Your First Real(ish) App

CS 347 Mobile Application Development

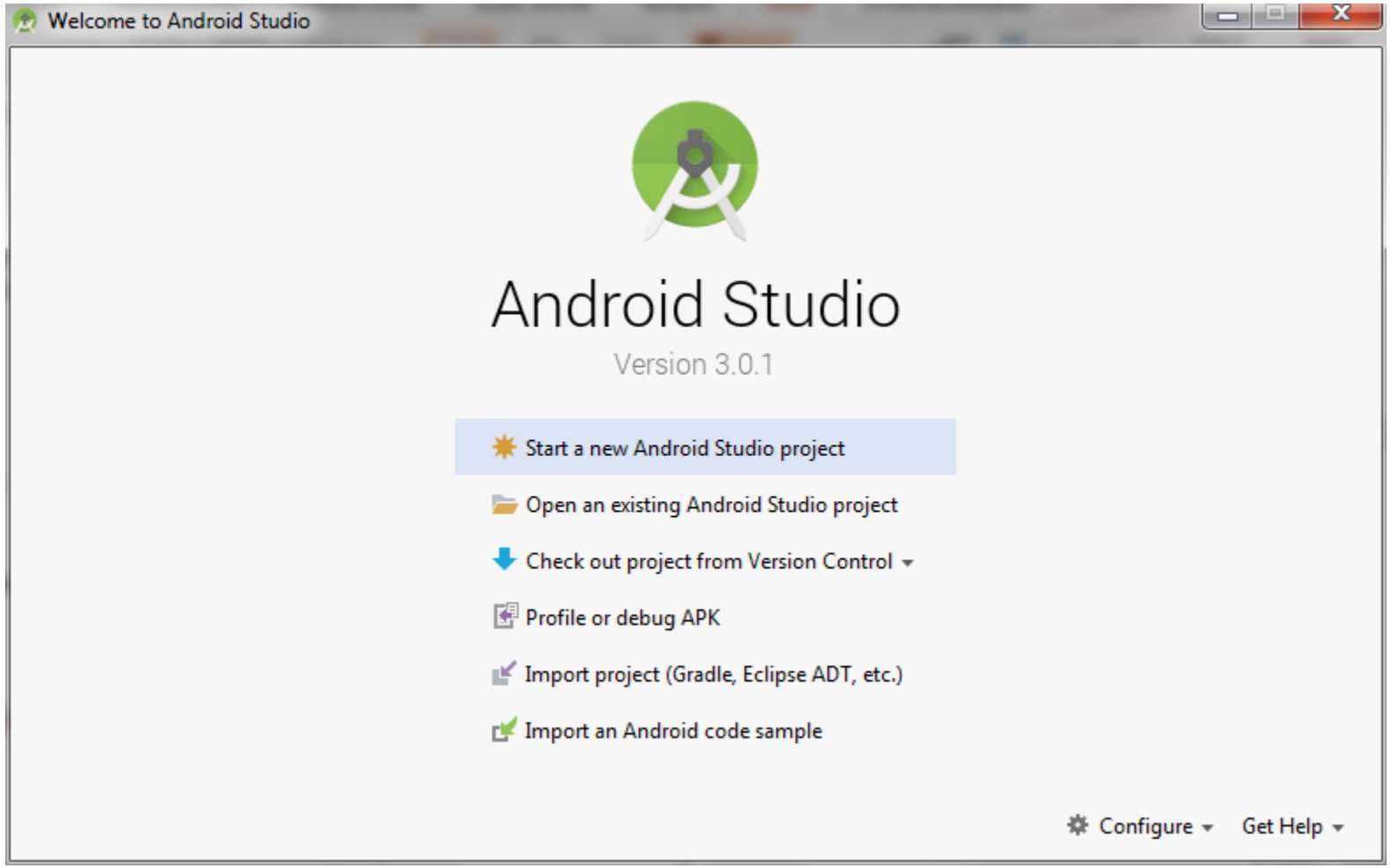
Philip F. Garofalo, M.S., Instructor

NEIU

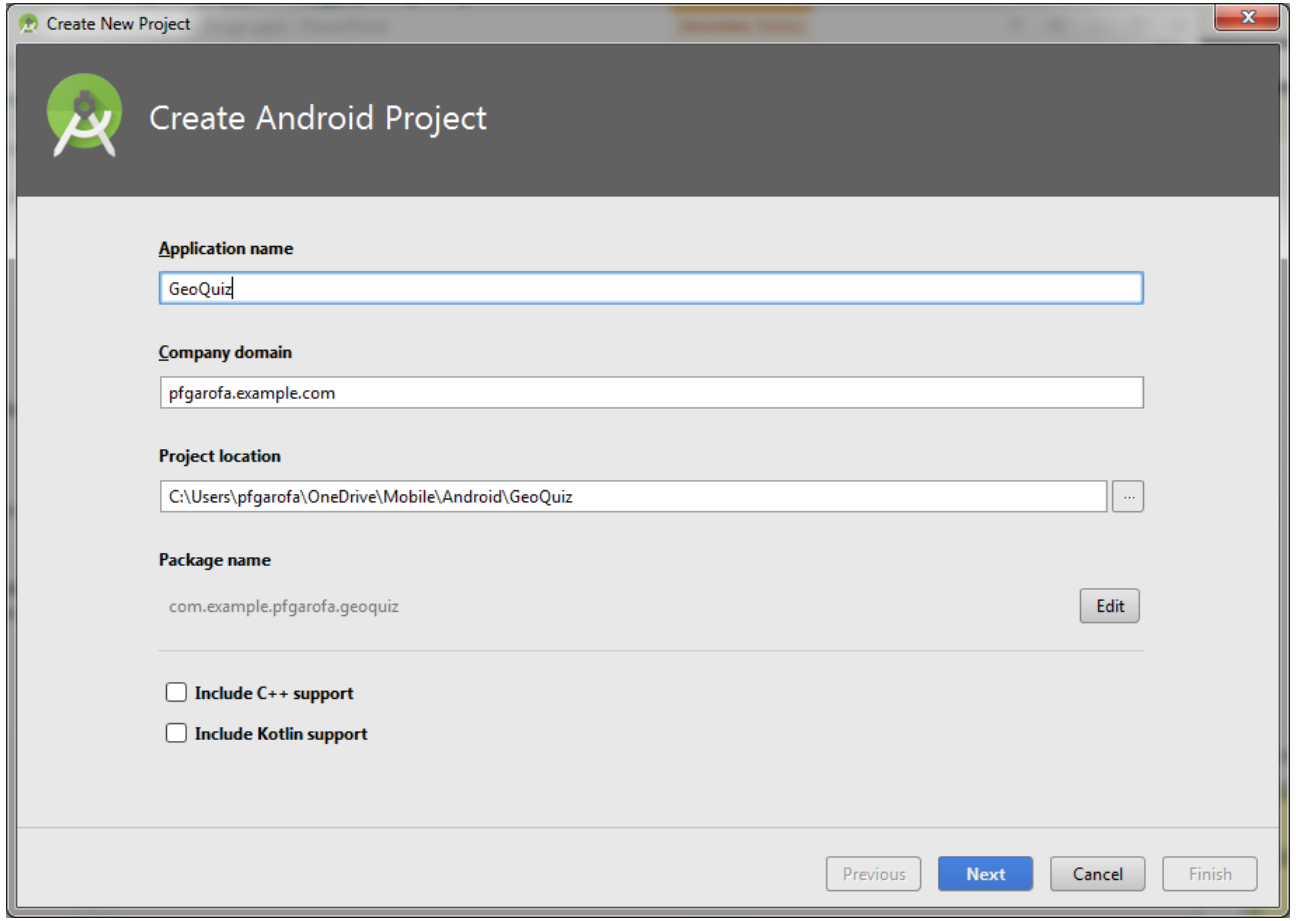
App Basics

- Activity
  - An activity in Android is a unit of execution combined with a user interface.
  - It is an instance of the Activity class (or AppCompatActivity)
  - It is responsible for managing user interaction on a single screen.
- Layout
  - Defines a set of user interface controls, their visual formats, and their positions and sizes on a screen.
  - Android layouts are XML format file.
  - XML stands for Extensible Markup Language.

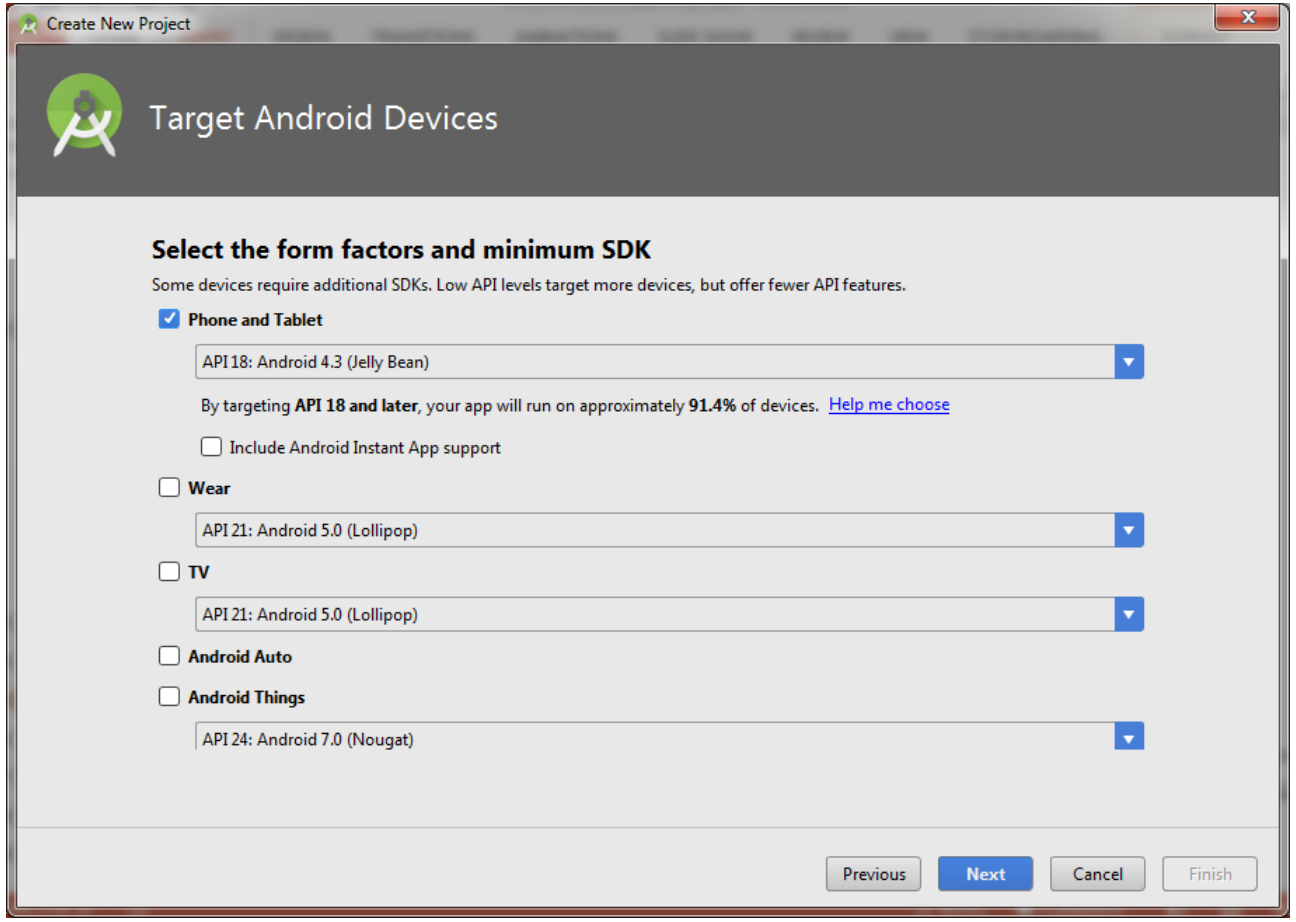
Creating an Android Project



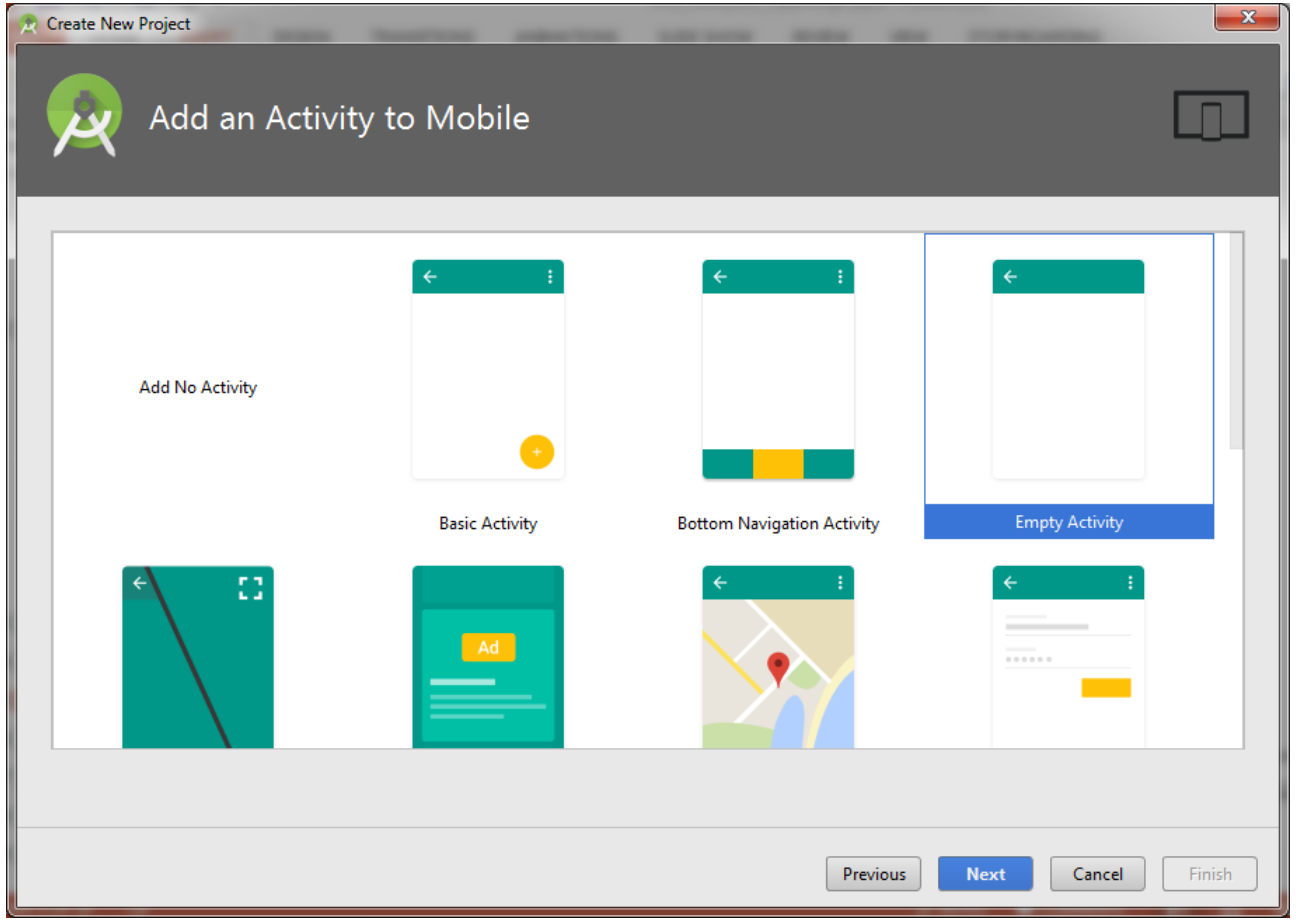
Creating an Android Project (Part 2)



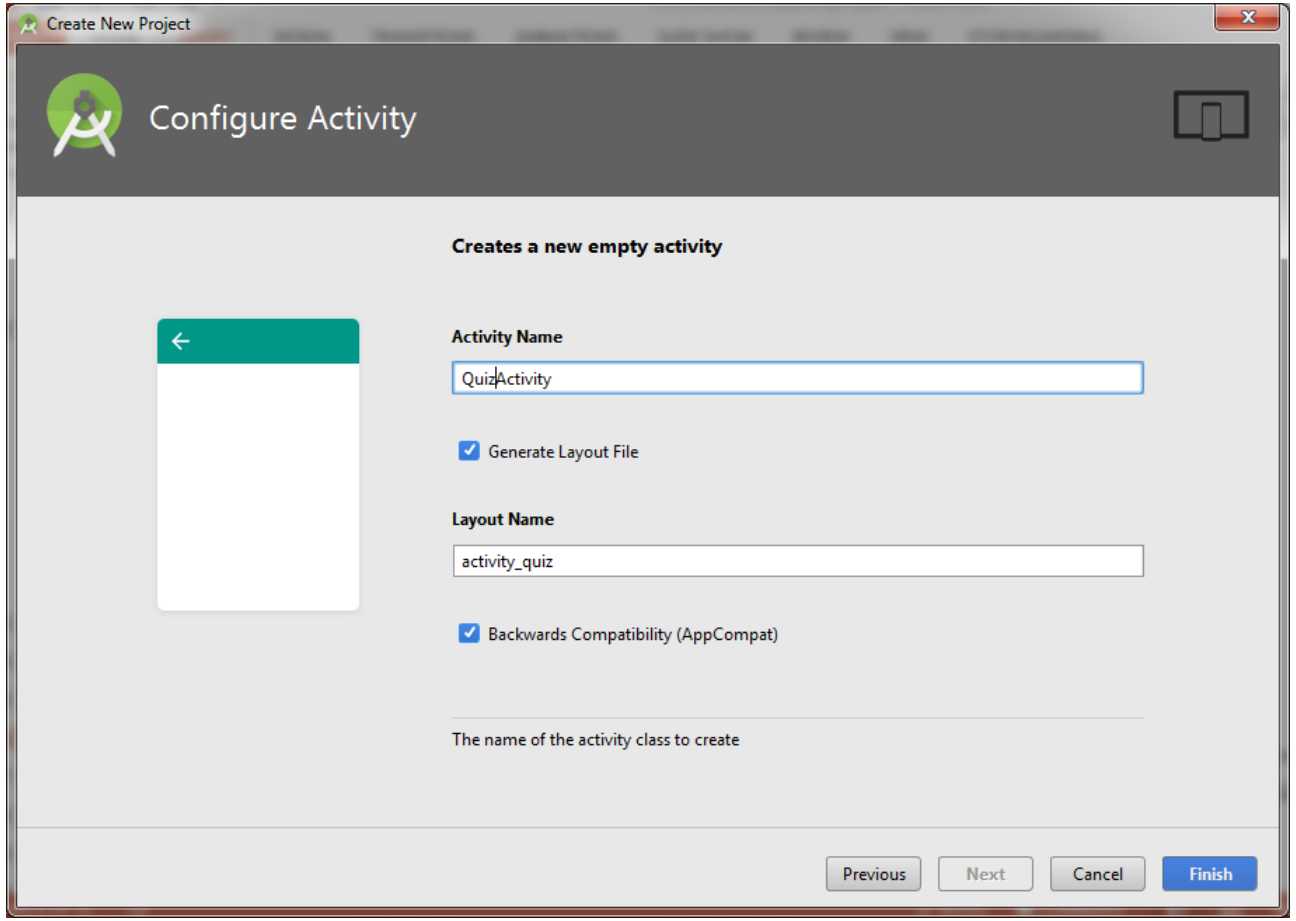
Creating an Android Project (Part 3)



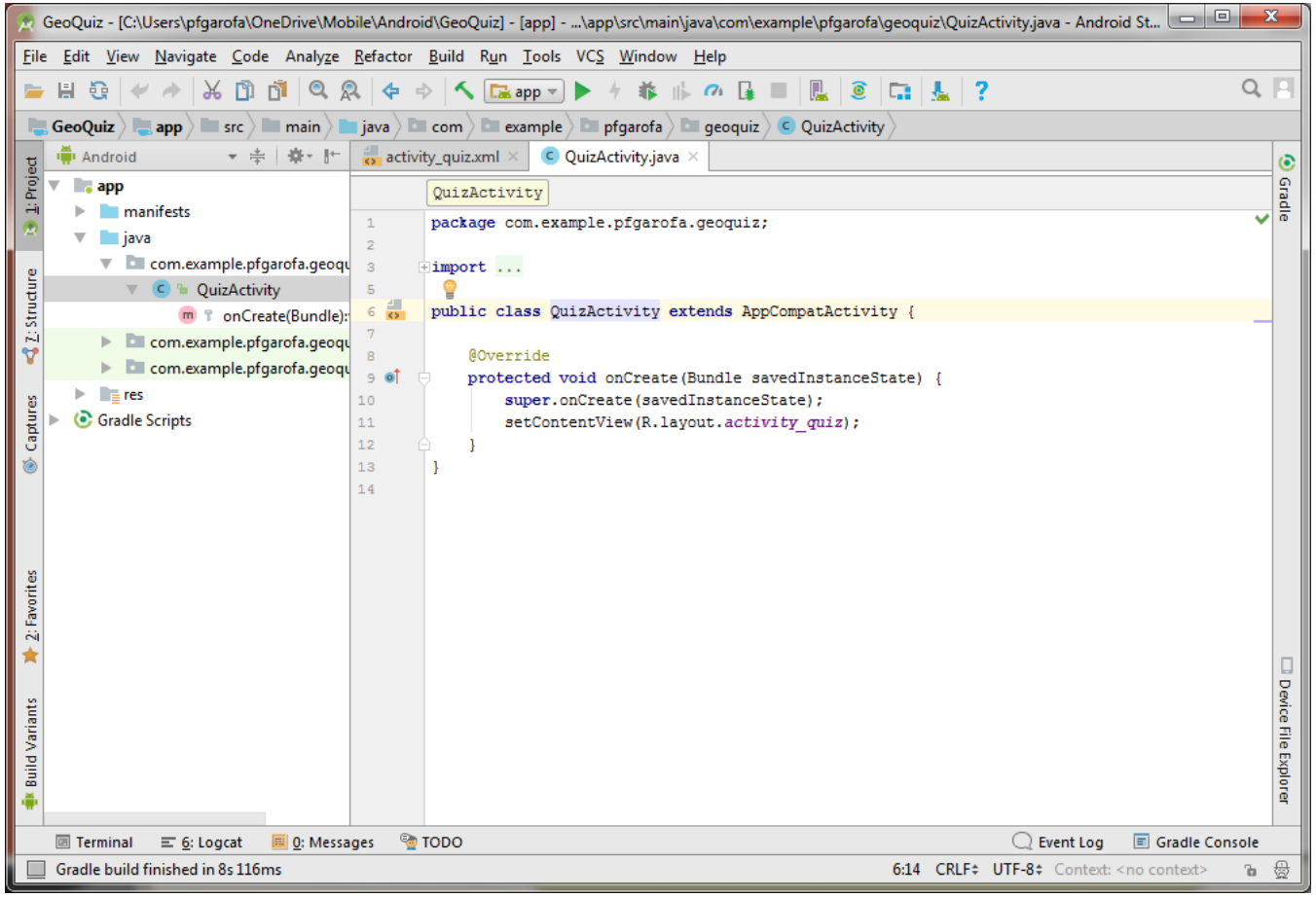
Creating an Android Project (Part 4)



Creating an Android Project (Part 4)



Basic Project Created



Laying Out the User Interface

GeoQuiz - [C:\Users\pfgarofa\OneDrive\Mobile\Android\GeoQuiz] - [app] - ...\app\src\main\res\layout\activity_quiz.xml - Android Studio 3.0.1

File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help

app

GeoQuiz > app > src > main > res > layout > activity_quiz.xml

Android

- app
  - manifests
  - java
    - com.example.pfgarofa.geoqu
      - QuizActivity
        - onCreate(Bundle):
    - com.example.pfgarofa.geoqu
    - com.example.pfgarofa.geoqu
  - res
    - drawable
    - layout
      - activity_quiz.xml
    - mipmap
    - values
  - Gradle Scripts

activity_quiz.xml | QuizActivity.java

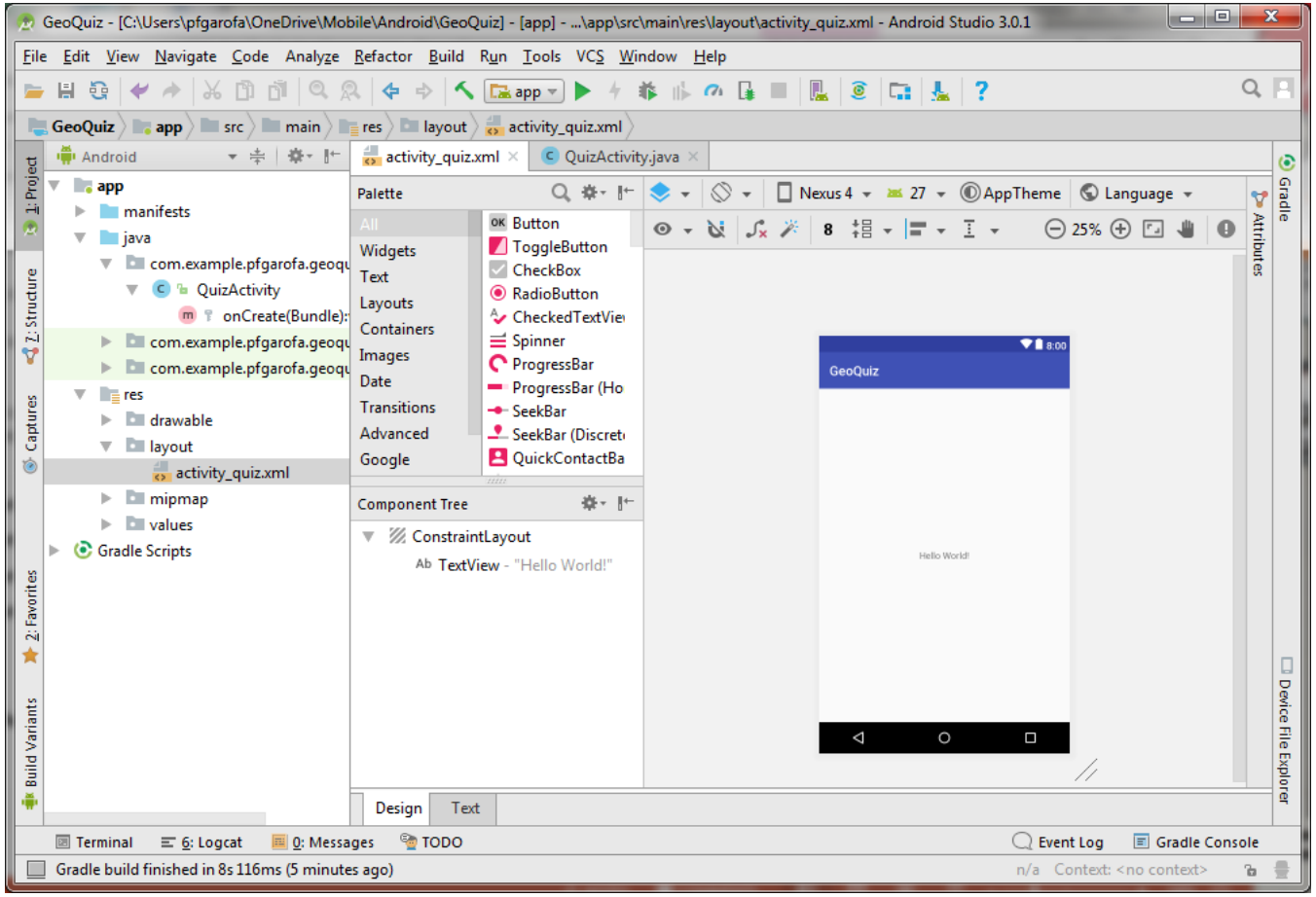
```
<?xml version="1.0" encoding="utf-8"?>
<android.support.constraint.ConstraintLayout xmlns:android="http://schemas.android.com/apk/r
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context="com.example.pfgarofa.geoquiz.QuizActivity">

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Hello World!"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintRight_toRightOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

</android.support.constraint.ConstraintLayout>
```

Design | Text

Terminal | 6: Logcat | 0: Messages | TODO | Event Log | Gradle Console

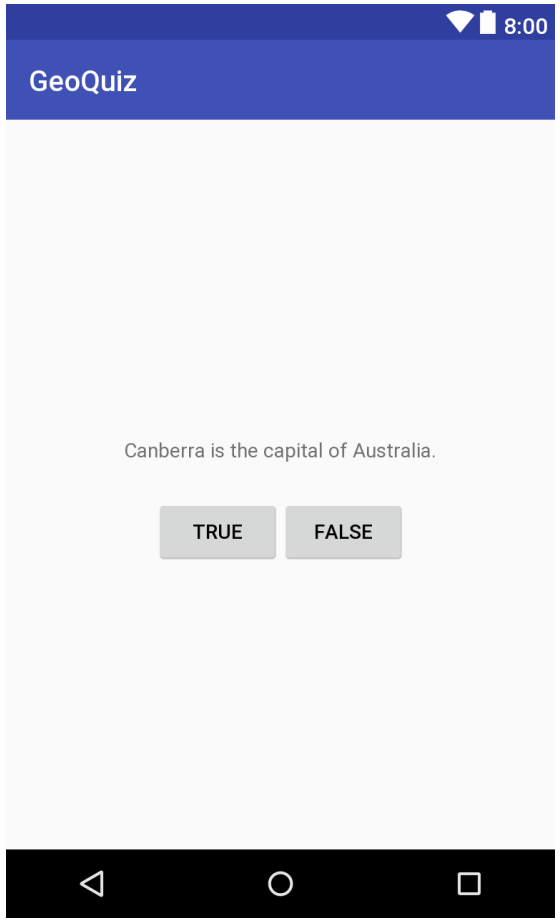
Gradle build finished in 8s 116ms (2 minutes ago) 1:1 CRLF UTF-8 Context: <no context>

Laying Out the User Interface (Part 2)



Laying Out the User Interface (Part 3)

This is the layout we're going to create.



Laying Out the User Interface (Part 3)

Change layout to a LinearLayout by typing over Constraint Layout.

```
<?xml version="1.0" encoding="utf-8"?>
<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"
    tools:context="com.example.pfgarofa.geoquiz.QuizActivity">

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Hello World!"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintRight_toRightOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

</LinearLayout>
```

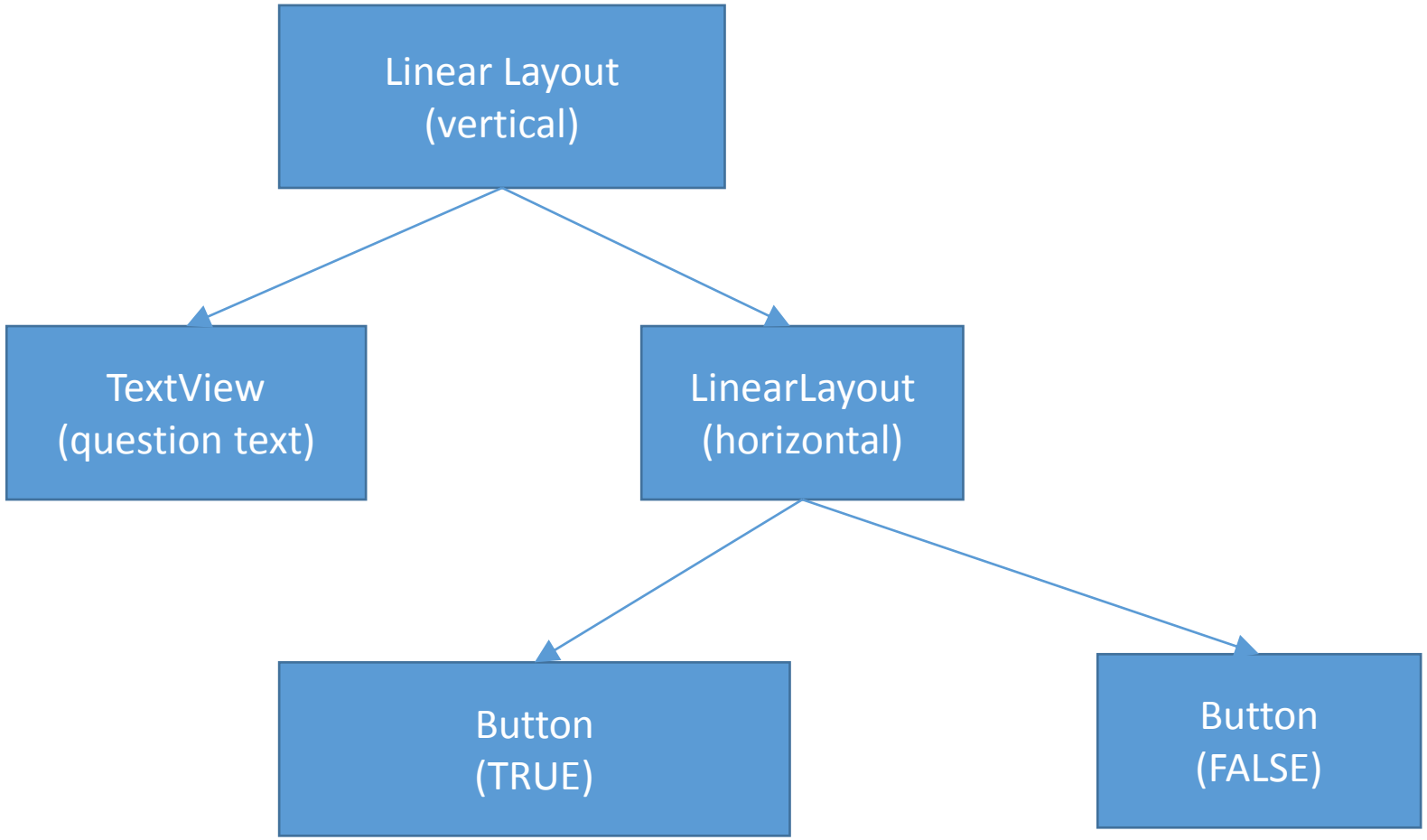
Laying Out the User Interface (Part 5)

1. We’re going to use the Design View and the Attributes panel to add and configure the widgets.
2. Set the LinearLayout orientation to vertical in the Attributes panel.
3. Set its gravity to center.
4. Drag out a TextView onto the canvas.
5. Drag out a LinearLayout below the Text view.
6. Set its orientation to horizontal.
7. Drag a Button onto the horizontal LinearLayout.
8. Drag another button to the right of the first Button.
9. Set the layout widths and heights to wrap content.

Laying Out the User Interface (Part 7)

```xml
<?xml version="1.0" encoding="utf-8"?>
<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.geoquiz.QuizActivity">

    <TextView
        android:id="@+id/textView"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:padding="24dp"
        android:text="Canberra is the capital of Australia."
/>

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

        <Button
            android:id="@+id/button"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_weight="1"
            android:text="True" />

        <Button
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_weight="1"
            android:text="False" />
    </LinearLayout>
</LinearLayout>
```

The View Hierarchy

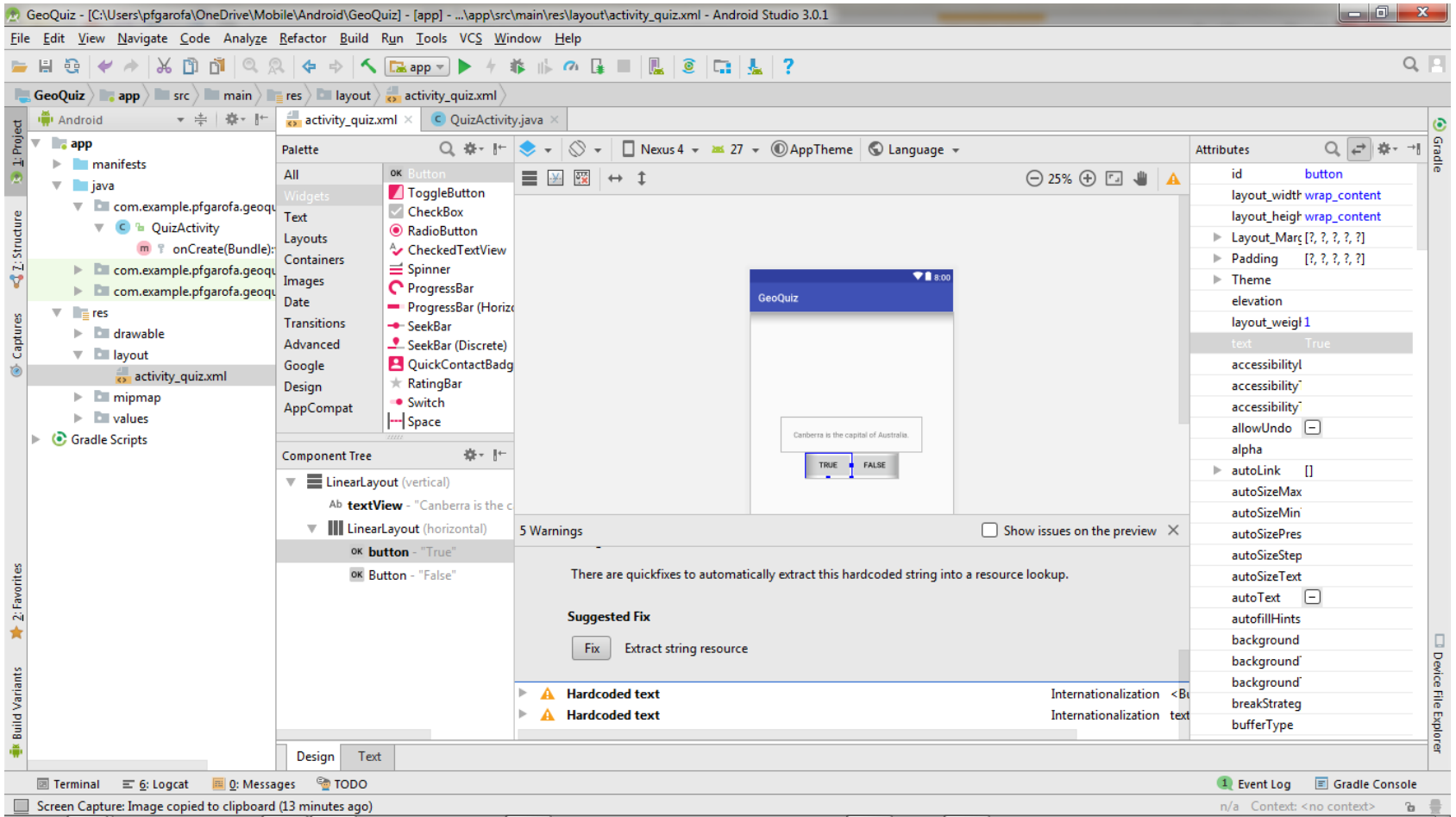


Widget Attributes

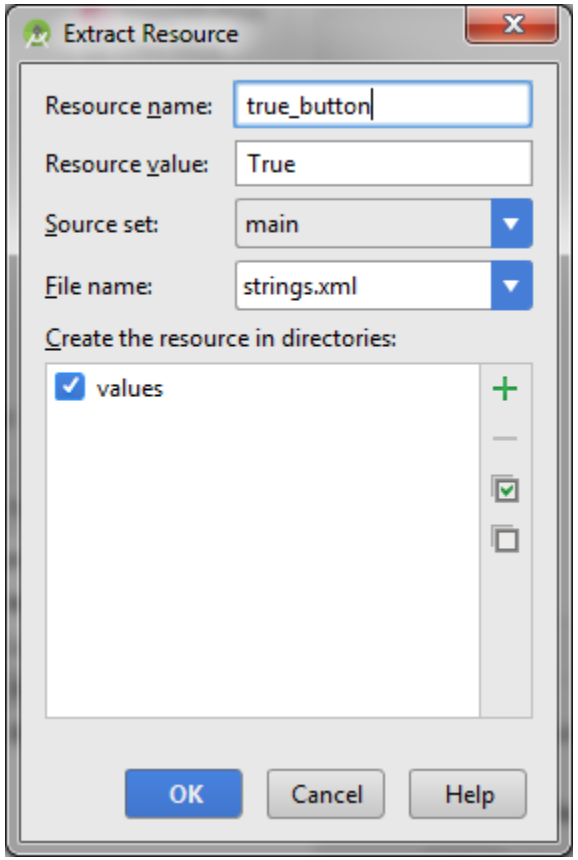
- android:layout_width and android:layout_height
  - match_parent – view expands to the size of the parent view.
  - wrap_content – view borders hug content.
- android:orientation – for LinearLayout, either vertical or horizontal.
- android:text – for TextView, Button, and others. The displayed text content.

String Resources

We're going to let Android Studio help create the string resources.



String Resources (Part 2)



Enter in the resource name for each of the controls:

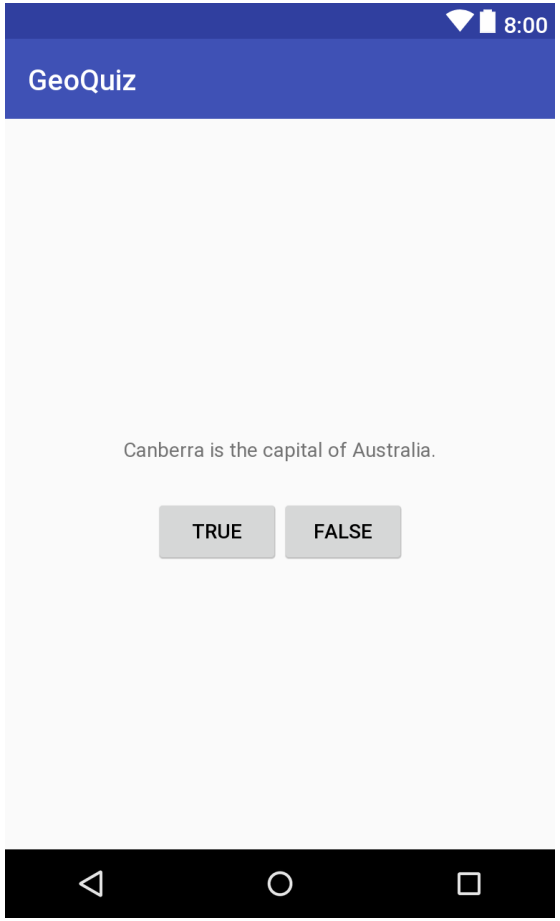
- question_text
- true_button
- false_button

String Resources (Part 3)

Click on res/values/strings.xml in the Navigation panel and you'll see the following string resource entries.

```
<resources>
    <string name="app_name">GeoQuiz</string>
    <string name="true_button">True</string>
    <string name="false_button">False</string>
    <string name="question_text">Canberra is the capital of Australia.</string>
</resources>
```

Review the Layout



From Layout XML to View Objects

- The activity you created will “inflate” the layout to display it on the screen.

```
package com.example.pfgarofa.geoquiz;

import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;

public class QuizActivity extends AppCompatActivity {

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

Resources

- A layout is a resource, as are strings, icons, and other static data your app needs.
- Editable resources are found in the app/res directory.
  - Layouts, like activity_quiz.xml, are in app/res/layout
  - Strings are in app/res/values/strings.xml
- Resources get compiled into the hidden **R.java** file.
  - Open the Project view in the Navigation panel.
  - To see it, open GeoQuiz\app\build\generated\source\r\debug\com.your.package.name\geoquiz\R.java

Resource IDs

- Resources have IDs.
- That’s how you initially reference them for instantiation as objects in your program.
- You do this with the findViewById() method of Activity.

Add Resource IDs to the Widgets

- You can either edit the XML or use the Attribute panel to add IDs to the widgets.
- Give your buttons the IDs **true_button** and **false_button**.

Instantiating Widgets

```
package com.example.pfgarofa.geoquiz;

import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.widget.Button;

public class QuizActivity extends AppCompatActivity {

    private Button mTrueButton;
    private Button mFalseButton;

    @Override
    protected 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);
    }
}
```

Use Alt-Enter (when prompted) to automatically add imports

Add these lines.

Setting Listeners

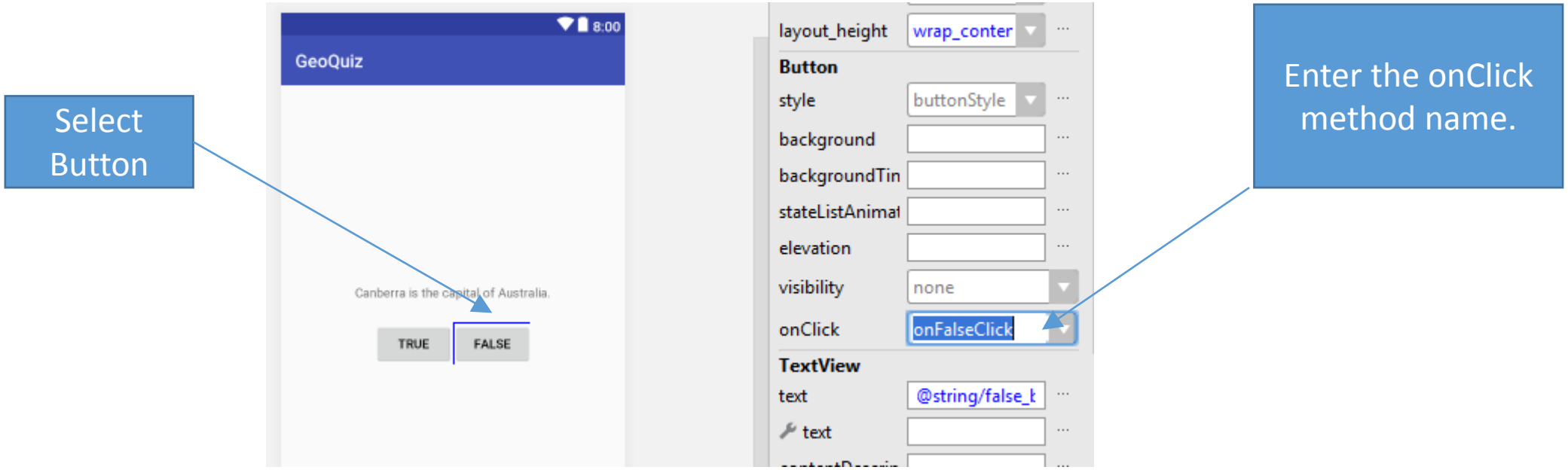
Add the following bolded lines. Use Andrdoid Studio's code completion feature by starting to type the suggested method name.

```
@Override
    protected 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) {
                // will do something soon!
            }
        });
        mFalseButton = (Button) findViewById(R.id.false_button);
    }
}
```

Anonymous Inner Class

Another Way to Set an onClick Listener

In the layout Detail View, select the FALSE button and in the onClick attribute, enter the name of response method.



Another Way to Set an onClick Listener

Add the listener method in your activity. It’s declared public, returns void and takes a View object as its only argument.

```
@Override
   protected 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) {
               // will do something soon!
           }
       });
       mFalseButton = (Button) findViewById(R.id.false_button);
   }

   public void onFalseClick(View v) {
       // will also do something soon!
   }
```

Toasts

- Toasts are brief little pop-up messages your app can present to users.
- Use them to inform them of some pertinent app action.
- Use code completion, or better yet, an editor “live template”.
- For code complete, start typing the line and select from the suggested line completions in the menu that pops up automatically.
- For Live Templates (in Settings, Editor), type the keyword (“Toast”) and press enter.
- Click on the light bulb icon to the right of the line to add strings to your resources.

Toasts (Part 2)

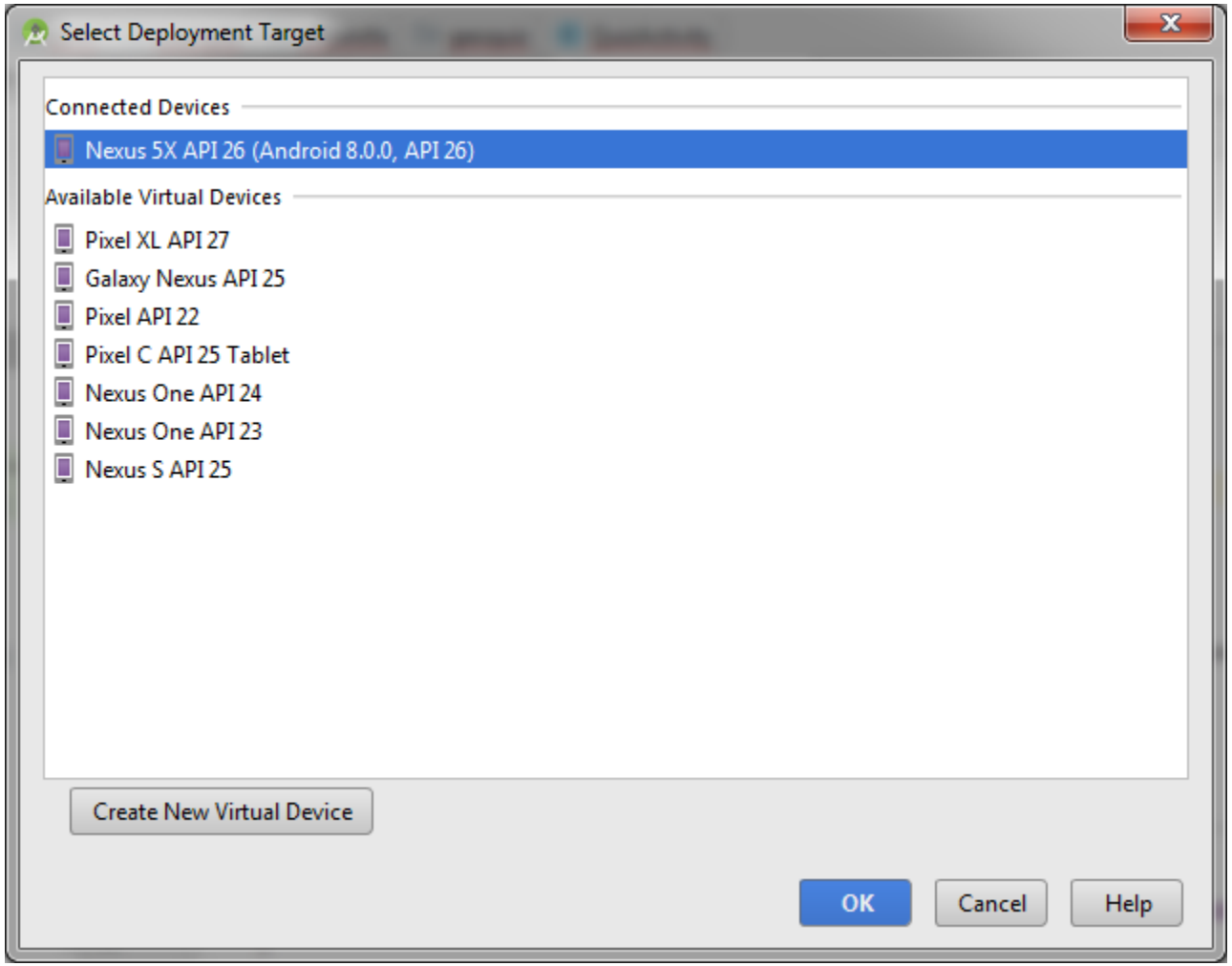
Add the following toasts to your on-click listener methods.

```
mTrueButton = (Button) findViewById(R.id.true_button);
        mTrueButton.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                Toast.makeText(QuizActivity.this, "True clicked", Toast.LENGTH_SHORT).show();
            }
        });
        mFalseButton = (Button) findViewById(R.id.false_button);
    }

    public void onFalseClick(View v) {
        Toast.makeText(this, "False clicked!", Toast.LENGTH_SHORT).show();
    }
```

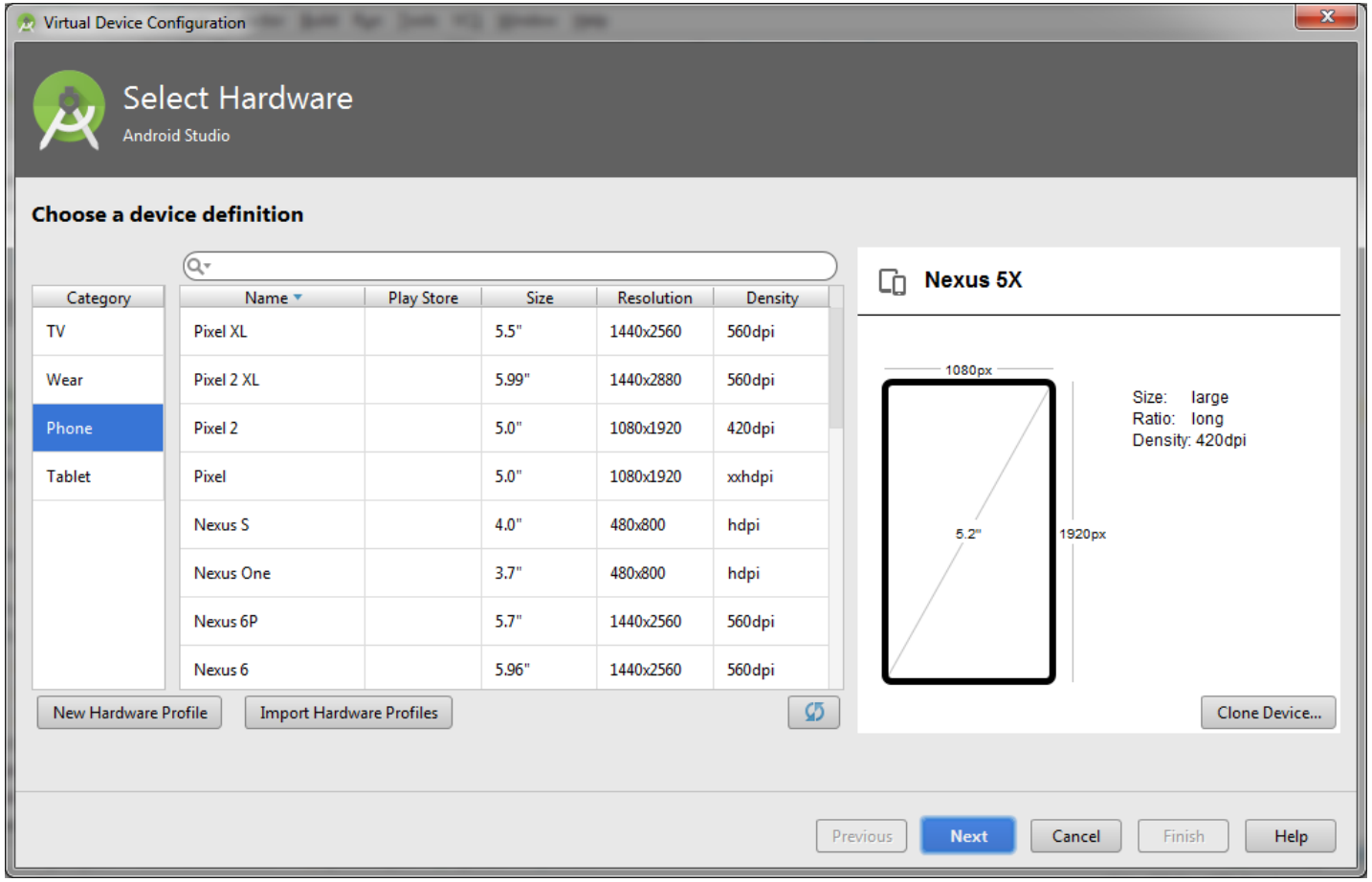
Scope qualifier needed here.

Running on an Emulator

- Click the green run button on Android Studio's toolbar and you'll be able to run your app on a software emulation of an actual device.
- If you haven't run an emulator before, you might need to create a new virtual device definition.



Creating A Virtual Device



Make It Your Own

1. Implement the Challenge at the end of Chapter 1.
   - Customize the toast.
2. Add a modification of your own design.
   - It can be something like changing a widget or layout attribute or something a more interesting.