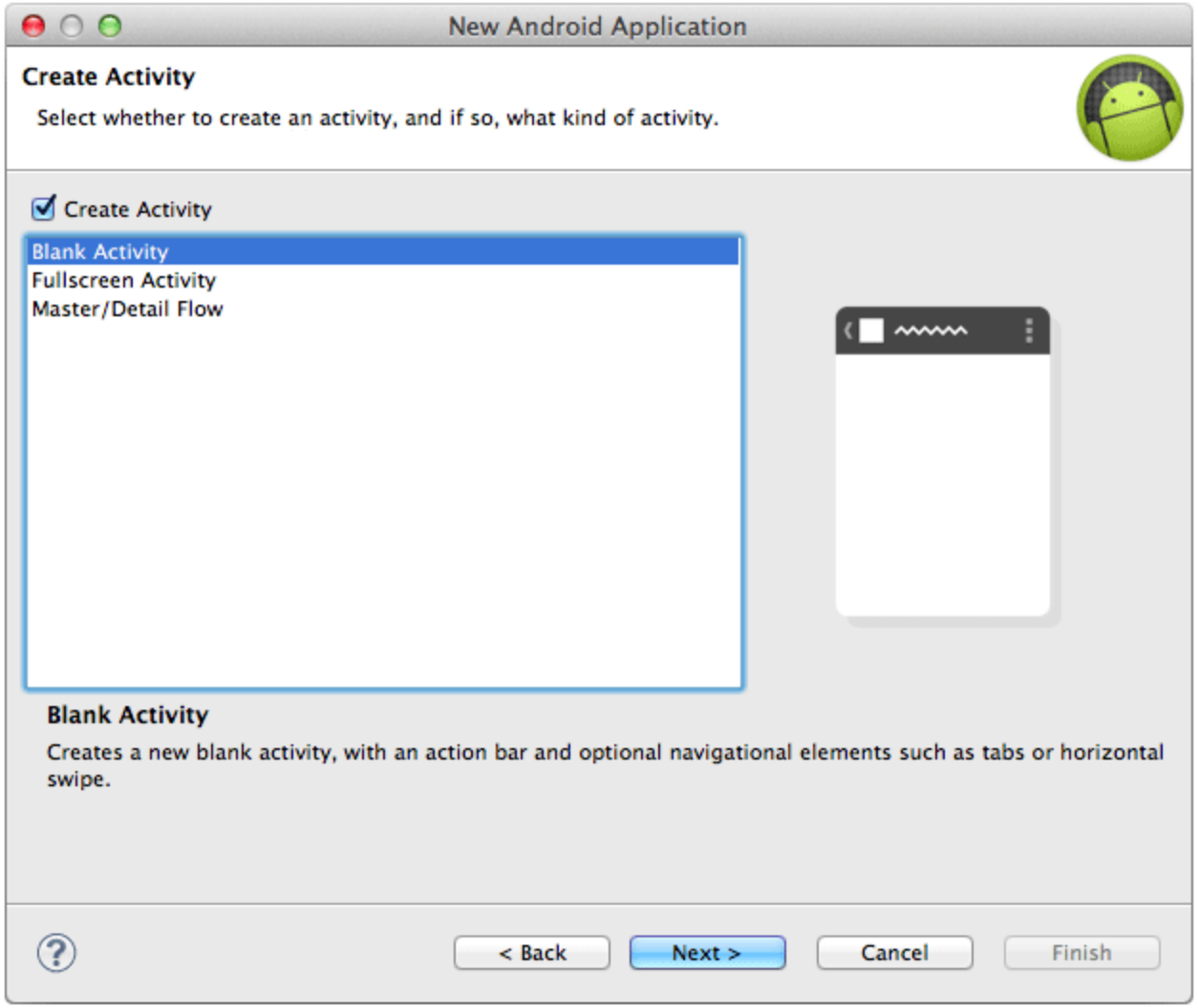
Creating Android Project



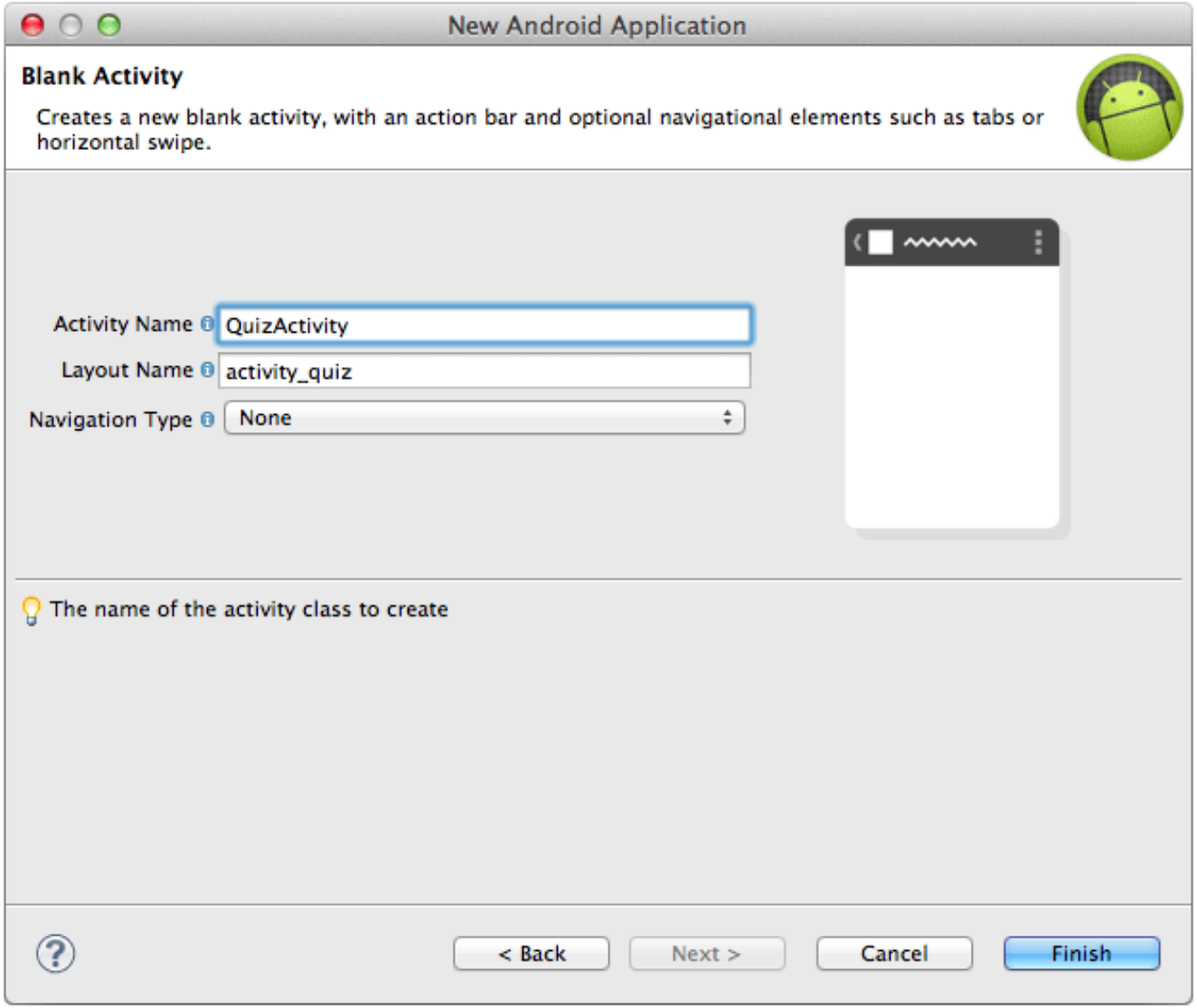
Configuring Android Project



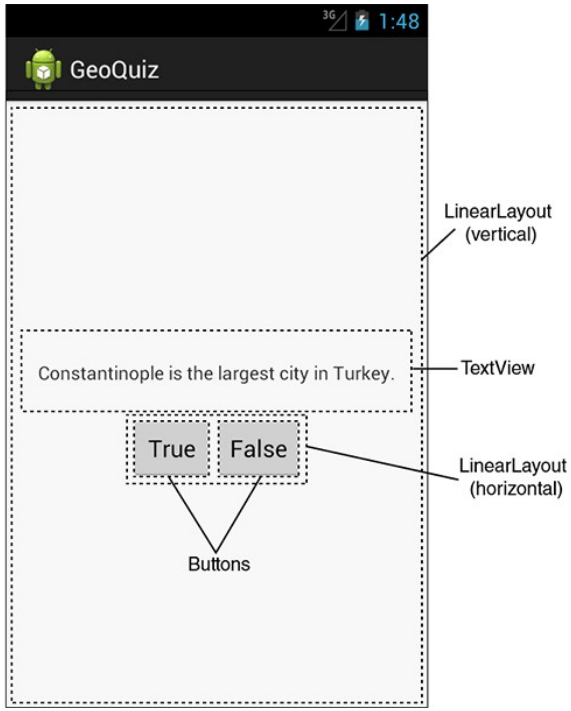
Creating a New Activity

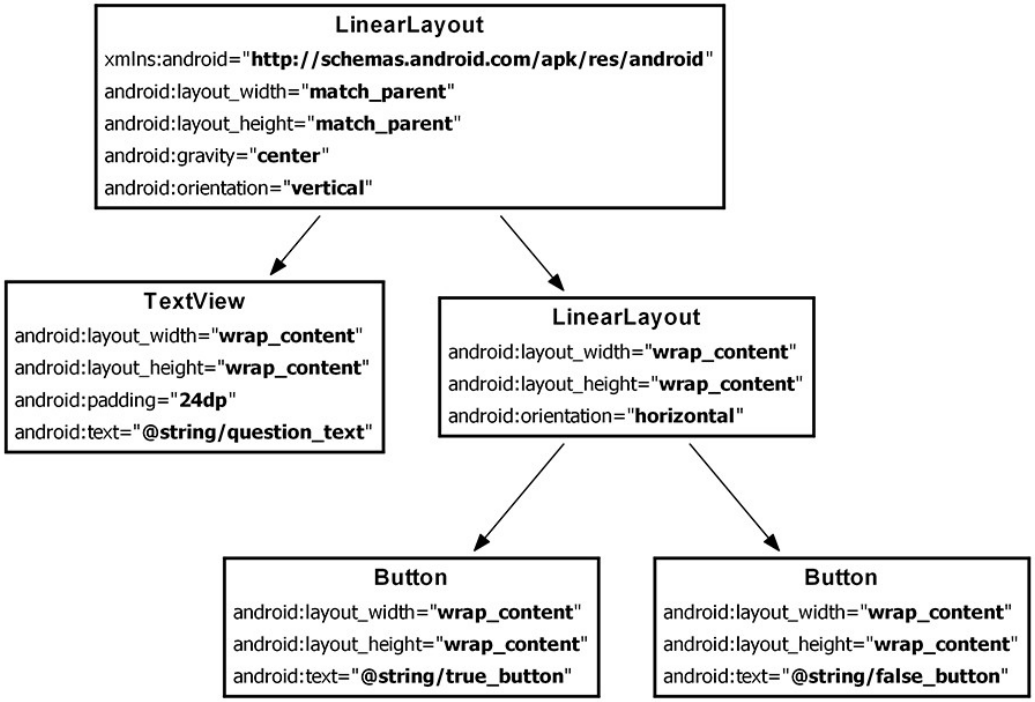


Configuring the New Activity



Laying Out the User Interface





The View Hierarchy

```
<LinearLayout
        xmlns:android="http://schemas.android.com/apk/res/android
  android:layout_width="match_parent"
  android:layout_height="match_parent"
  android:gravity="center"
  android:orientation="vertical" >
  <TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:padding="24dp"
    android:text="@string/question_text" />
```

```
<LinearLayout
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:orientation="horizontal" >

      <Button
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:text="@string/true_button" />

      <Button
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:text="@string/false_button" />
  </LinearLayout>

</LinearLayout>
```

Widget Attributes

android:layout_width and android:layout_height

Have values set one of the below:

match_parent: view will be as big as its parent

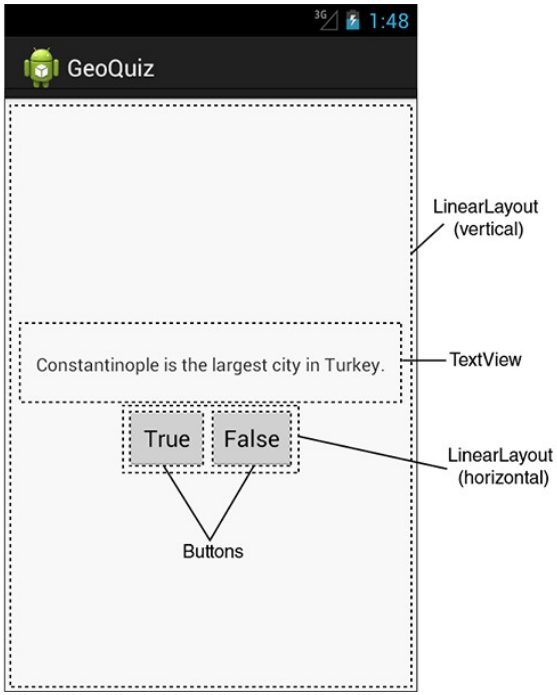
wrap_content: view will be as big as its contents require

android:orientation

Determines if their children will appear vertically or horizontally.

In a vertical LinearLayout, the first child defined will appear topmost.

In a horizontal LinearLayout, the first child defined will be leftmost.



android:text

The **TextView** and **Button** widgets have android:text attributes to display text content.

Creating String Resources

Every project includes a default strings file named strings.xml.

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

  <string name="app_name">GeoQuiz</string>
  <string name="hello_world">Hello, world!</string>
  <string name="question_text">
        Constantinople is the largest city in Turkey.</string>
  <string name="true_button">True</string>
  <string name="false_button">False</string>
  <string name="menu_settings">Settings</string>

</resources>
```

From Layout XML to View Objects

```
package com.bignerdranch.android.geoquiz;

import android.app.Activity;
import android.os.Bundle;
import android.view.Menu;

public class QuizActivity extends Activity {
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_quiz);
    }
    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        getMenuInflater().inflate(R.menu.activity_quiz, menu);
        return true;
    }
}
```

The **onCreate(Bundle)** method is called when an instance of the activity subclass is created.

When an activity is created, it needs a user interface to manage. To get the activity its user interface, you call the following Activity method:

```
public void setContentView(int layoutResID)
```

This method inflates a layout and puts it on screen.

When a layout is inflated, each widget in the layout file is instantiated as defined by its attributes. You specify which layout to inflate by passing in the layout's resource ID.

Resources and resource IDs

A layout is a resource.

A resource is a piece of your application that is not code – things like image files, audio files, and XML files.

Resources for your project live in a subdirectory of the res directory. In the package explorer, you can see that *activity_quiz.xml* lives in *res/layout/*. Your strings file, which contains string resources, lives in *res/values/*.

To access a resource in code, you use its resource ID. The resource ID for your layout is *R.layout.activity_quiz*. For example,

```
setTitle(R.string.app_name);
```

```
/* AUTO-GENERATED FILE.  DO NOT MODIFY … */

package com.bignerdranch.android.geoquiz;
public final class R {
    public static final class attr {}
    public static final class drawable {
        public static final int ic_launcher=0x7f020000;  }
    public static final class id {
        public static final int menu_settings=0x7f070003;   }
    public static final class layout {
        public static final int activity_quiz=0x7f030000; }
    public static final class menu {
       public static final int activity_quiz=0x7f060000;
    }
    public static final class string {
        public static final int app_name=0x7f040000;
        public static final int false_button=0x7f040003;
        public static final int menu_settings=0x7f040006;
        public static final int question_text=0x7f040001;
        public static final int true_button=0x7f040002;
    }

    …
}
```

Adding IDs to buttons (activity_quiz.xml)

```
<LinearLayout
xmlns:android="http://schemas.android.com/apk/res/android"
... >
  <TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:padding="24dp"
    android:text="@string/question_text" />
  <LinearLayout
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:orientation="horizontal">
    <Button
      android:id="@+id/true_button"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:text="@string/true_button" />
```

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

To generate a resource ID for a widget, you include an android:id attribute in the widget's definition.
Check R.java to confirm that you have two new resource IDs in the R.id inner class.

```
public final class R {
   ...
   public static final class id {
      public static final int false_button=0x7f070001;
      public static final int menu_settings=0x7f070002;
      public static final int true_button=0x7f070000;
   }
...
```

Wiring Up Widgets

Getting references to widgets

```
public class QuizActivity extends Activity {

    private Button mTrueButton;
    private Button mFalseButton;
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_quiz);

        mTrueButton = (Button)findViewById(R.id.true_button);
        mFalseButton = (Button)findViewById(R.id.false_button);
    } ...}
```

Set listener for True and False button

```
@Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_quiz);

        mTrueButton = (Button)findViewById(R.id.true_button);
        mTrueButton.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                // Does nothing yet, but soon!
            }
        });
        mFalseButton = (Button)findViewById(R.id.false_button);
        mFalseButton.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                // Does nothing yet, but soon!
            }
        });
    }
}
```

Making Toasts

A toast is a short message that informs the user of something but does not require any input or action.

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
  <string name="app_name">GeoQuiz</string>
  <string name="question_text">Constantinople is the
largest city in Turkey.</string>
  <string name="true_button">True</string>
  <string name="false_button">False</string>
  <string name="correct_toast">Correct!</string>
  <string name="incorrect_toast">Incorrect!</string>
  <string name="menu_settings">Settings</string>
</resources>
```

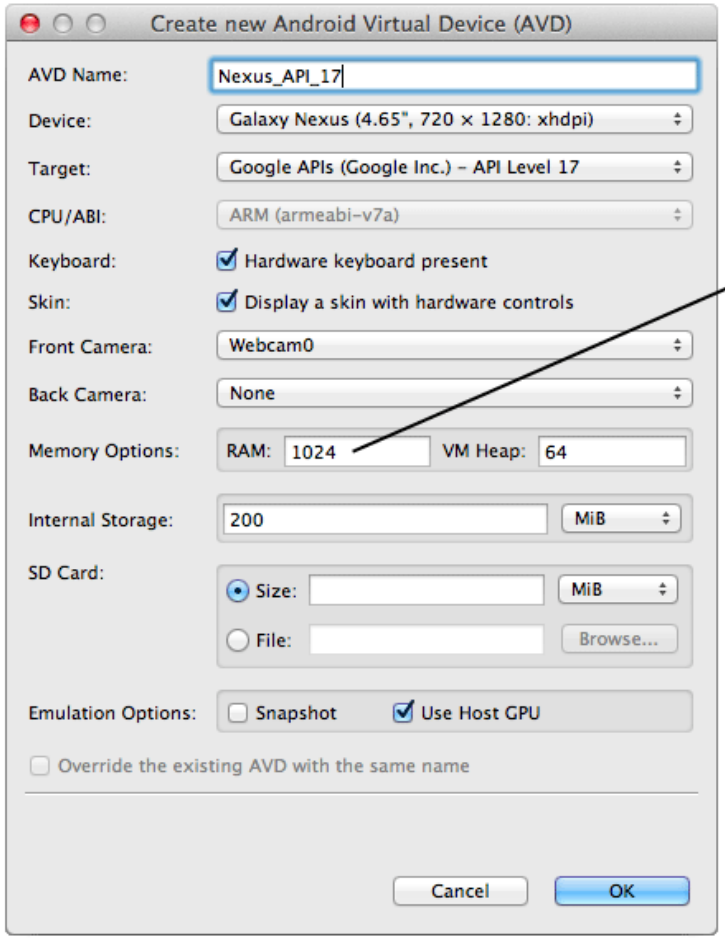
```
mTrueButton.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        Toast.makeText(QuizActivity.this,
                R.string.incorrect_toast,
                Toast.LENGTH_SHORT).show();
    }
});
```

Running on the Emulator

To create an Android virtual device (AVD), choose Window → Android Virtual Device Manager.

When the AVD Manager appears, click the New... button on the righthand side of the window.

In the package explorer, right-click the GeoQuiz project folder. From the context menu, choose Run As → Android Application.



Android Basics
Android and Model-View-Controller

Javed Hasan

BJIT Limited

Android and Model-View-Controller

New Features

- Multiple Questions Support

Model-View-Controller (MVC) and Android

- MVC flow with User Input
- Object Diagram of GeoQuiz App

MVC Pattern Applied in GeoQuiz App

- Adding Ids to button resources
- Wiring up widgets in Android code
- Making Toast Messages

Running Android App on Emulator

New Features of GeoQuiz

Model-View-Controller and Android

Android applications are designed around an architecture called Model-View-Controller.

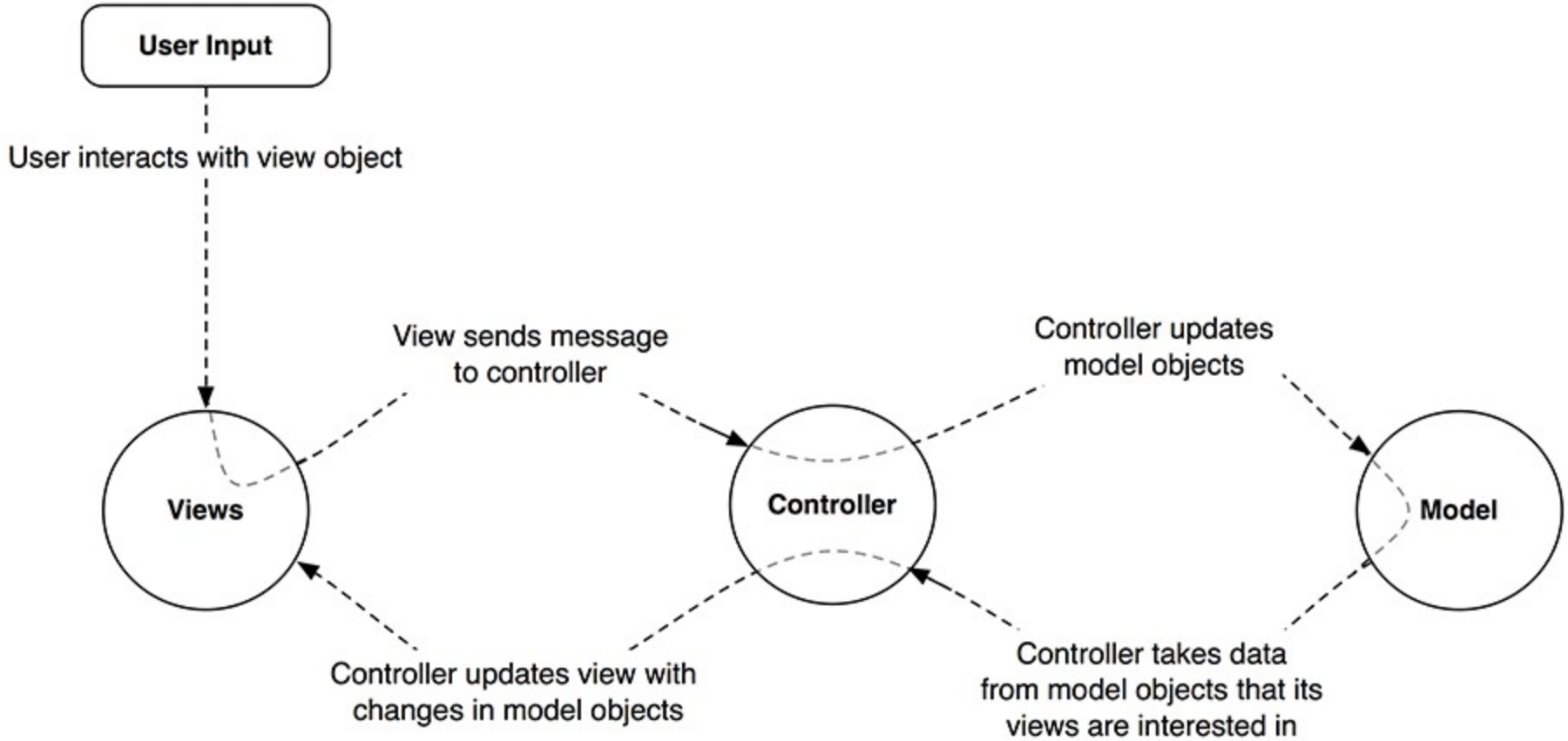
A *model* object holds the application's data and "business logic." In Android applications, model classes are generally custom classes you create.

View objects know how to draw themselves on the screen and how to respond to user input, like touches.

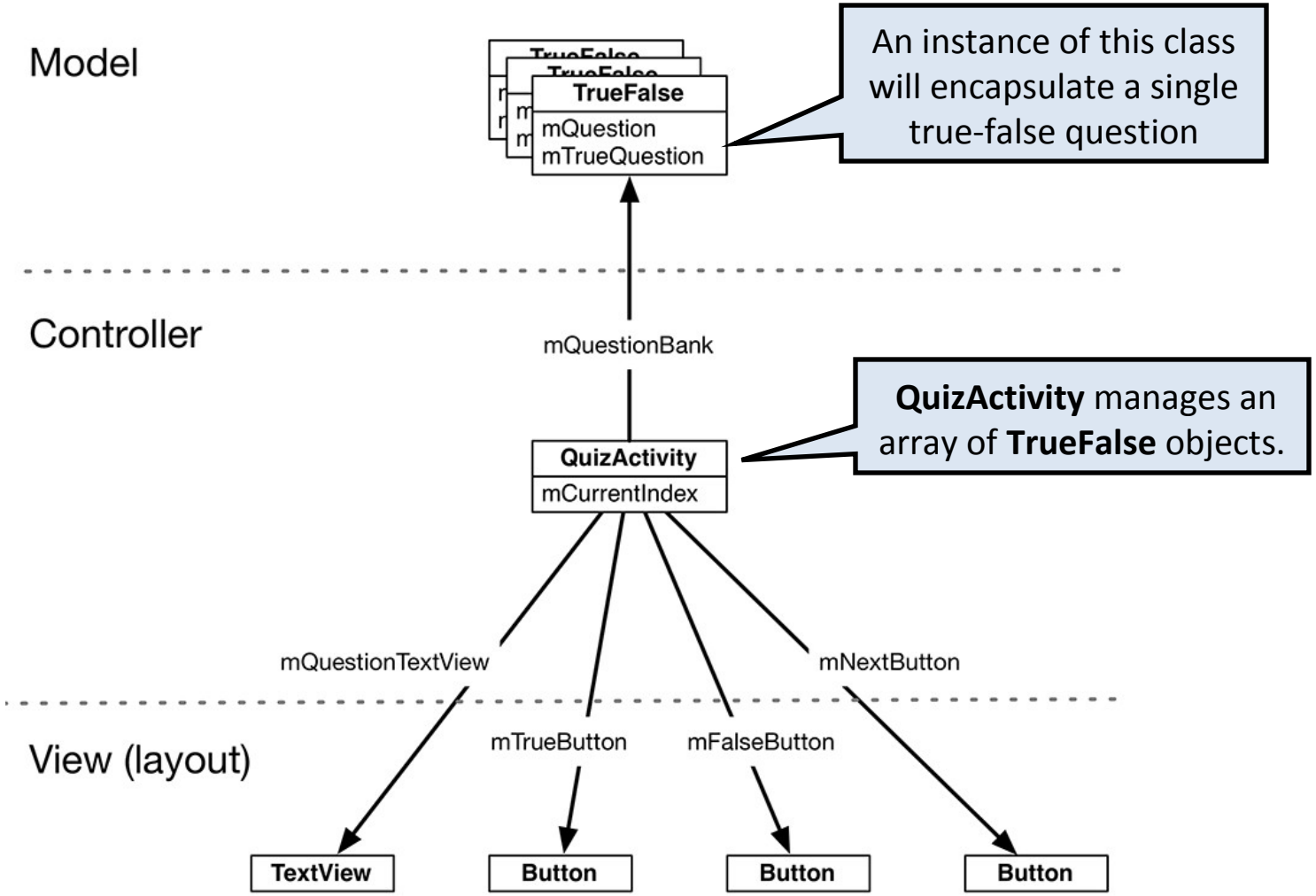
Controller objects tie the view and model objects together. They contain "application logic."

Controllers are designed to respond to various events triggered by view objects and to manage the flow of data to and from model objects and the view layer.

MVC flow with user input

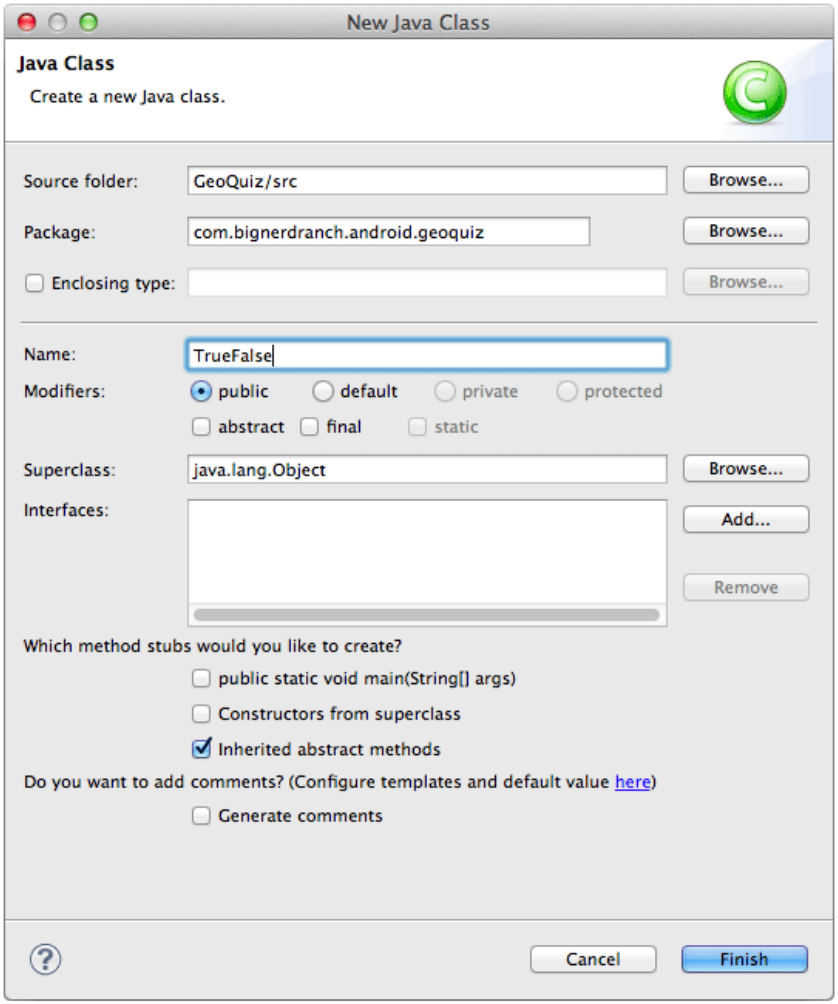


Object Diagram GeoQuiz Application

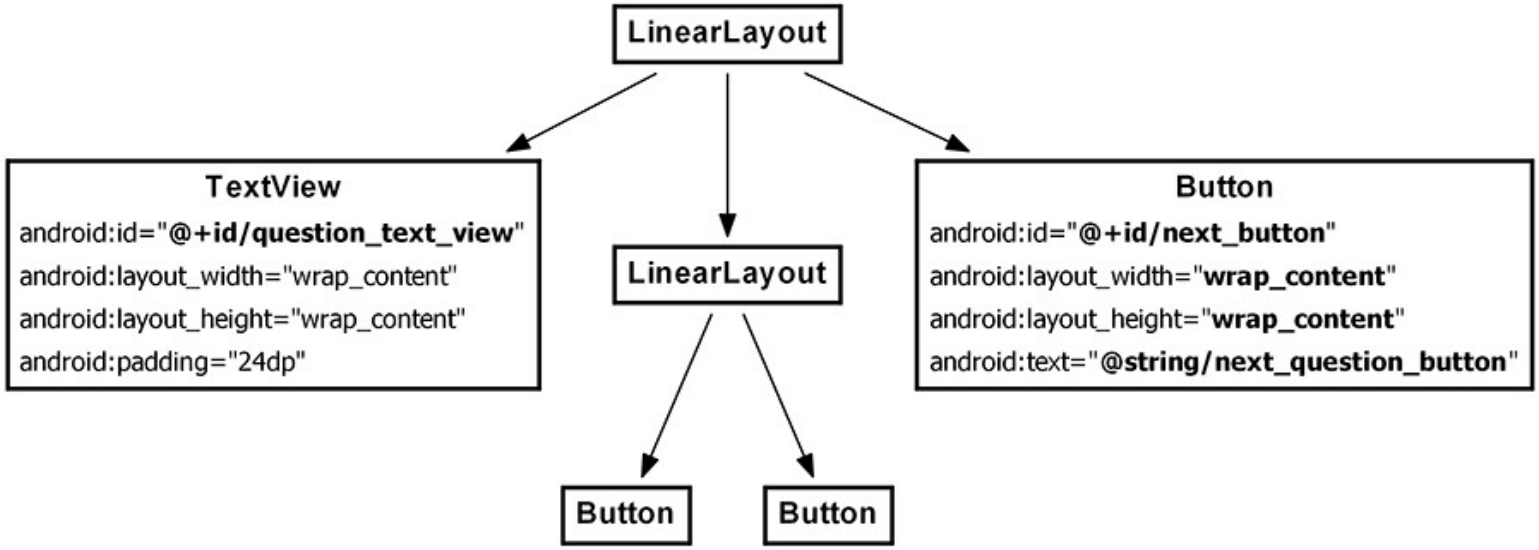


Model: TrueFalse Class

```
public class TrueFalse {
    private int mQuestion;
    private boolean mTrueQuestion;

    public TrueFalse(int question, boolean trueQuestion) {
        mQuestion = question;
        mTrueQuestion = trueQuestion;
    }
    public int getQuestion() {
        return mQuestion;
    }
    public void setQuestion(int question) {
        mQuestion = question;
    }
    public boolean isTrueQuestion() {
        return mTrueQuestion;
    }
    public void setTrueQuestion(boolean trueQuestion) {
        mTrueQuestion = trueQuestion;
    }
}
```



Views: GeoQuiz View Hierarchy



Adding Question Strings

```
…
  <string name="incorrect_toast">Incorrect!</string>
  <string name="menu_settings">Settings</string>
  <string name="question_oceans">
        The Pacific Ocean is larger than  the Atlantic Ocean.</string>
  <string name="question_mideast">
        The Suez Canal connects the Red Sea  and the Indian Ocean.</string>
  <string name="question_africa">The source of the Nile River is in Egypt.</string>
  <string name="question_americas">
        The Amazon River is the longest river in the Americas.</string>
  <string name="question_asia">
        Lake Baikal is the world\'s oldest and deepest freshwater lake.</string>
...
```

Controller: QuizActivity

You have multiple questions to retrieve and display, QuizActivity will have to work harder to tie GeoQuiz's model and view layers together.

```
public class QuizActivity extends Activity {

    private Button mTrueButton;
    private Button mFalseButton;
    private Button mNextButton;
    private TextView mQuestionTextView;

    private TrueFalse[] mQuestionBank = new TrueFalse[] {
        new TrueFalse(R.string.question_oceans, true),
        new TrueFalse(R.string.question_mideast, false),
        new TrueFalse(R.string.question_africa, false)
        new TrueFalse(R.string.question_americas, true),
        new TrueFalse(R.string.question_asia, true),
    };

    private int mCurrentIndex = 0;
    ...
```

Wiring Up Widgets: TextView 1/2

```
public class QuizActivity extends Activity {
    ...

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

        mQuestionTextView = (TextView)findViewById(R.id.question_text_view);
        int question = mQuestionBank[mCurrentIndex].getQuestion();
        mQuestionTextView.setText(question);

        mTrueButton = (Button)findViewById(R.id.true_button);
        ...
    }
}
```

Wiring Up Widgets: Button 2/2

```
public class QuizActivity extends Activity {
  ...
  protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_quiz);
    mQuestionTextView =
      (TextView)findViewById(R.id.question_text_view);
    int question =
       mQuestionBank[mCurrentIndex].getQuestion();
    mQuestionTextView.setText(question);
    ...
    mFalseButton.setOnClickListener(new
       View.OnClickListener() {
      public void onClick(View v) {
        Toast.makeText(QuizActivity.this,
            R.string.correct_toast,
            Toast.LENGTH_SHORT).show();}
    });
```

```
mNextButton = (Button)findViewById(R.id.next_button);
    mNextButton.setOnClickListener(new View.OnClickListener() {
      @Override
      public void onClick(View v) {
        mCurrentIndex =
            (mCurrentIndex + 1) % mQuestionBank.length;
        int question =
            mQuestionBank[mCurrentIndex].getQuestion();
        mQuestionTextView.setText(question);
      }  });}}
```

Code Refactoring 1/2

```
public class QuizActivity extends Activity {
  ...
  private void updateQuestion() {
    int question = mQuestionBank[mCurrentIndex].getQuestion();
    mQuestionTextView.setText(question);
  }
  protected void onCreate(Bundle savedInstanceState) {
    ...
    mQuestionTextView = (TextView)findViewById(R.id.question_text_view);
    int question = mQuestionBank[mCurrentIndex].getQuestion();
    mQuestionTextView.setText(question);

    mNextButton.setOnClickListener(new View.OnClickListener() {
        public void onClick(View v) {
            mCurrentIndex = (mCurrentIndex + 1) % mQuestionBank.length;
            int question = mQuestionBank[mCurrentIndex].getQuestion();
            mQuestionTextView.setText(question);
            updateQuestion();
        }
    });
    updateQuestion();
  }}
```

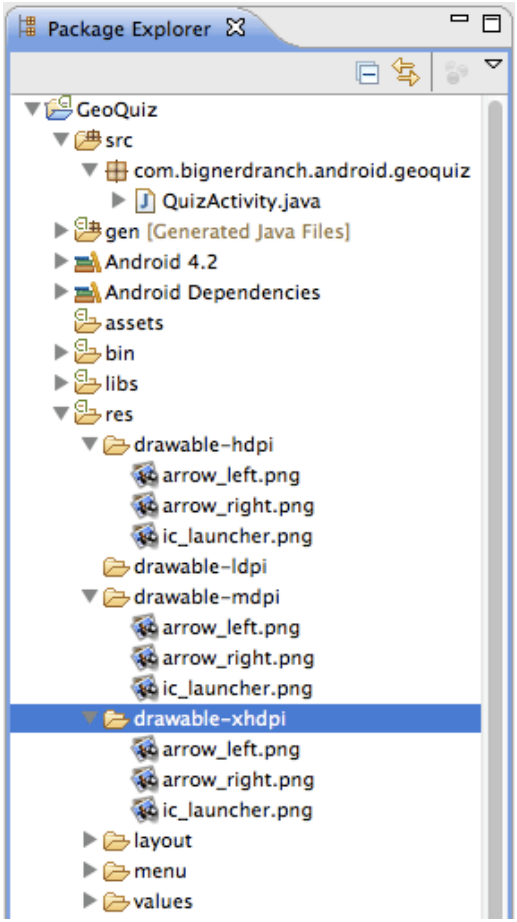
Code Refactoring 2/2

```
public class QuizActivity extends Activity {
    ...
    private void updateQuestion() {
        int question = mQuestionBank[mCurrentIndex].getQuestion();
        mQuestionTextView.setText(question);
    }
    private void checkAnswer(boolean userPressedTrue) {
        boolean answerIsTrue =
            mQuestionBank[mCurrentIndex].isTrueQuestion();
        int messageResId = 0;
        if (userPressedTrue == answerIsTrue) {
            messageResId = R.string.correct_toast;
        } else {
            messageResId = R.string.incorrect_toast;
        }
        Toast.makeText(this, messageResId, Toast.LENGTH_SHORT)
            .show();
    }
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        ...
    }}
```

```
public class QuizActivity extends Activity {
    ...
    protected void onCreate(Bundle savedInstanceState) {
        ...
        mTrueButton = (Button)findViewById(R.id.true_button);
        mTrueButton.setOnClickListener(new View.OnClickListener() {
            public void onClick(View v) {
                Toast.makeText(QuizActivity.this,
                        R.string.incorrect_toast,
                        Toast.LENGTH_SHORT).show();
                checkAnswer(true);
            }
        });
        mFalseButton = (Button)findViewById(R.id.false_button);
        mFalseButton.setOnClickListener(new View.OnClickListener() {
            public void onClick(View v) {
                Toast.makeText(QuizActivity.this,
                        R.string.correct_toast,
                        Toast.LENGTH_SHORT).show();
                checkAnswer(false);
            }});
        mNextButton = (Button)findViewById(R.id.next_button);
        ...
    }}
```

UI Enhancement – Adding an Icon

Arrow icons in GeoQuiz drawable directories



Referencing icon resources in XML

```
<LinearLayout
  ... >
  ...
  <LinearLayout
    ... >
    ...
  </LinearLayout>
  <Button
    android:id="@+id/next_button"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="@string/next_question_button"
    android:drawableRight="@drawable/arrow_right"
    android:drawablePadding="4dp"
    />
</LinearLayout>
```

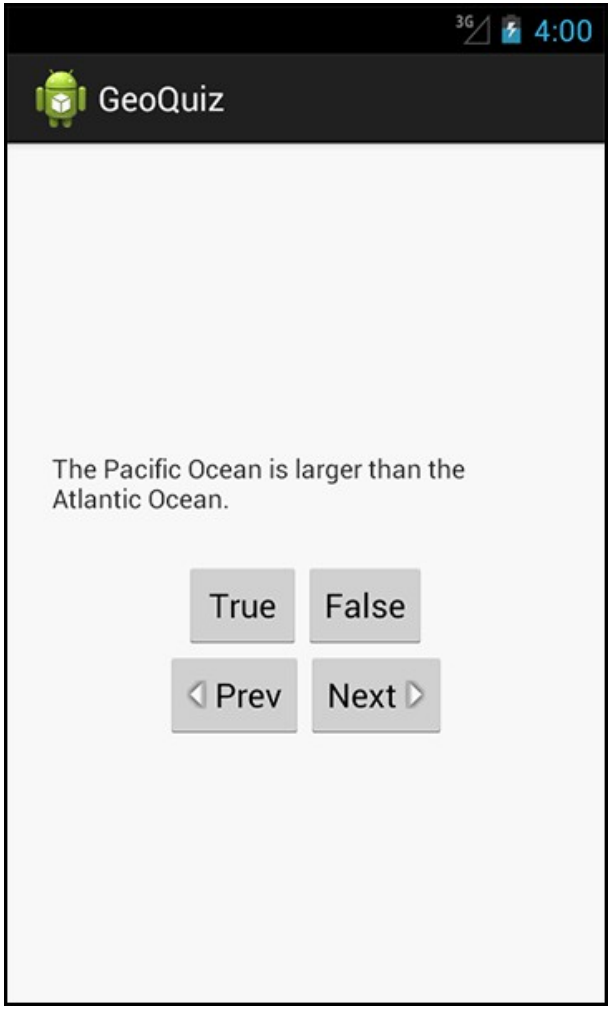
Challenge 1: Add a Listener to the TextView

Your Next button is nice, but you could also make it so that a user could press the `TextView` itself to see the next question.

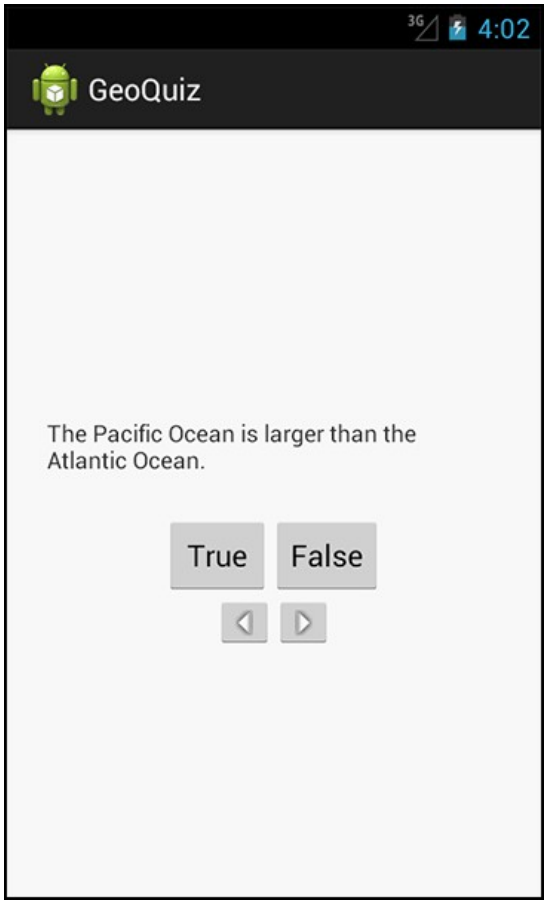
Hint: You can use the `View.OnClickListener` listener for the `TextView` that you have used with the `Button` because `TextView` also inherits from `View`.

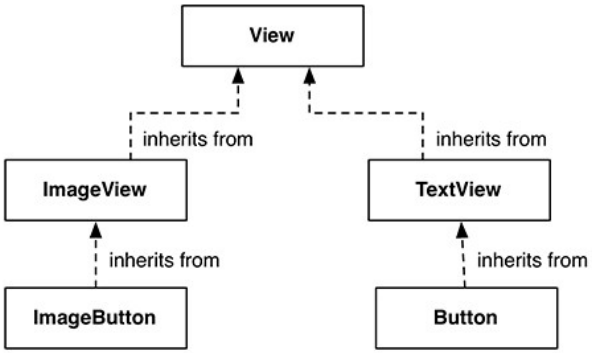
Challenge 2: Add a Previous Button

Add a button that the user can press to go back one question.



Challenge 3: From Button to ImageButton





You can replace the text and drawable attributes on the Next button with a single ImageView attribute:

```
<~~Button~~ ImageButton
  android:id="@+id/next_button"
  android:layout_width="wrap_content"
  android:layout_height="wrap_content"
  ~~android:text="@string/next_question_button"~~
  ~~android:drawableRight="@drawable/arrow_right"~~
  ~~android:drawablePadding="4dp"~~
  android:src="@drawable/arrow_right"
  />
```

Of course, you will need to modify QuizActivity to work with ImageButton.

Android Basics
Activity Lifecycle

Javed Hasan

BJIT Limited

Activity Lifecycle Overview

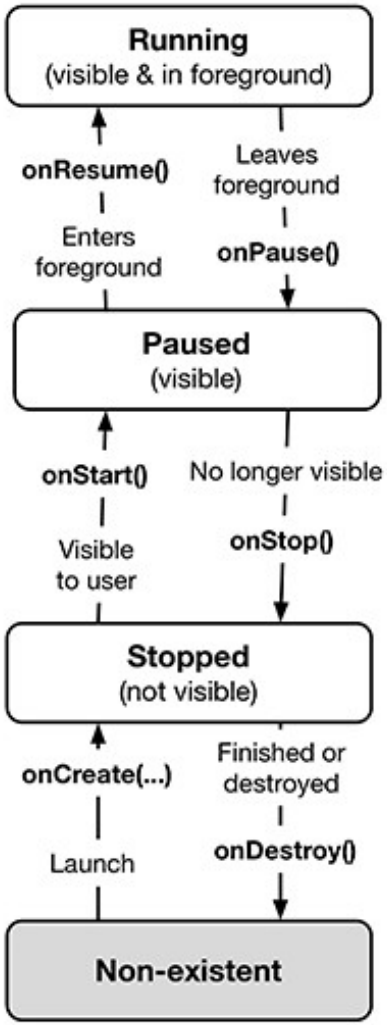
Every instance of Activity has a lifecycle.

During this lifecycle, an activity transitions between three possible states:

- Running
- Paused
- Stopped

For each transition, there is an Activity method that notifies the activity of the change in its state.

Never call onCreate(...) or any of the other Activity lifecycle methods yourself. You override them in your activity subclasses, and Android calls them at the appropriate time.



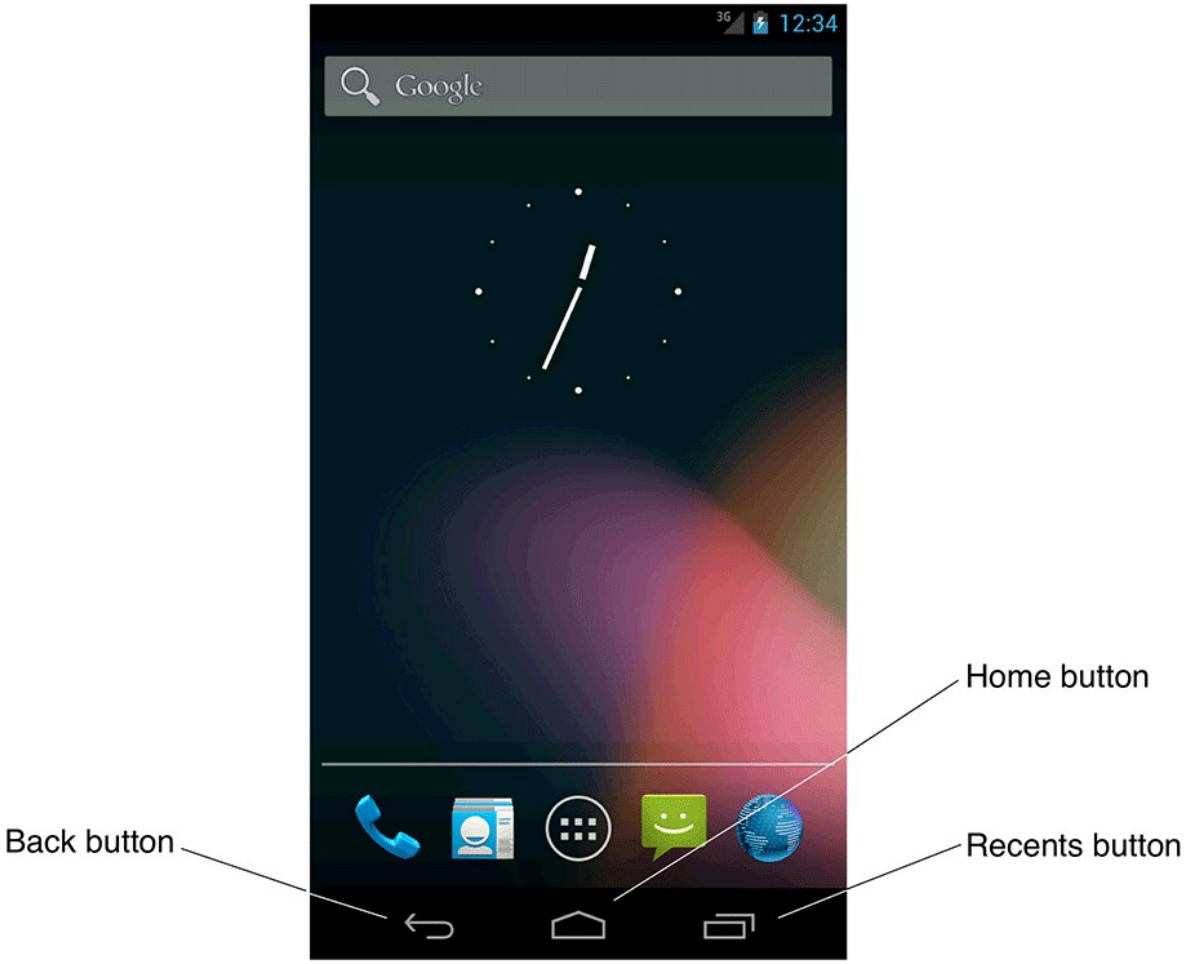
onCreate(…)

Typically, an activity overrides **onCreate(…)** to prepare the specifics of its user interface:

- inflating widgets and putting them on screen (in the call to (**setContentView(int))**
- getting references to inflated widgets
- setting listeners on widgets to handle user interaction
- connecting to external model data

```
@Override
  public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_quiz);

    mTrueButton = (Button)findViewById(R.id.true_button);
    mTrueButton.setOnClickListener(new View.OnClickListener() {
      @Override
      public void onClick(View v) {
        // Does nothing yet, but soon!
      }
    });

    mFalseButton = (Button)findViewById(R.id.false_button);
  }
```

Latest Versions of an Android Device



Logging the Activity Lifecycle

```
@Override
    public void onStart() {
        super.onStart();
        Log.d(TAG, "onStart() called");
    }
    @Override
    public void onPause() {
        super.onPause();
        Log.d(TAG, "onPause() called");
    }
    @Override
    public void onResume() {
        super.onResume();
        Log.d(TAG, "onResume() called");
    }
    @Override
    public void onStop() {
        super.onStop();
        Log.d(TAG, "onStop() called");
    }
    @Override
    public void onDestroy() {
        super.onDestroy();
        Log.d(TAG, "onDestroy() called");
    }
```

Launching the app creates, starts, and resumes an activity

LogCat

Saved Filters

All messages (no filters

QuizActivity

Search for messages. Accepts Java regexes. Prefix with pid:, app:, tag: or text: to limit scope.

verbose

Level	Time	PID	TID	Application	Tag	Text
D	12-30 12:32:45.014	1097	1097	com.bignerdranch…	QuizActivity	onCreate
D	12-30 12:32:45.755	1097	1097	com.bignerdranch…	QuizActivity	onStart
D	12-30 12:32:45.785	1097	1097	com.bignerdranch…	QuizActivity	onResume
D	12-30 12:48:59.245	1097	1097	com.bignerdranch…	QuizActivity	onPause
D	12-30 12:49:01.284	1097	1097	com.bignerdranch…	QuizActivity	onStop
D	12-30 12:49:01.284	1097	1097	com.bignerdranch…	QuizActivity	onDestroy

Pressing the Back button destroys the activity

LogCat

Saved Filters

All messages (no filters

QuizActivity

Search for messages. Accepts Java regexes. Prefix with pid:, app:, tag: or text: to limit scope.

verbose

Level	Time	PID	TID	Application	Tag	Text
D	12-30 12:32:45.014	1097	1097	com.bignerdranch…	QuizActivity	onCreate
D	12-30 12:32:45.755	1097	1097	com.bignerdranch…	QuizActivity	onStart
D	12-30 12:32:45.785	1097	1097	com.bignerdranch…	QuizActivity	onResume
D	12-30 12:48:59.245	1097	1097	com.bignerdranch…	QuizActivity	onPause
D	12-30 12:49:01.284	1097	1097	com.bignerdranch…	QuizActivity	onStop
D	12-30 12:49:01.284	1097	1097	com.bignerdranch…	QuizActivity	onDestroy

Pressing the Home button stops the activity

LogCat

Saved Filters

All messages (no filters

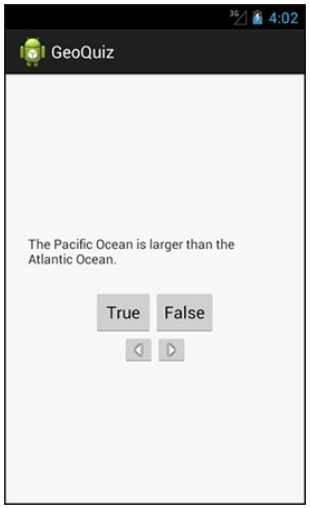
QuizActivity

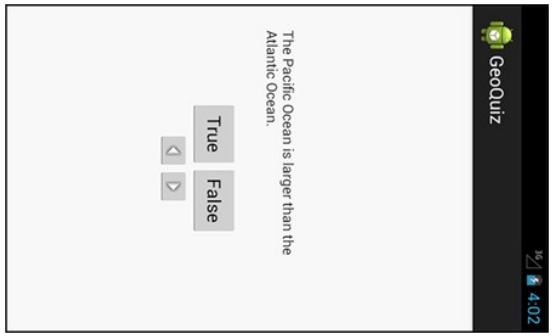
Search for messages. Accepts Java regexes. Prefix with pid:, app:, tag: or text: to limit scope.

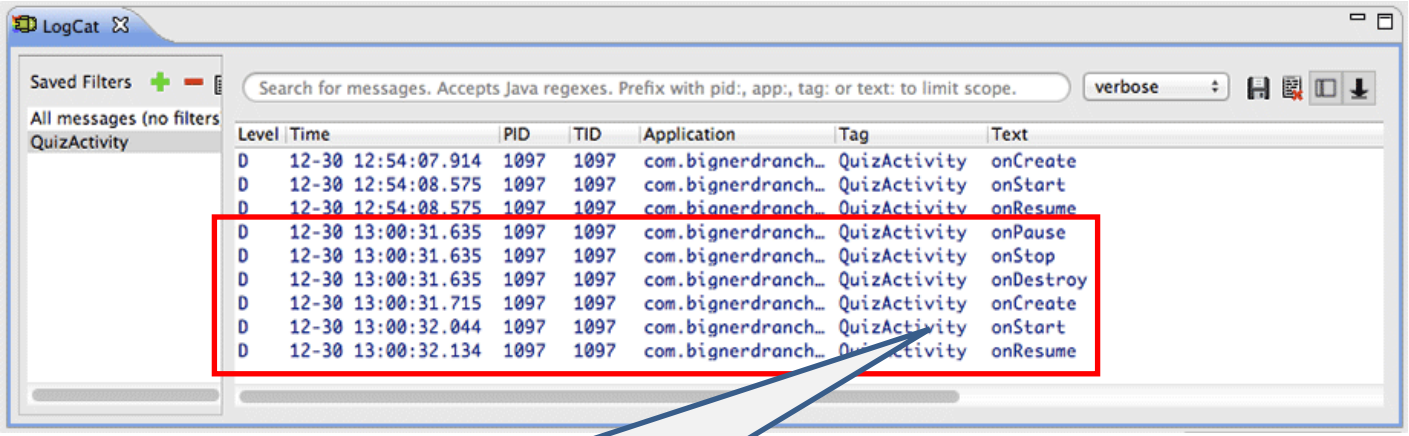
verbose

Level	Time	PID	TID	Application	Tag	Text
D	12-30 12:49:01.284	1097	1097	com.bignerdranch…	QuizActivity	onStop
D	12-30 12:49:01.284	1097	1097	com.bignerdranch…	QuizActivity	onDestroy
D	12-30 12:50:01.087	1097	1097	com.bignerdranch…	QuizActivity	onCreate
D	12-30 12:50:01.715	1097	1097	com.bignerdranch…	QuizActivity	onStart
D	12-30 12:50:01.715	1097	1097	com.bignerdranch…	QuizActivity	onResume
D	12-30 12:50:47.075	1097	1097	com.bignerdranch…	QuizActivity	onPause
D	12-30 12:50:49.945	1097	1097	com.bignerdranch…	QuizActivity	onStop

Rotation and the Activity Lifecycle







Activity is dead and reborn again!

Saving Data Across Rotation

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

// Checking Bundle in onCreate(…)
...

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

      updateQuestion();
   }
```

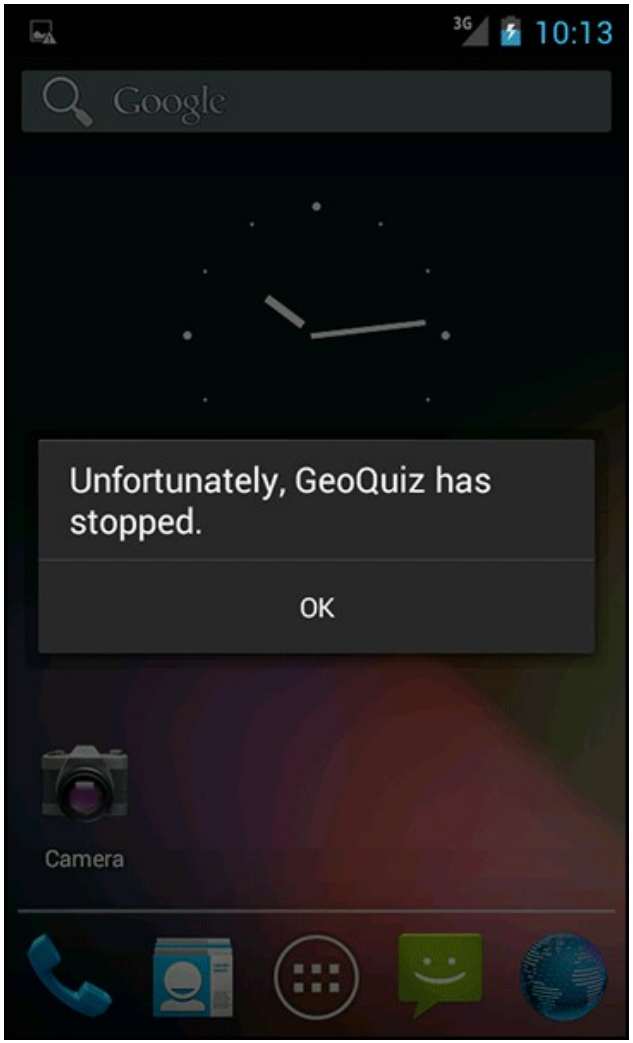
Android Basics
Debugging Android Apps

Javed Hasan

BJIT Limited

Debugging Overview

Find out what
to do when
apps get buggy.



Debugging Tools Overview

LogCat

- Show Exceptions and Stack Traces
- Diagnose misbehaviors with logging stack traces

Debugger

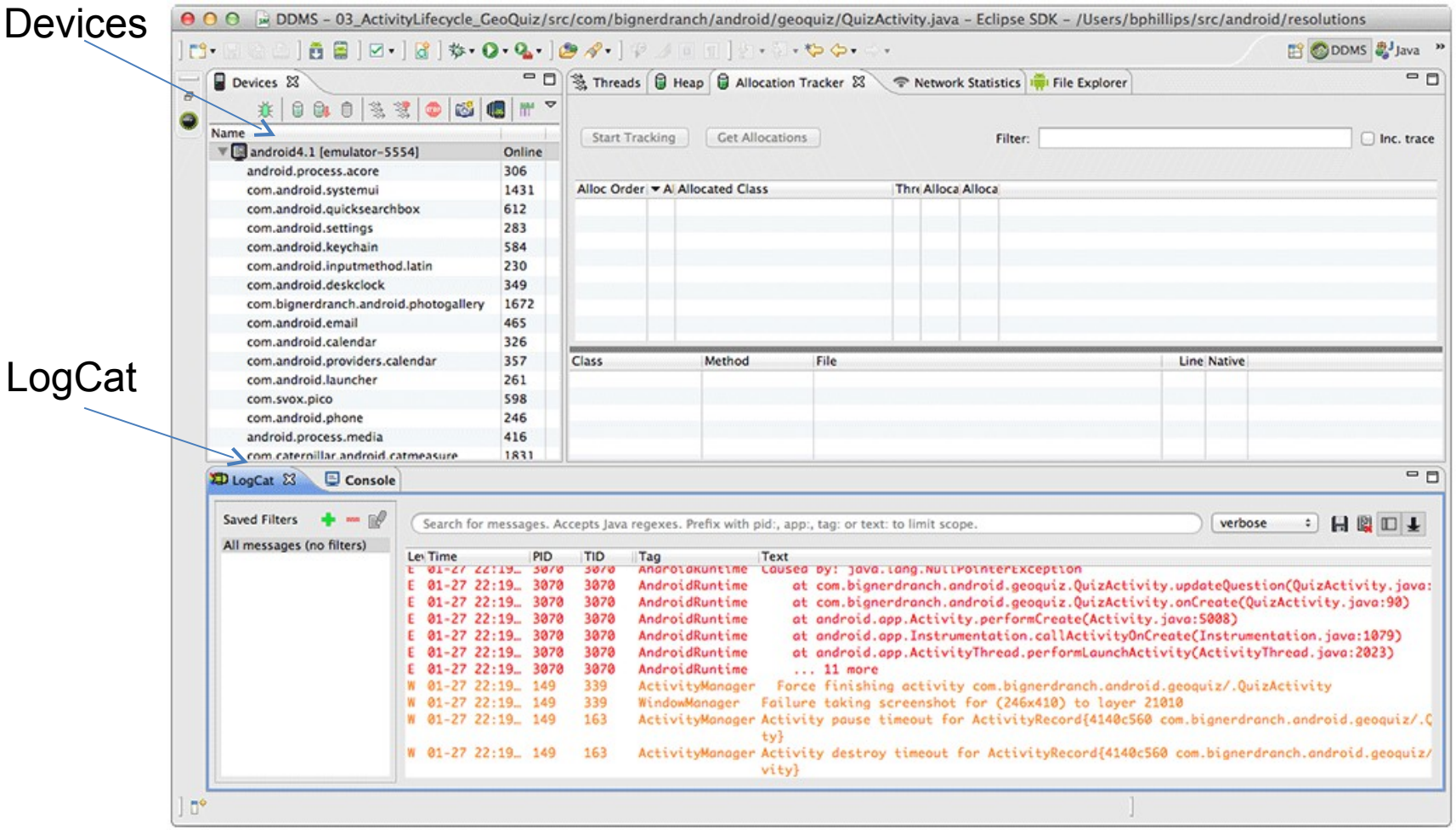
- Diagnose misbehaviors with breakpoint

Android Lint

- Examine codes to find defects without running it

LogCat DDMS Perspective

DDMS does the footwork for all Android debugging. The DDMS perspective includes LogCat and the Devices view.



LogCat Stack Traces

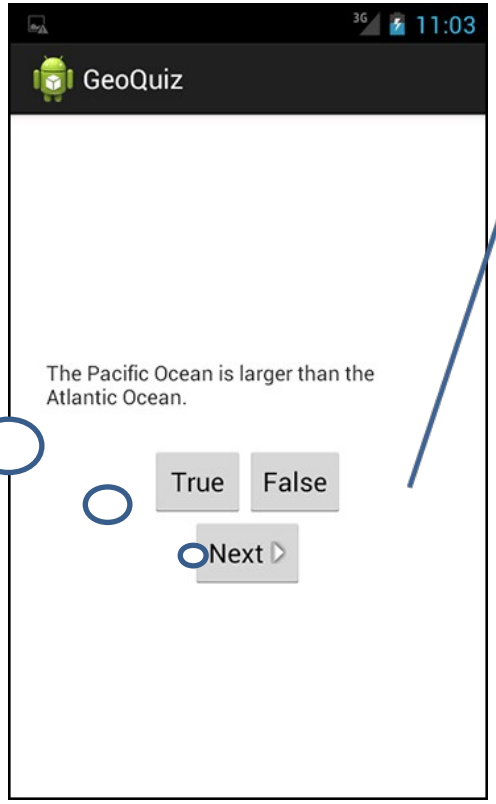
Tag	Text
dalvikvm	Not late-enabling CheckJNI (already on)
ActivityManager	Start proc com.bignerdranch.android.geoquiz
	Activity: pid=3116 uid=10052 gids={1028}
Trace	error opening trace file: No such file or di
QuizActivity	onCreate() called
AndroidRuntime	Shutting down VM
dalvikvm	threadid=1: thread exiting with uncaught exc
AndroidRuntime	FATAL EXCEPTION: main
AndroidRuntime	java.lang.RuntimeException: Unable to start
	quiz/com.bignerdranch.android.geoquiz.QuizAc
AndroidRuntime	at android.app.ActivityThread.performLau
AndroidRuntime	at android.app.ActivityThread.handleLaun
AndroidRuntime	at android.app.ActivityThread.access$600
AndroidRuntime	at android.app.ActivityThread$H.handleMe
AndroidRuntime	at android.os.Handler.dispatchMessage(Ha
AndroidRuntime	at android.os.Looper.loop(Looper.java:13
AndroidRuntime	at android.app.ActivityThread.main(Activ
AndroidRuntime	at java.lang.reflect.Method.invokeNative
AndroidRuntime	at java.lang.reflect.Method.invoke(Metho
AndroidRuntime	at com.android.internal.os.ZygoteInit$Me
AndroidRuntime	at com.android.internal.os.ZygoteInit.ma
AndroidRuntime	at dalvik.system.NativeStart.main(Native
AndroidRuntime	Caused by: java.lang.NullPointerException
AndroidRuntime	at com.bignerdranch.android.geoquiz.QuizActivity.updateQuestion(QuizActivity.ja
AndroidRuntime	at com.bignerdranch.android.geoquiz.QuizActivity.onCreate(QuizActivity.java:90)
AndroidRuntime	at android.app.Activity.performCreate(Activity.java:5008)
AndroidRuntime	at android.app.Instrumentation.callActivityOnCreate(Instrumentation.java:1079)
AndroidRuntime	at android.app.ActivityThread.performLaunchActivity(ActivityThread.java:2023)
AndroidRuntime	... 11 more
ActivityManager	Force finishing activity com.bignerdranch.android.geoquiz/.QuizActivity
WindowManager	Failure taking screenshot for (246x410) to layer 21010
Choreographer	Skipped 31 frames! The application may be doing too much work on its main thread.
ActivityManager	Activity pause timeout for ActivityRecord{413bbdc0 com.bignerdranch.android.geoquiz.

```
//QuizActivity.java
protected void onCreate(Bundle
savedInstanceState) {
    ...
    setContentView(R.layout.activity_quiz);
    mQuestionTextView =
(TextView)findViewById(R.id.question_text_view);
    //mQuestionTextView =
(TextView)findViewById(R.id.question_text_view);
    mTrueButton =
...
}
```

Diagnosing Misbehaviors Stack Trace 1/2

Apps may show sometimes non-crashing misbehavior.

Pressing Next does nothing




```
protected void onCreate(Bundle
savedInstanceState) {
   super.onCreate(savedInstanceState);
   ...
   mNextButton =
(Button)findViewById(R.id.next_button);
   mNextButton.setOnClickListener(new
View.OnClickListener() {
      @Override
      public void onClick(View v) {
         mCurrentIndex = (mCurrentIndex + 1) %
mQuestionBank.length;
         //mCurrentIndex = (mCurrentIndex + 1) %
mQuestionBank.length;
         updateQuestion();
      }
   });
   ...
}
```

Diagnosing Misbehaviors Stack Trace 2/2

Tag	Text
	eNotFoundException: /proc/net/xt_qtaguid/iface_stat_all: o
	irectory)
SizeAdaptiveLa…	com.android.internal.widget.SizeAdaptiveLayout@41ccc060chi
	cd2c30 measured out of bounds at 95px clamped to 96px
QuizActivity	Updating question text for question #0
QuizActivity	java.lang.Exception
QuizActivity	at com.bignerdranch.android.geoquiz.QuizActivity.updat
QuizActivity	at com.bignerdranch.android.geoquiz.QuizActivity.acces
QuizActivity	at com.bignerdranch.android.geoquiz.QuizActivity$3.onC
QuizActivity	at android.view.View.performClick(View.java:4084)
	lick.run(View.java:16966)
	allback(Handler.java:615)
	hMessage(Handler.java:92)
	per.java:137)
	.main(ActivityThread.java
	nvokeNative(Native Method
	nvoke(Method.java:511)
	goteInit$MethodAndArgsCal
	goteInit.main(ZygoteInit.
	main(Native Method)

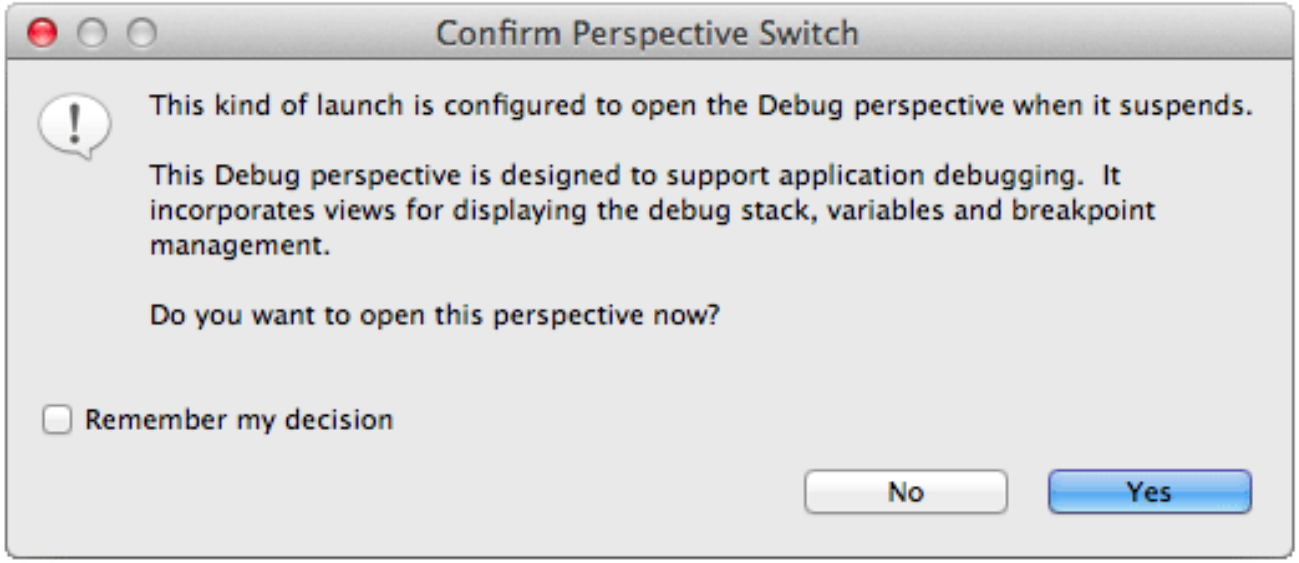
```
public class QuizActivity extends Activity {
    ...
    public void updateQuestion() {
        Log.d(TAG, "Updating question text for question #" + mCurrentIndex,
          new Exception());
      int question = mQuestionBank[mCurrentIndex].getQuestion();
      mQuestionTextView.setText(question);
    }
```

Diagnosing Misbehaviors Set Breakpoint 1/2

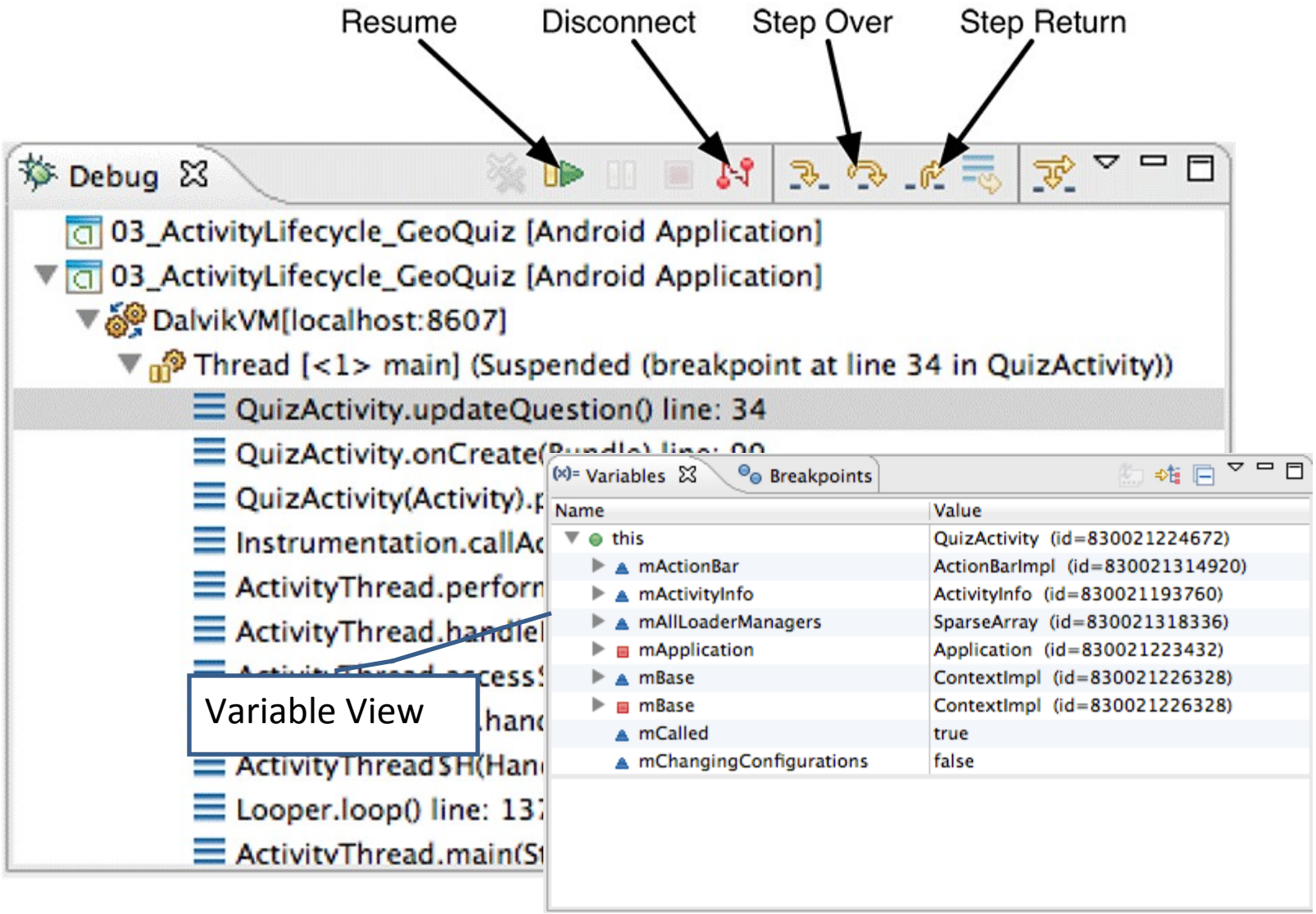
Set breakpoint by double clicking grey area at left margin

```
private void updateQuestion() {
    int question = mAnswerKey[mCurrentIndex].getQuestion();
    mQuestionTextView.setText(question);
}
```

Switch to Debug Perspective
Right-click GeoQuiz project and select Debug As → Android Application

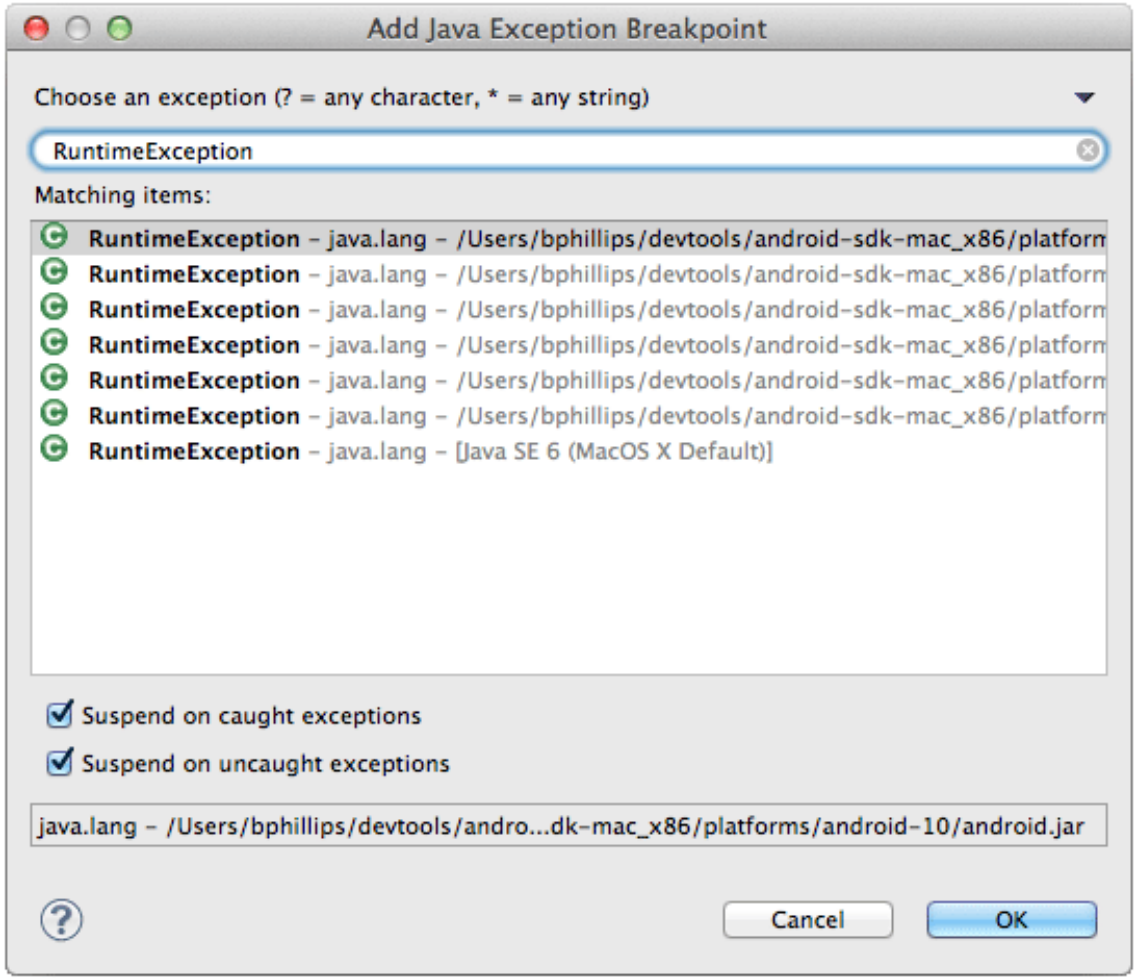


Diagnosing Misbehaviors Set Breakpoint 2/2



Diagnosing Misbehaviors Set Exception Breakpoint

Use, Run → Add Java Exception Breakpoint... to catch exception breakpoints



Android Specific Debug – Android Lint

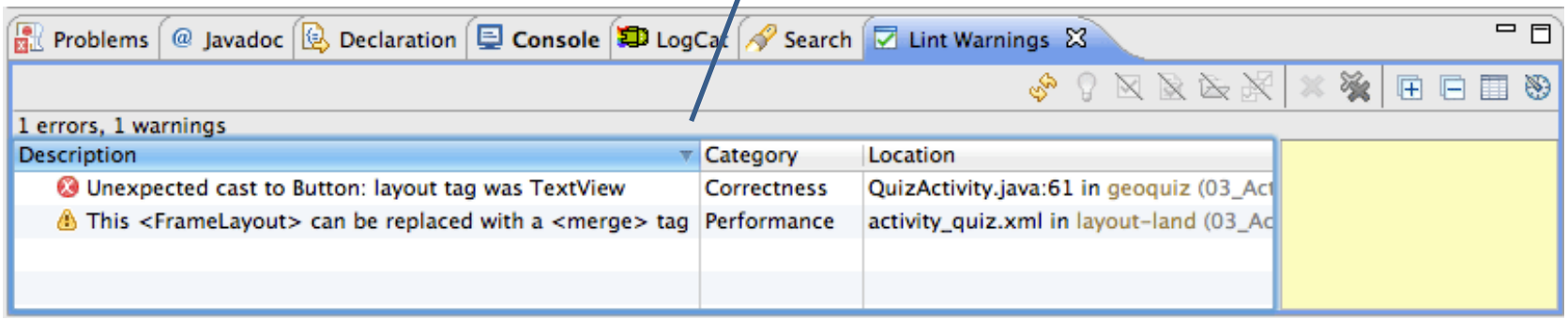
Android Lint is a static analyzer for Android code.

A static analyzer is a program that examines your code to find defects without running it.

Android Lint uses its knowledge of the Android frameworks to look deeper into your code and find problems that the compiler cannot.

In the package explorer, right-click the GeoQuiz project and select Android Tools → Run Lint: Check for Common Errors to see the Lint Warnings view.

```
@Override
protected void onCreate(Bundle
savedInstanceState) {
    super.onCreate(savedInstanceState);
    Log.d(TAG, "onCreate() called");
    setContentView(R.layout.activity_quiz);
    mQuestionTextView =
(TextView)findViewById(R.id.question_text_vie
w);
    mTrueButton =
(Button)findViewById(R.id.question_text_view);
    mTrueButton =
(Button)findViewById(R.id.true_button);
    ...
}
```



Android Basics
Multi-Activity App

Javed Hasan

BJIT Limited

Multi-Activity GeoQuiz App Overview

Multi-Activity GeoQuiz App Overview

- Main Screens

Setup new Cheat Activity

- Define Layout and Activity subclass
- Define it in manifest
- Add Cheat! Button in Quiz Activity to start Cheat Activity

Starting Cheat Activity

- Using Intent

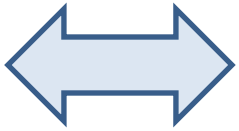
Passing data between activity

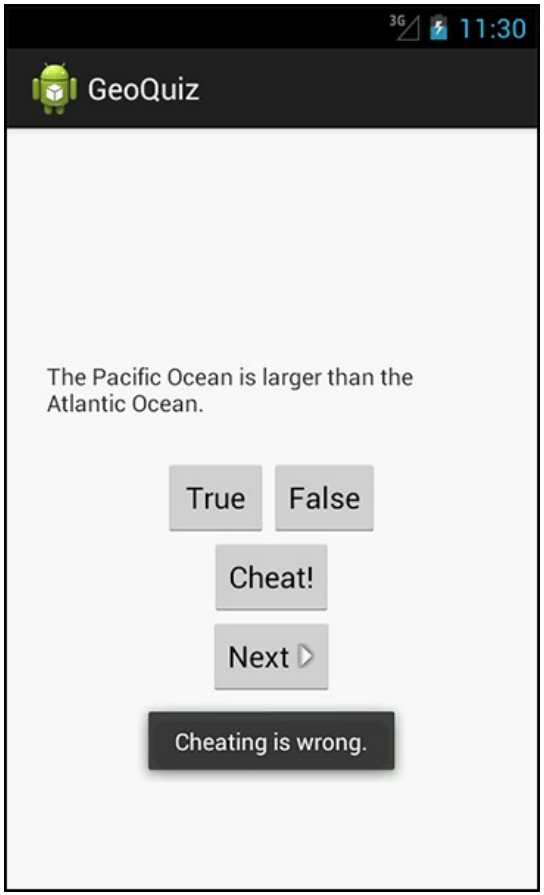
- Using Intent extra
- Get a result from Child Activity

Main Screens

CheatActivity offers the chance to peek at the answer

QuizActivity knows if you've been cheating





Setup New Cheat Activity

Add Strings

Create New Layout

- Creating a new layout file
- Naming and configuring new layout file

Create a New Cheat Activity Subclass

Declaring Activity in Manifest

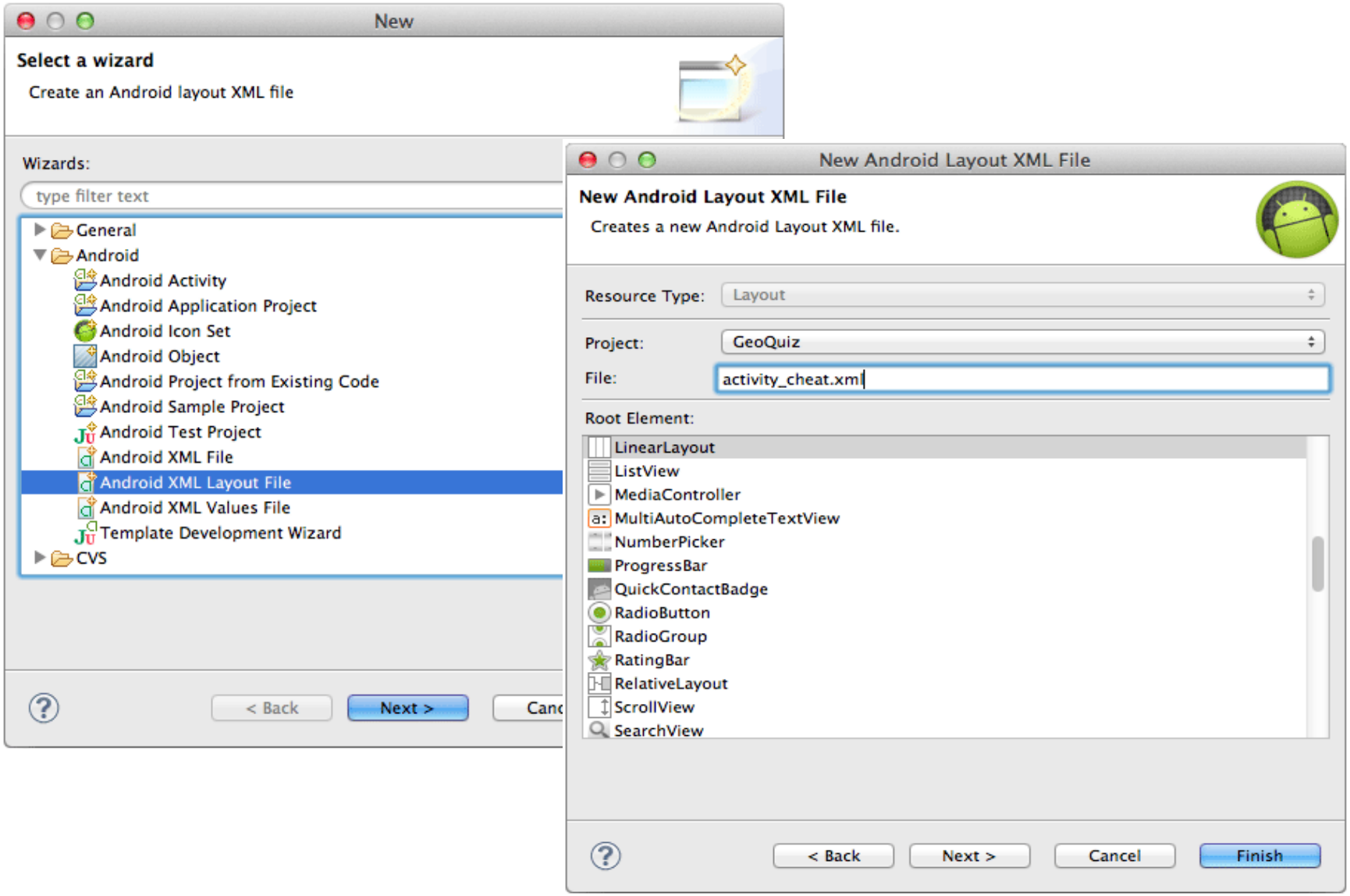
Add a Cheat Button in Quiz Activity

- Wiring up Cheat Button

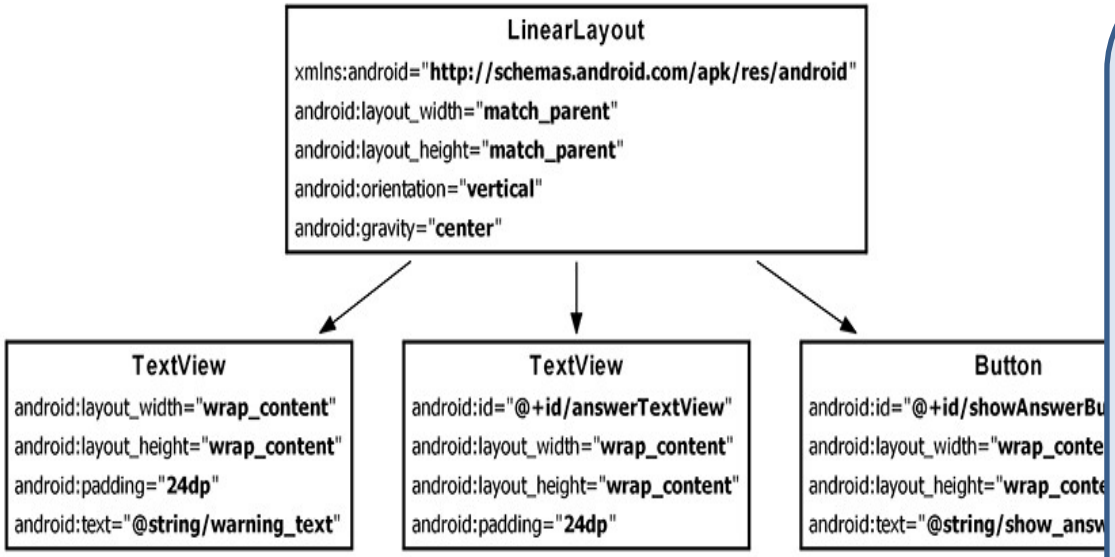
Add Strings (strings.xml)

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

</resources>
```

Create New Layout File

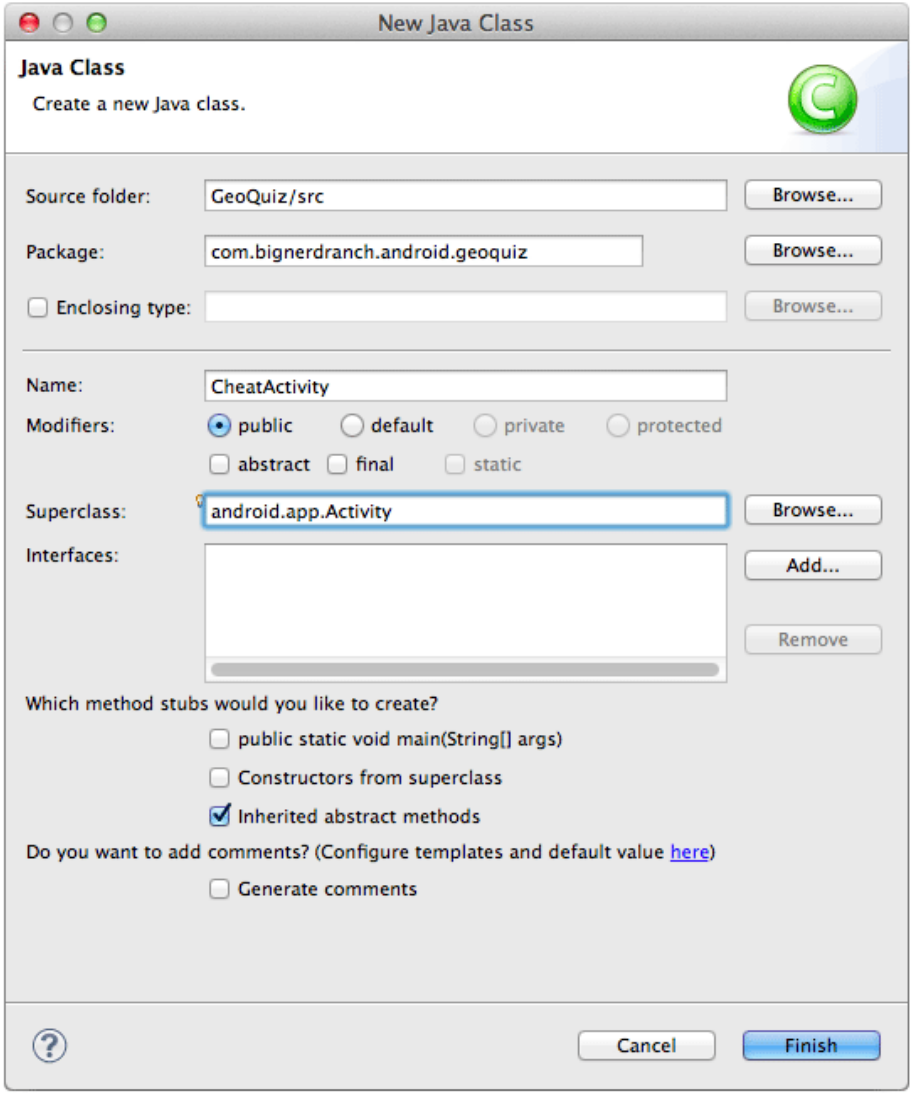


Design and Implement New Layout




```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
xmlns:android="http://schemas.android.com/apk/res
/android" android:layout_width="match_parent"
android:layout_height="match_parent"
android:orientation="vertical"
android:gravity="center">
<TextView android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:padding="24dp"
android:text="@string/warning_text" />
<TextView android:id="@+id/answerTextView"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:padding="24dp" />
 <Button android:id="@+id/showAnswerButton"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:text="@string/show_answer_button" />
</LinearLayout>
```

Create New CheatActivity Subclass



Override onCreate(…) CheatActivity.java

```
public class CheatActivity extends Activity {

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

}
```

Declaring CheatActivity in the Mainfest

```
<?xml version="1.0" encoding="utf-8"?>
<manifest
xmlns:android="http://schemas.android.com/apk/res/android"
  package="com.bignerdranch.android.geoquiz"
  android:versionCode="1"
  android:versionName="1.0" >
  <uses-sdk
    android:minSdkVersion="8"
    android:targetSdkVersion="17" />
  <application
    android:allowBackup="true"
    android:icon="@drawable/ic_launcher"
    android:label="@string/app_name"
    android:theme="@style/AppTheme" >
 <activity
android:name="com.bignerdranch.android.geoquiz.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/app_name" />
  </application>
</manifest>
```

Add a Cheat! Button to QuizActivity

Adding a Cheat! button to the default layout (layout/activity_quiz.xml)

```
...
</LinearLayout>
<Button
  android:id="@+id/cheat_button"
  android:layout_width="wrap_content"
  android:layout_height="wrap_content"
  android:text="@string/cheat_button" />

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

</LinearLayout>
```

Adding a Cheat! button to the landscape layout (layout/activity_quiz.xml)

```
...
</LinearLayout>
<Button
  android:id="@+id/cheat_button"
  android:layout_width="wrap_content"
  android:layout_height="wrap_content"
  android:layout_gravity="bottom|center"
  android:text="@string/cheat_button" />

<Button
  android:id="@+id/next_button"
  android:layout_width="wrap_content"
  android:layout_height="wrap_content"
  android:layout_gravity="bottom|right"
  android:text="@string/next_button"
  android:drawableRight="@drawable/arrow_right"
  android:drawablePadding="4dp" />

</FrameLayout>
```

Wireup Cheat! Button (QuizActivity.java)

```
public class QuizActivity extends Activity {
    ...
    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) {
                // Start CheatActivity
            }
        });
      updateQuestion();
    }
 ...
}
```

Starting an Activity with Intent

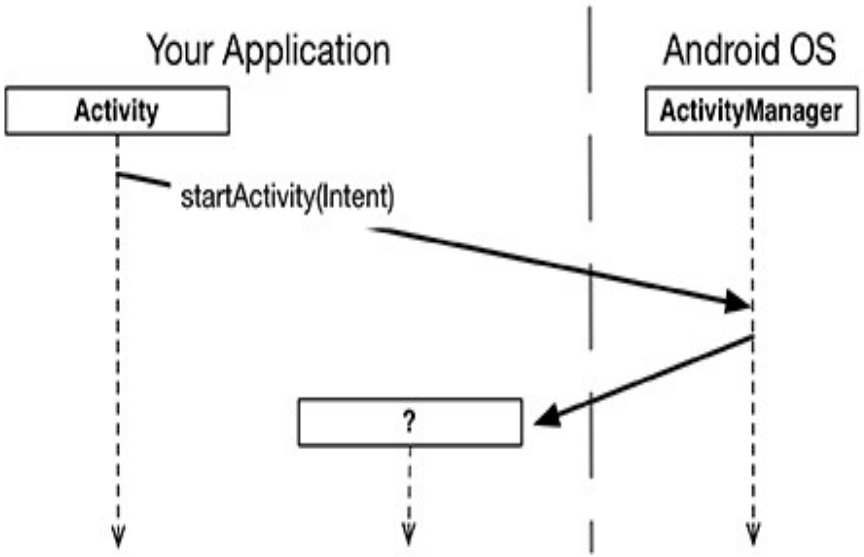
An intent is an object that a component can use to communicate with the OS

Here, you are using an intent to tell the ActivityManager which activity to start

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

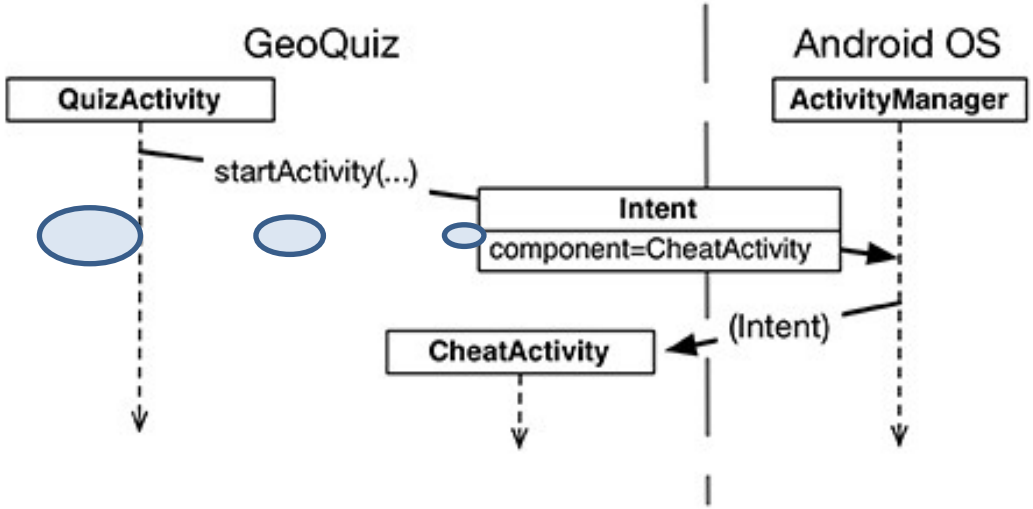
The Class object specifies the activity that the ActivityManager should start

The Context object tells the ActivityManager which package the Class object can be found in.



Communicating with Intent

The Intent: telling ActivityManager what to do

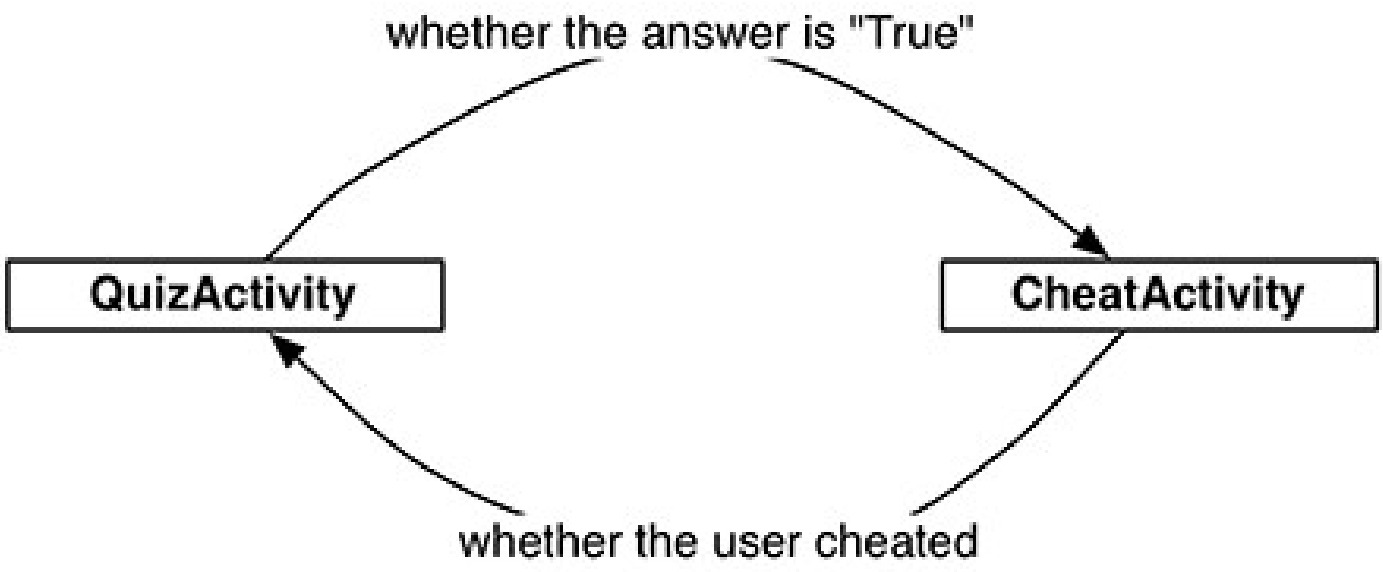


Starting CheatActivity (QuizActivity.java)

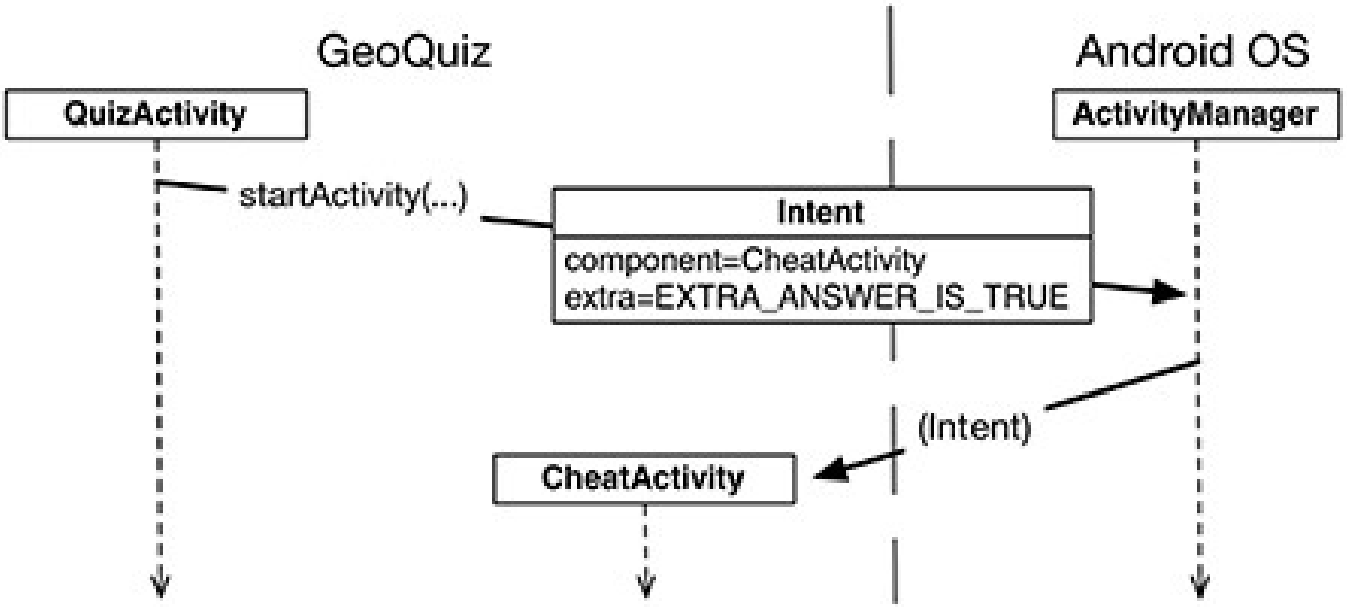
```
...
    mCheatButton = (Button)findViewById(R.id.cheat_button);
    mCheatButton.setOnClickListener(new View.OnClickListener() {

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

Passing Data Between Activity

The conversation between QuizActivity and CheatActivity



Using Intent Extras 1/2



Using Intent Extras 2/2

1. Add extra constant (CheatActivity.java)

```
public class CheatActivity extends Activity {

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

2. Putting an extra on the intent (QuizActivity.java)

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

        updateQuestion();
    }
```

3. Using an extra (CheatActivity.java)

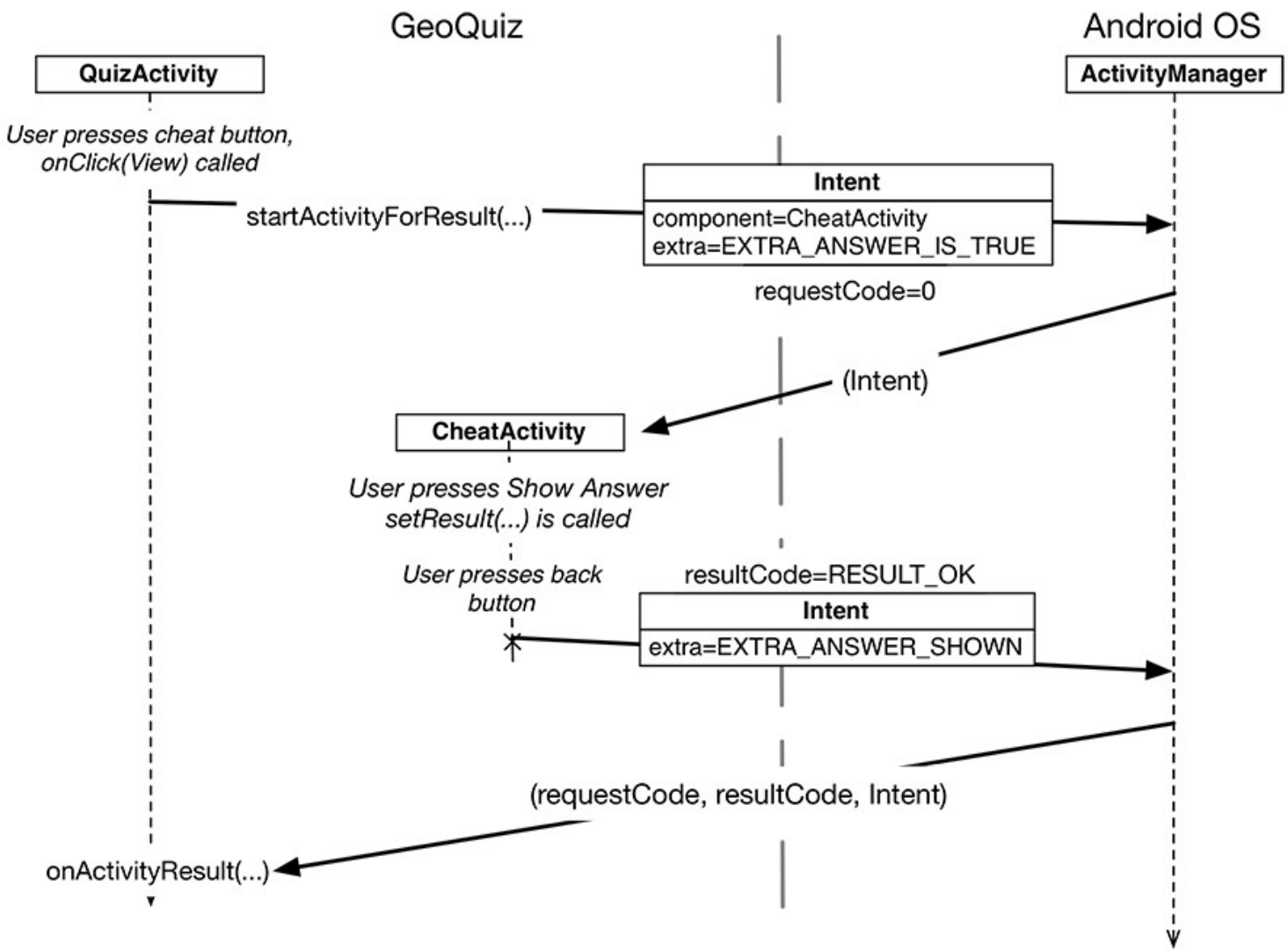
```
public class CheatActivity extends Activity {
    ...
    private boolean mAnswerIsTrue;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_cheat);
        mAnswerIsTrue =
                getIntent().getBooleanExtra(EXTRA_ANSWER_IS_TRUE, false);
    }
}
```

Enable Cheating (CheatActivity.java)

4. In CheatActivity, wire up the answer TextView and the Show Answer button to use the retrieved value.

```
public class CheatActivity extends Activity {
    ...
    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.answerTextView);

        mShowAnswer = (Button)findViewById(R.id.showAnswerButton);
        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);
                }
            }
        });
    }
}
```

Getting a Result Back from a Child Activity



Setting the Result 1/2

1. Calling startActivityForResult(…) (QuizActivity.java)

```
...
    mCheatButton.setOnClickListener(new View.OnClickListener() {
      @Override
      public void onClick(View v) {
        Intent i = new Intent(QuizActivity.this, CheatActivity.class);
        boolean answerIsTrue = mQuestionBank[mCurrentIndex].isTrueQuestion();
        i.putExtra(CheatActivity.EXTRA_ANSWER_IS_TRUE, answerIsTrue);
        startActivityForResult(i, 0);
      }
    });
    updateQuestion();
  }
```

Setting the Result 2/2

2. Setting a result (CheatActivity.java)

```
public class CheatActivity extends Activity {

  public static final String EXTRA_ANSWER_IS_TRUE =
    "com.bignerdranch.android.geoquiz.answer_is_true";
  public static final String EXTRA_ANSWER_SHOWN =
    "com.bignerdranch.android.geoquiz.answer_shown";

  ...

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

```
@Override
  protected void onCreate(Bundle savedInstanceState) {
    ...

    // Answer will not be shown until the user
    // presses the button
    setAnswerShownResult(false);
    ...
    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);
        }
        setAnswerShownResult(true);
      }
    });
  }
}
```

Handle the Result 1/2

1. Implementing onActivityResult(…) (QuizActivity.java)

```
public class QuizActivity extends Activity {
    ...
    private int mCurrentIndex = 0;
    private boolean mIsCheater;
    ...
    @Override
     protected void onActivityResult(int requestCode, int resultCode, Intent data) {
         if (data == null) {
           return;
         }
         mIsCheater = data.getBooleanExtra(CheatActivity.EXTRA_ANSWER_SHOWN, false);
     }
    ...
}
```

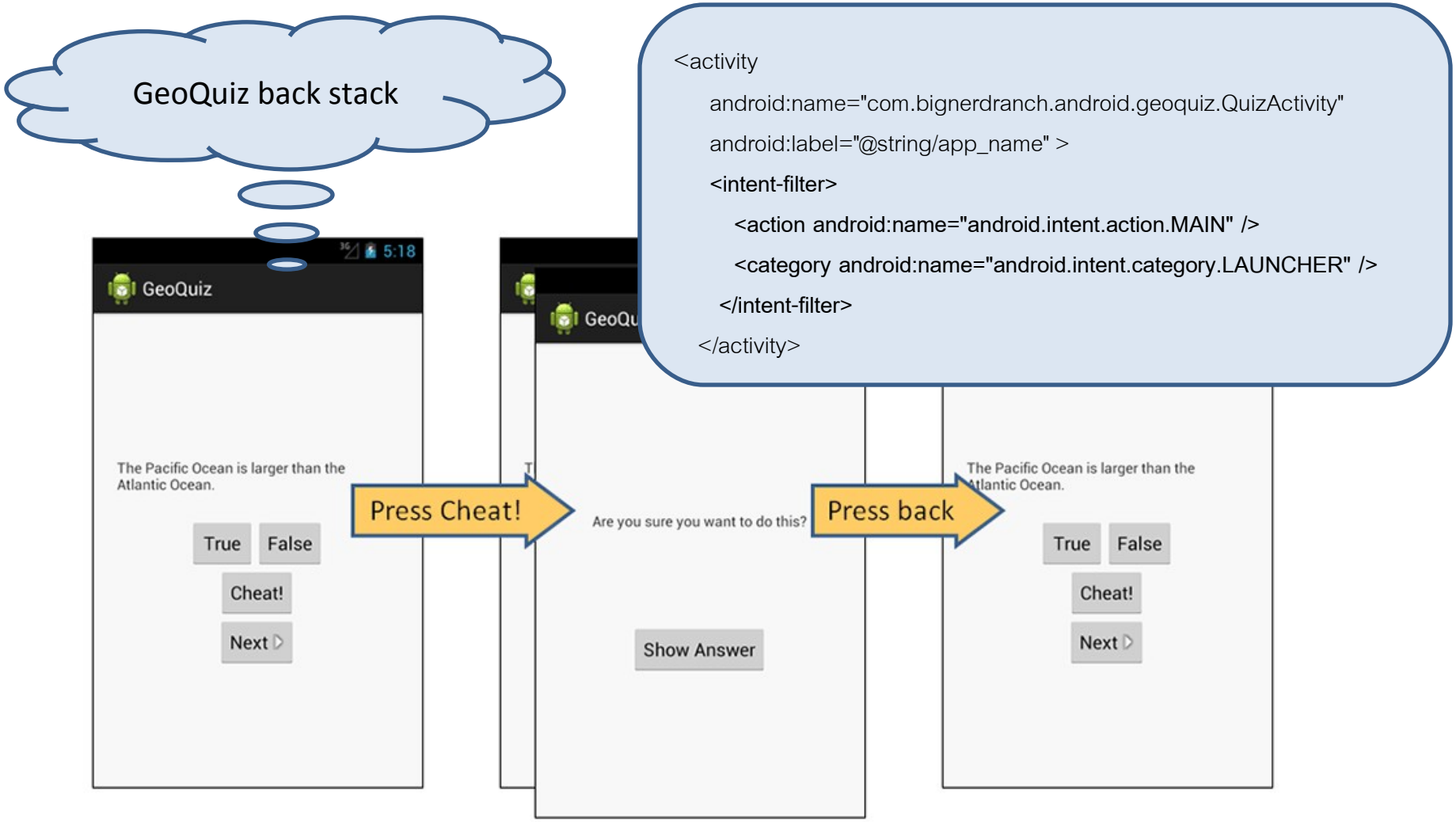
Handle the Result 2/2

2. Use the result in Application Logic (QuizActivity.java)

```
private void checkAnswer(boolean userPressedTrue) {
    boolean answerIsTrue =
        mQuestionBank[mCurrentIndex].isTrueQuestion();

    int messageResId = 0;

    if (mIsCheater) {
        messageResId = R.string.judgment_toast;
    } else {
        if (userPressedTrue == answerIsTrue) {
            messageResId = R.string.correct_toast;
        } else {
            messageResId = R.string.incorrect_toast;
        }
    }

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

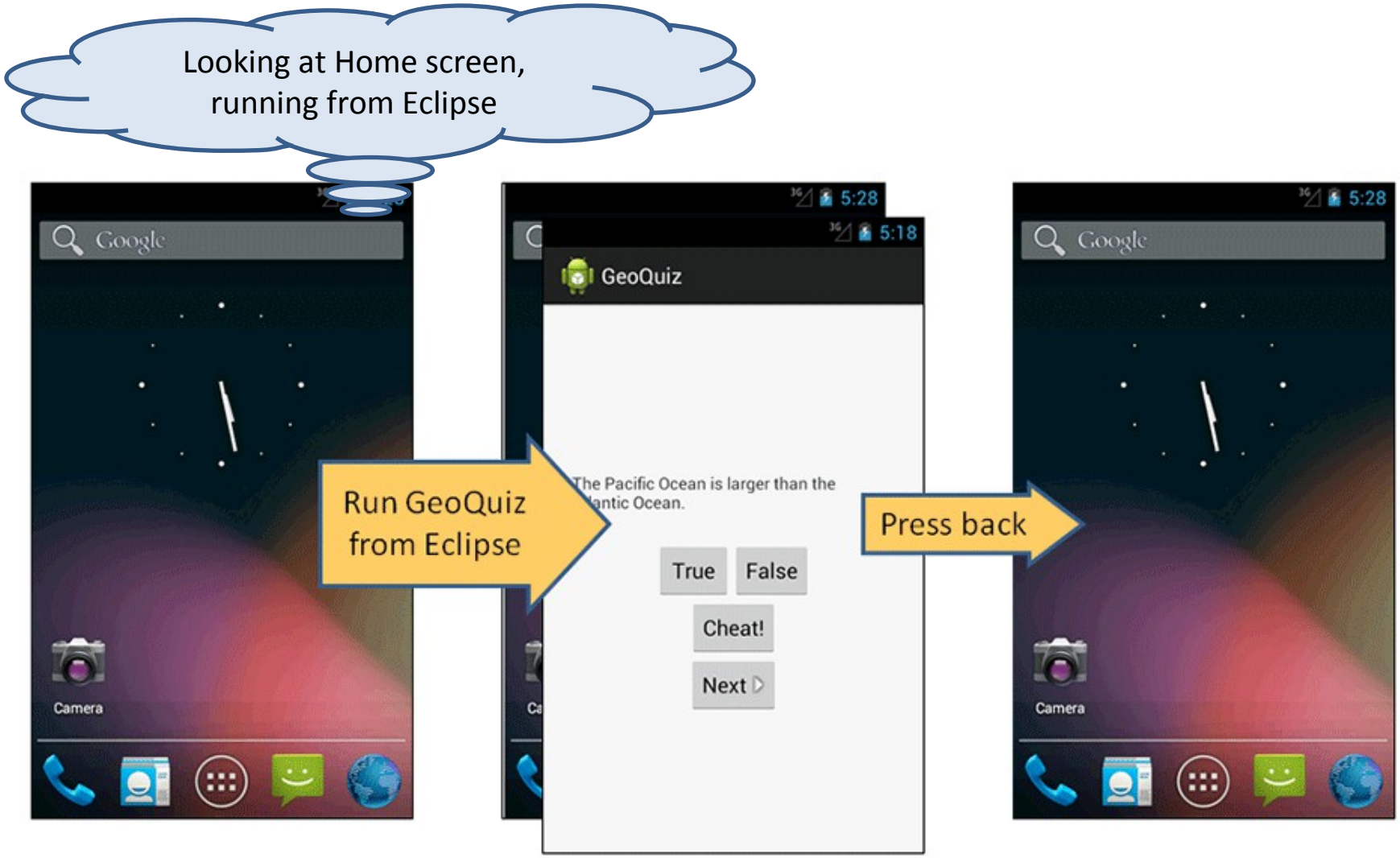
```
@Override
  protected void onCreate(Bundle savedInstanceState) {
    ...
    mNextButton = (Button)findViewById(R.id.next_button);
    mNextButton.setOnClickListener(new View.OnClickListener() {
      @Override
      public void onClick(View v) {
        mCurrentIndex =
            (mCurrentIndex + 1) % mQuestionBank.length;
        mIsCheater = false;
        updateQuestion();
      }
    });
    ...
  }
```

How Android Sees Your Activities 1/3




```
<activity
    android:name="com.bignerdranch.android.geoquiz.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>
```

How Android Sees Your Activities 2/3



How Android Sees Your Activities 3/3

Challenges

GeoQuiz has a few major loopholes. For this challenge, you will busy yourself with closing them.

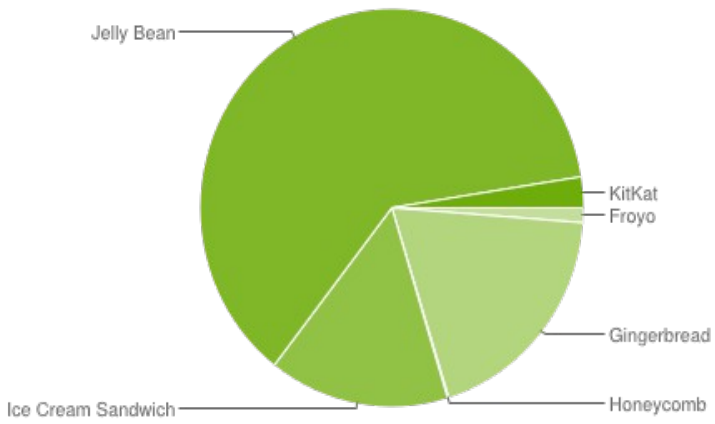
1. Users can rotate **CheatActivity** after they cheat to clear out the cheating result.
2. Once they get back, users can rotate QuizActivity to clear out **mIsCheater**.
3. Users can press **Next** until the question they cheated on comes back around.

Android Basics
Android SDK Versions and Compatibility

Javed Hasan

BJIT Limited

Android API Levels, Firmware Versions, and % of Devices in Use



Version	Codename	API	Distribution
2.2	Froyo	8	1.2%
2.3.3 - 2.3.7	Gingerbread	10	19.0%
3.2	Honeycomb	13	0.1%
4.0.3 - 4.0.4	Ice Cream Sandwich	15	15.2%
4.1.x	Jelly Bean	16	35.3%
4.2.x		17	17.1%
4.3		18	9.6%
4.4	KitKat	19	2.5%

Data collected during a 7-day period ending on March 3, 2014.
Any versions with less than 0.1% distribution are not shown.
URL: http://developer.android.com/about/dashboards/index.html

Compatibility and Android Programming

The delay in upgrades combined with regular new releases makes compatibility an important issue in Android programming.

To reach a broad market, Android developers must create apps that perform well on devices running Froyo, Gingerbread, Honeycomb, Ice Cream Sandwich, and Jelly Bean versions of Android, as well as on different device form factors.

Targeting different sizes of devices is easier than you might think. Phone screens are a variety of sizes, but the Android layout system does a good job at adapting.

The release of **Honeycomb** was a major shift in Android and introduced a new UI and new architectural components.

Thus, Android developers must spend time ensuring backwards compatibility and bridging the gap between Gingerbread (API level 10) and Honeycomb (API level 11) and beyond. Android has provided help for maintaining backwards compatibility.

SDK Versions (API Level)

Minimum SDK Version

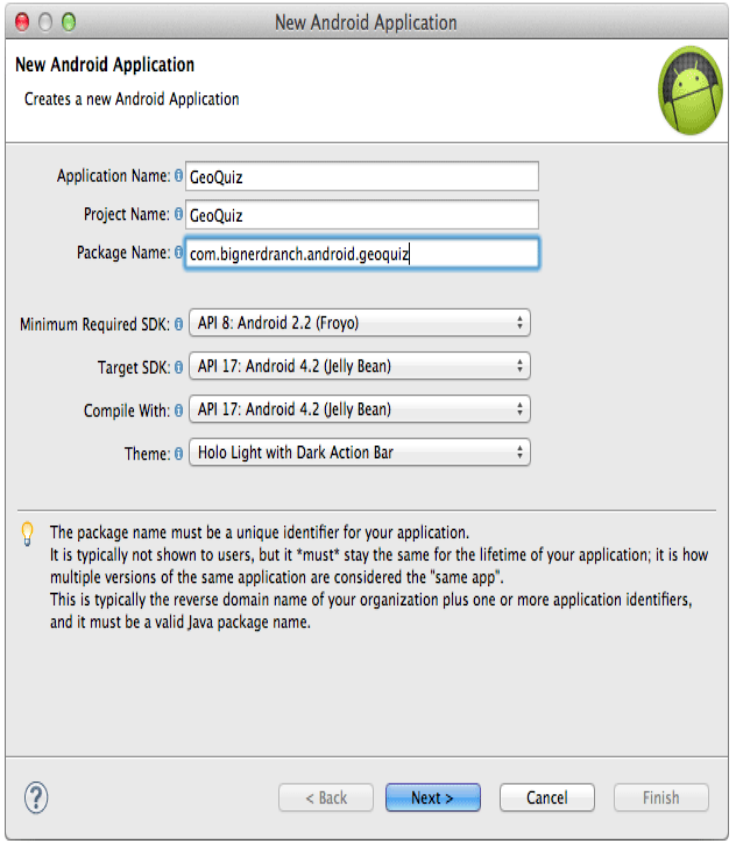
- Below this version OS will refuse to install app

Target SDK Version

- API level your app designed to run on (most often latest Android release)

Build SDK Version

- Specify which version to use building your own code.



Adding Code from Later API Safely

Android Lint can detect potential problems caused by calling newer code on older devices. If you use code from a higher version than your minimum SDK, Android Lint will report build errors.
To use later API code, first wrap later API code in a conditional statement that checks devices build version of Android. Then, suppress lint errors using Annotation.

Step 1: Check the Device's Build Version First

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

     if (Build.VERSION.SDK_INT >=
          Build.VERSION_CODES.HONEYCOMB)
    {
         ActionBar actionBar = getActionBar();
         actionBar.setSubtitle("Bodies of Water");
     }
}
```

Step 2: Suppress Lint Errors

```
@TargetApi(11)
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    Log.d(TAG, "onCreate() called");
    setContentView(R.layout.activity_quiz);

    if (Build.VERSION.SDK_INT >=
         Build.VERSION_CODES.HONEYCOMB) {
        ActionBar actionBar = getActionBar();
        actionBar.setSubtitle("Bodies of Water");
    }
}
```

Using The Android Developer Documentation

Activity | Android Developers

http://developer.android.com/reference/android/app/Activity.html

Developers | Design | Develop | Distribute

Android Training | API Guides | Reference | Tools

Android APIs API level: 16

android
android.accessibilityservice
android.accounts
android.animation
android.app
android.app.admin
android.app.backup
android.appwidget
android.bluetooth
android.content
android.content.pm
android.content.res
android.database

ActionBar
ActionBar.LayoutParams
ActionBar.Tab
Activity
ActivityGroup
ActivityManager
ActivityManager.MemoryInfo
ActivityManager.ProcessErro
ActivityManager.RecentTask
ActivityManager.RunningApp
ActivityManager.RunningSer

Use Tree Navigation

public class

Activity

Summary: Constants | Inherited Constants | Fields | Ctors | Methods | Protected Methods | Inherited Methods | [Expand All]

Since: API Level 1

extends ContextThemeWrapper

implements ComponentCallbacks2 KeyEvent.Callback LayoutInflater.Factory2 View.OnCreateContextMenuListener Window.Callback

java.lang.Object
↳ android.content.Context
↳ android.content.ContextWrapper
↳ android.view.ContextThemeWrapper
↳ android.app.Activity

▸ Known Direct Subclasses

AccountAuthenticatorActivity, ActivityGroup, AliasActivity, BasicDream, ExpandableListActivity, FragmentActivity, ListActivity, NativeActivity

▸ Known Indirect Subclasses

LauncherActivity, PreferenceActivity, TabActivity

Class Overview

An activity is a single, focused thing that the user can do. Almost all activities interact with the user, so the Activity class takes care of creating a window for you in which you can place your UI with `setContentView(View)`. While activities are often presented to the user as full-screen windows, they can also be used in other ways: as floating windows (via a theme with `windowIsFloating` set) or embedded inside of another activity (using `ActivityGroup`). There are two methods almost all subclasses of Activity will implement:

- `onCreate(Bundle)` is where you initialize your activity. Most importantly, here you will usually call

Using The Android Developer Documentation

Activity | Android Developers

http://developer.android.com/reference/android/app/Activity.html#pubmethoc

Reader | Google

Apple | Yahoo! | Google Maps | YouTube | Wikipedia | News (352) | Popular

Android APIs API level: 8

- android
- android.accessibilityservice
- android.accounts
- android.animation
- android.app
- android.app.admin
- android.app.backup
- android.appwidget
- android.bluetooth
- android.content
- android.content.pm
- android.content.res
- android.database
- ...
- ActionBar
- ActionBar.LayoutParams
- ActionBar.Tab
- Activity
- ActivityGroup
- ActivityManager
- ActivityManager.MemoryInfo
- ActivityManager.ProcessErro
- ActivityManager.RecentTask
- ActivityManager.RunningApp
- ActivityManager.RunningSer
- ActivityManager.RunningTas
- ActivityOptions
- AlarmManager
- AlertDialog
- AlertDialog.Builder
- AliasActivity
- Application

Use Tree Navigation

	Print the Activity's state into the given stream.
View	findViewById (int id) Finds a view that was identified by the id attribute from the XML that was processed in onCrea
void	finish () Call this when your activity is done and should be closed.
void	finishActivity (int requestCode) Force finish another activity that you had previously started with startActivityForResult
void	finishActivityFromChild (Activity child, int requestCode) This is called when a child activity of this one calls its finishActivity().
void	finishAffinity () Finish this activity as well as all activities immediately below it in the current task that have th
void	finishFromChild (Activity child) This is called when a child activity of this one calls its finish () method.
ActionBar	getActionBar () Retrieve a reference to this activity's ActionBar.
final Application	getApplication () Return the application that owns this activity.
ComponentName	getCallingActivity () Return the name of the activity that invoked this activity.
String	getCallingPackage () Return the name of the package that invoked this activity.
int	getChangingConfigurations () If this activity is being destroyed because it can not handle a configuration parameter being ch onConfigurationChanged (Configuration) method is *not* being called), then you can use t while in the process of being destroyed.
ComponentName	getComponentName () Returns complete component name of this activity.
View	getCurrentFocus ()

Challenge: Reporting the Build Version