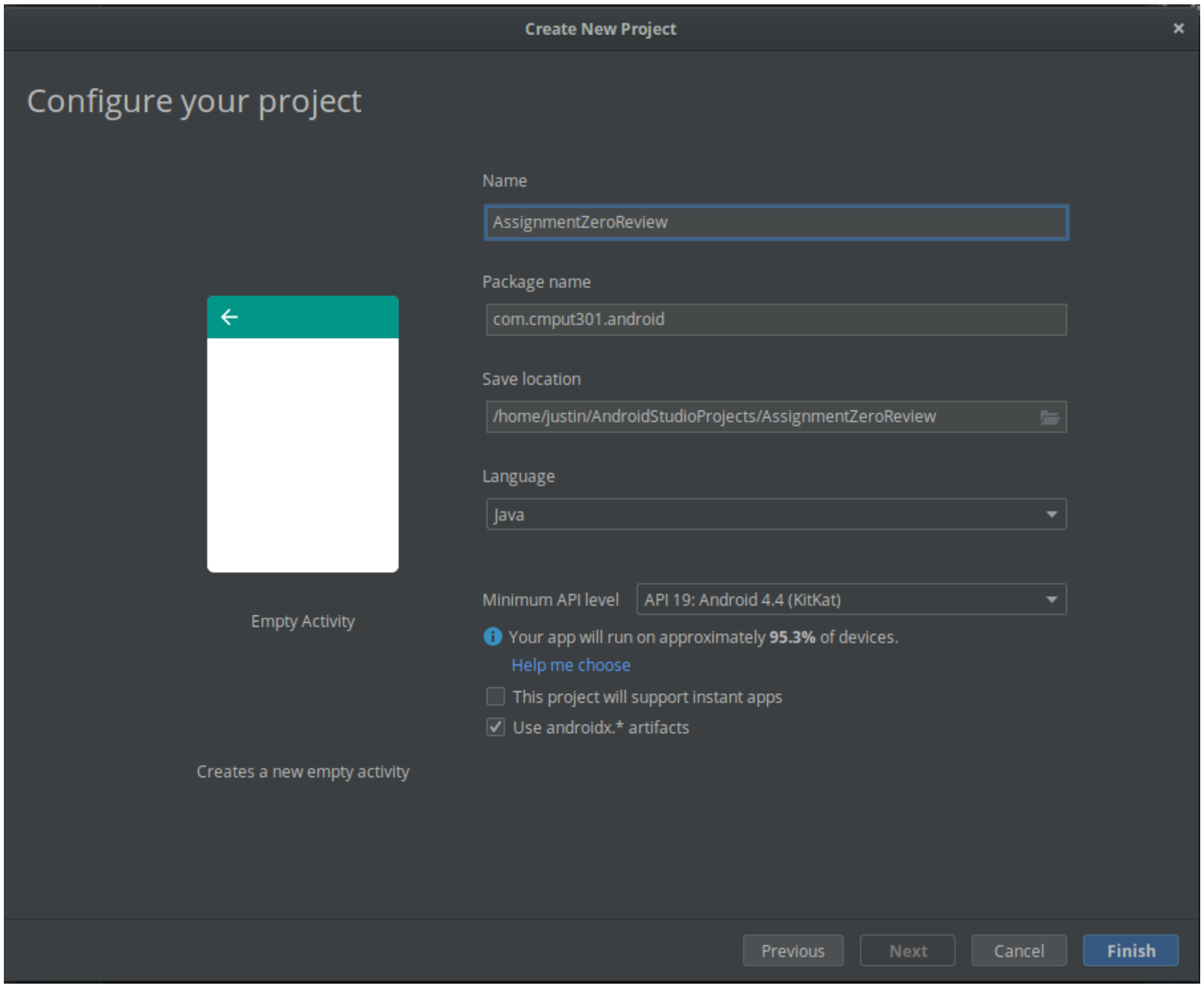
**Make a new Android project** (File → New → New Project)

Choose **Empty Activity**
Name it **AssignmentZeroReview**, make sure the Language selected is "**Java**". Everything else can be left alone.



Create New Project

Configure your project

Name: AssignmentZeroReview

Package name: com.cmput301.android

Save location: /home/justin/AndroidStudioProjects/AssignmentZeroReview

Language: Java

Minimum API level: API 19: Android 4.4 (KitKat)

Your app will run on approximately **95.3%** of devices.

Help me choose

- [ ] This project will support instant apps
- [x] Use androidx.* artifacts

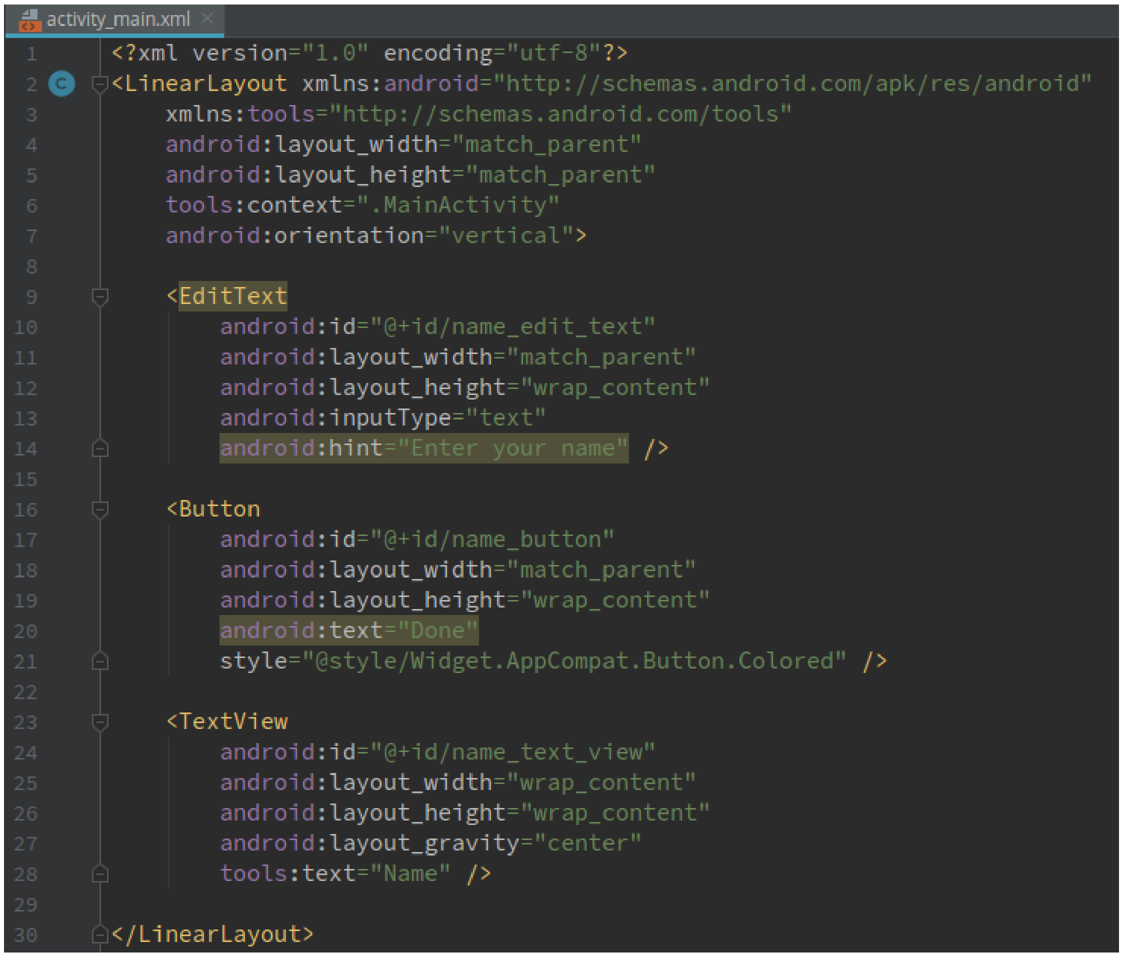
Empty Activity

Creates a new empty activity

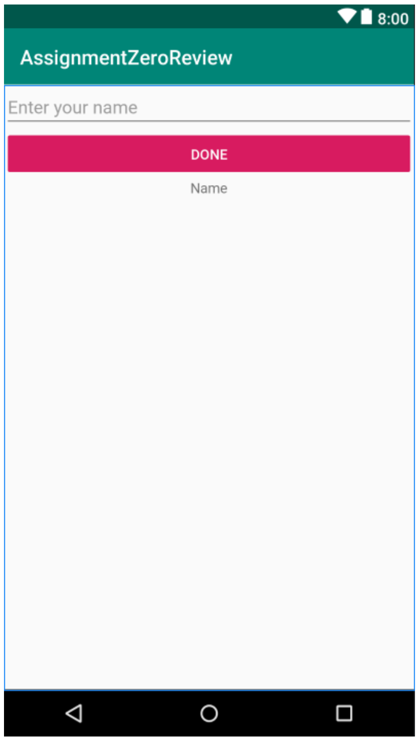
Click Finish

Go to **activity_main.xml**, change the root view to a **LinearLayout** instead of ConstraintLayout.
Add **android:orientation="vertical"** to the LinearLayout.
Add an EditText, Button and TextView. - check the picture below for how the file should look like.

**Note:** The android:hint attribute is exactly what it sounds like. It gives the app user a hint of what is expected of them in the EditText.
style in this case allows us to use a coloured theme for the Button.
android:layout_gravity lets us position the View within its ViewGroup. In this case, we center it.

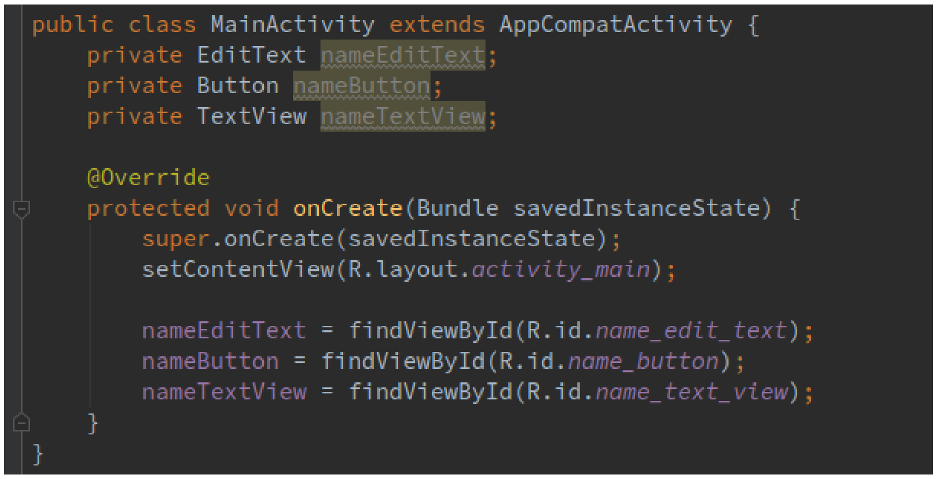

activity_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity"
    android:orientation="vertical">

    <EditText
        android:id="@+id/name_edit_text"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:inputType="text"
        android:hint="Enter your name" />

    <Button
        android:id="@+id/name_button"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Done"
        style="@style/Widget.AppCompat.Button.Colored" />

    <TextView
        android:id="@+id/name_text_view"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="center"
        tools:text="Name" />

</LinearLayout>
```



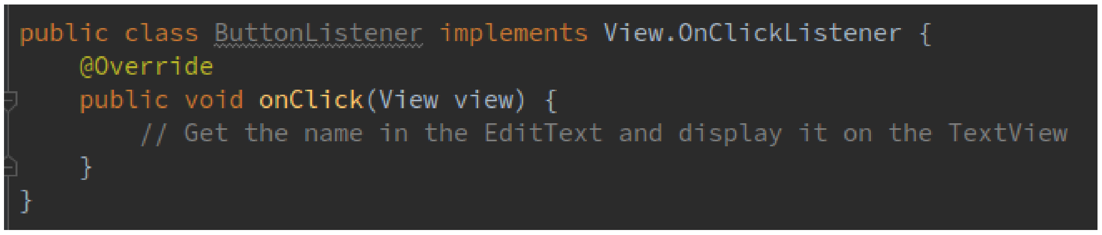
Run the app and make sure it's working. We will make it so that when the button is clicked, we get the name from the EditText and display it in the TextView.

Go to MainActivity.java and use findViewById to get references to all of the views.



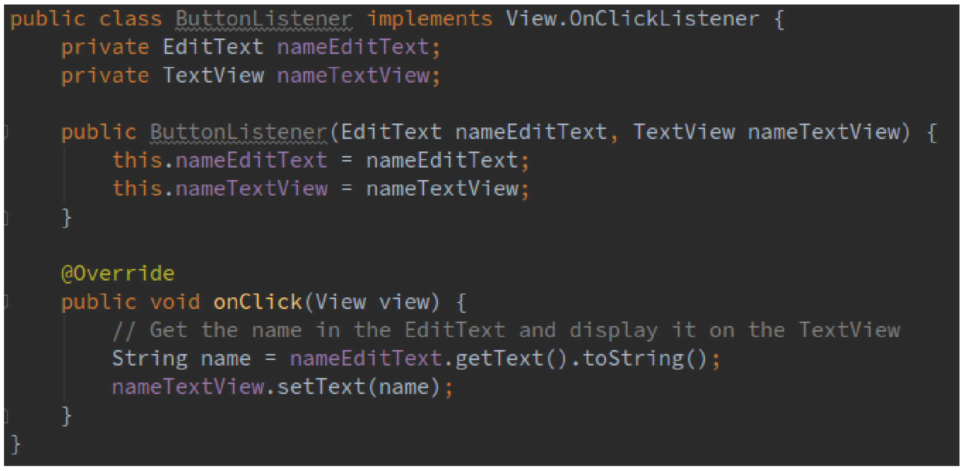
```java
public class MainActivity extends AppCompatActivity {
    private EditText nameEditText;
    private Button nameButton;
    private TextView nameTextView;

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

        nameEditText = findViewById(R.id.name_edit_text);
        nameButton = findViewById(R.id.name_button);
        nameTextView = findViewById(R.id.name_text_view);
    }
}
```

We want to define some behaviour for when the Button is clicked. To do this, we need to use **setOnClickListener**, but this method takes in a class that implements View.OnClickListener. Make a new class "ButtonListener". Make it implement View.OnClickListener.



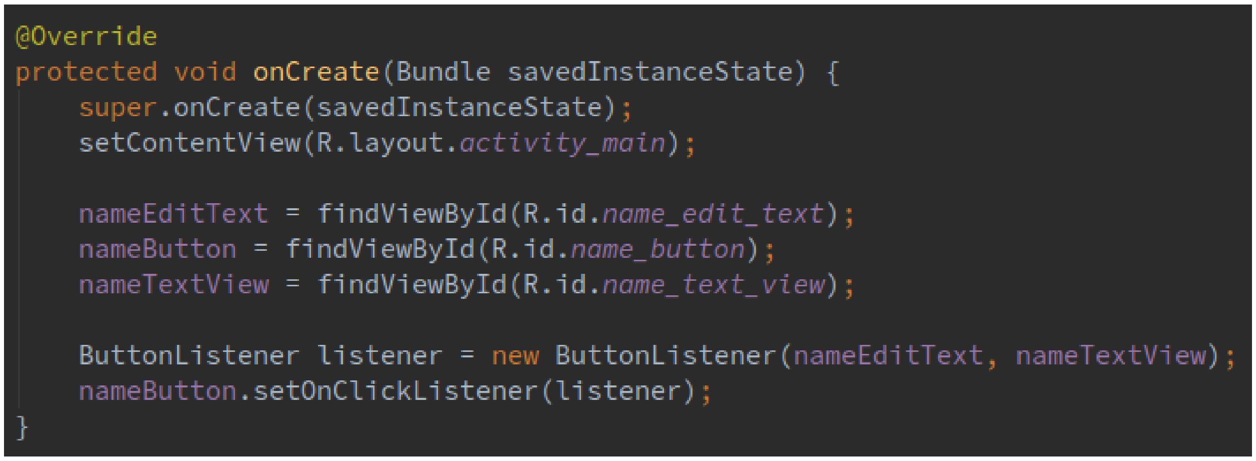
```java
public class ButtonListener implements View.OnClickListener {
    @Override
    public void onClick(View view) {
        // Get the name in the EditText and display it on the TextView
    }
}
```

We want to get the text from the EditText, and set it onto the TextView. In that case, we'll need the references to the EditText and TextView. The easiest way to do this is to add it as instance attributes and pass it into the constructor.



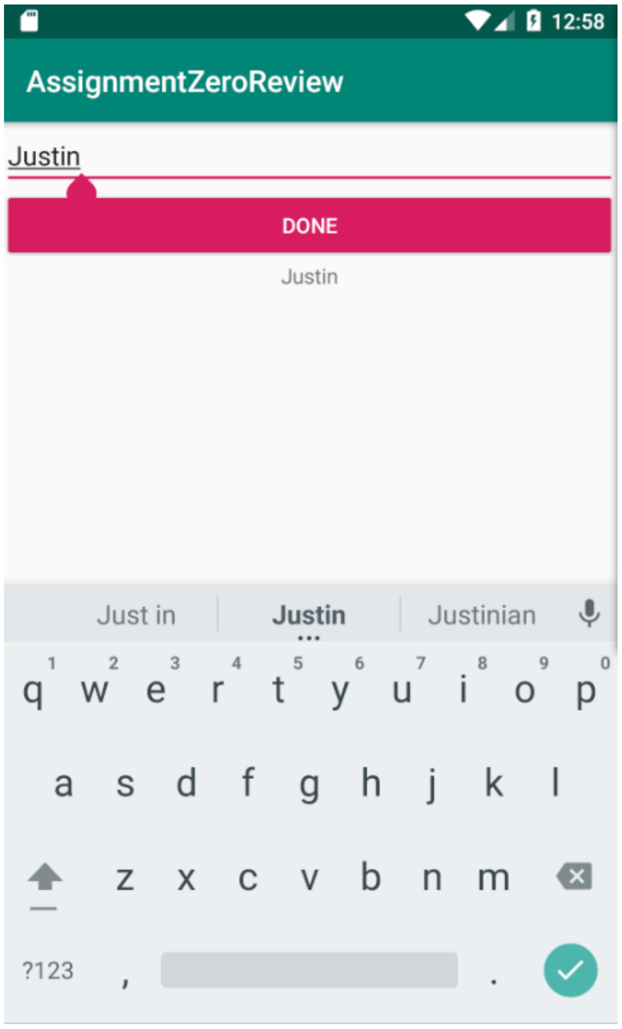
```java
public class ButtonListener implements View.OnClickListener {
    private EditText nameEditText;
    private TextView nameTextView;

    public ButtonListener(EditText nameEditText, TextView nameTextView) {
        this.nameEditText = nameEditText;
        this.nameTextView = nameTextView;
    }

    @Override
    public void onClick(View view) {
        // Get the name in the EditText and display it on the TextView
        String name = nameEditText.getText().toString();
        nameTextView.setText(name);
    }
}
```

Now we can go to MainActivity, make a new instance of this ButtonListener, and set it on the Button.



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

    nameEditText = findViewById(R.id.name_edit_text);
    nameButton = findViewById(R.id.name_button);
    nameTextView = findViewById(R.id.name_text_view);

    ButtonListener listener = new ButtonListener(nameEditText, nameTextView);
    nameButton.setOnClickListener(listener);
}
```

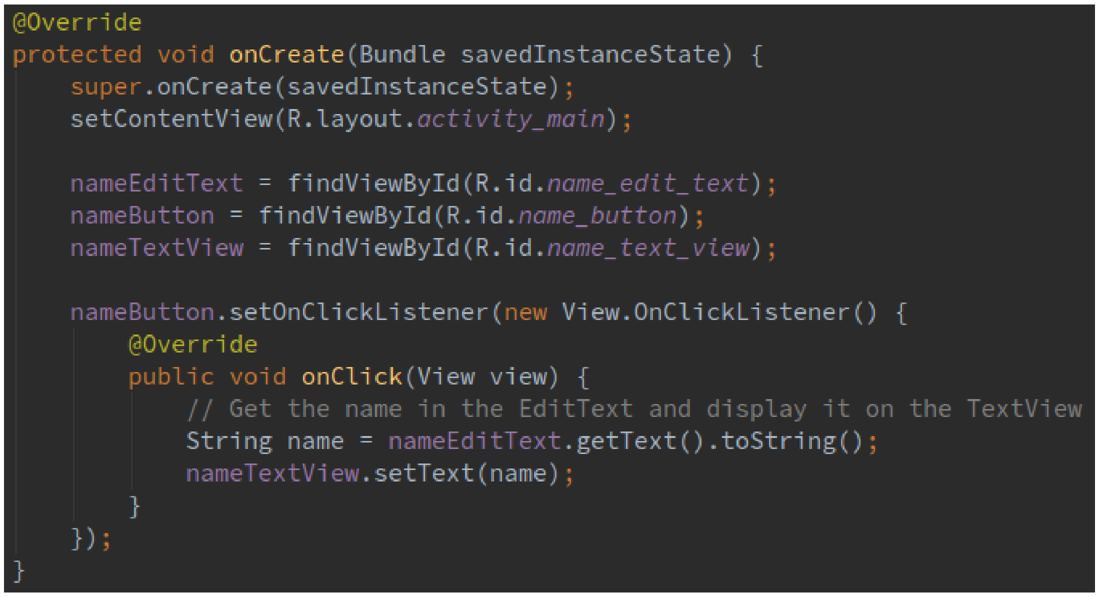
Run the program and make sure it works!



It'd be nice if we didn't have to go through the trouble of making a new class, having it implement View.OnClickListener, and sending in the other views through the constructor…

We can do this using an **anonymous class.** It's the exact same idea as the ButtonListener but is a shortcut - **it lets us define a new class that implements View.OnClickListener, and make an instance of it at the same time**. It's called an anonymous class since it doesn't have a name.

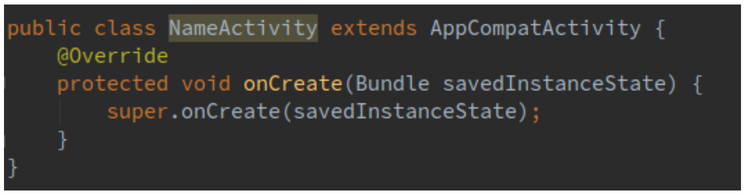
Since we can do this in onCreate of the MainActivity, we no longer have to explicitly define a new class, define EditText/TextView attributes and send it through the constructor. We can also copy the onClick() method over from ButtonListener.



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

    nameEditText = findViewById(R.id.name_edit_text);
    nameButton = findViewById(R.id.name_button);
    nameTextView = findViewById(R.id.name_text_view);

    nameButton.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View view) {
            // Get the name in the EditText and display it on the TextView
            String name = nameEditText.getText().toString();
            nameTextView.setText(name);
        }
    });
}
```

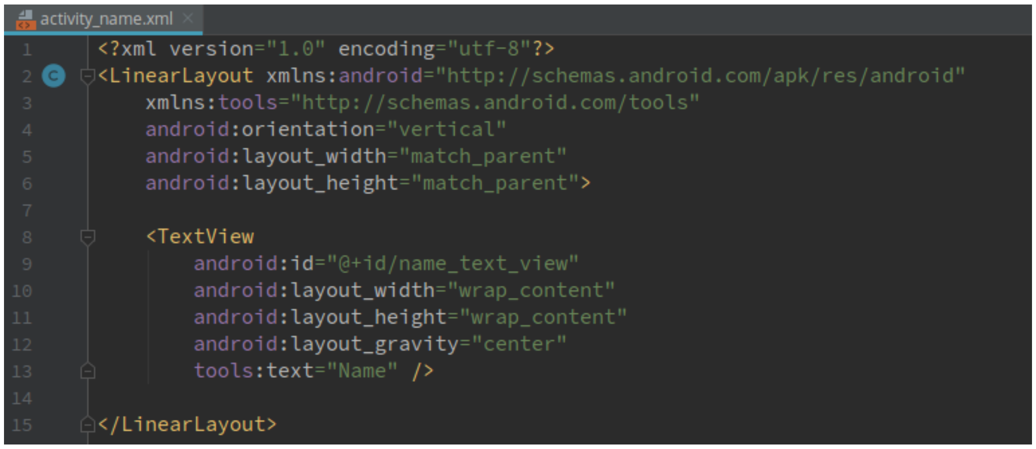
Re-run the program and ensure that it is still working as normal..

Next step: **Make a second activity**. **When we click the button on the first activity, we want to navigate to the second activity and show the name there.**
Make a new Java class **NameActivity**.



```
public class NameActivity extends AppCompatActivity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
    }
}
```

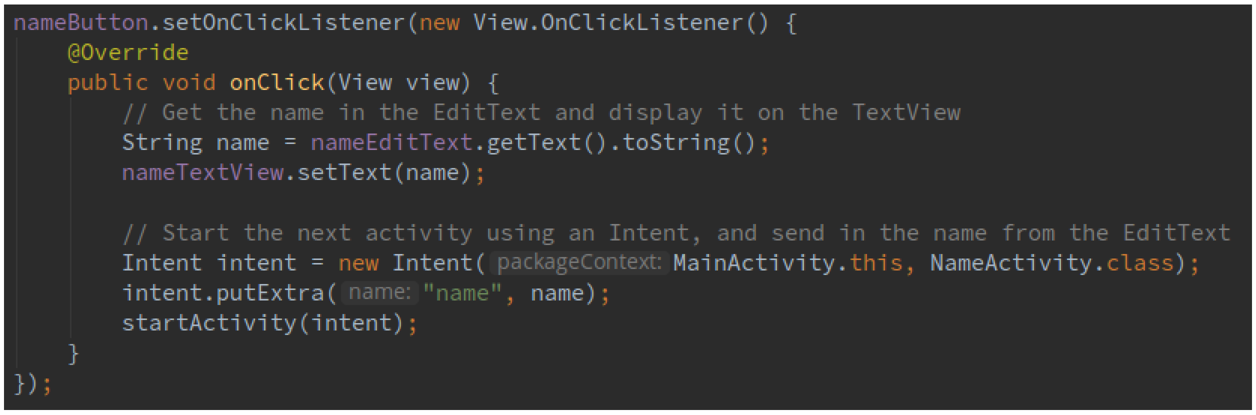
Go to res-->layout and make a **new layout resource file called activity_name.xml**. Make it a LinearLayout root element with a single TextView.



activity_name.xml

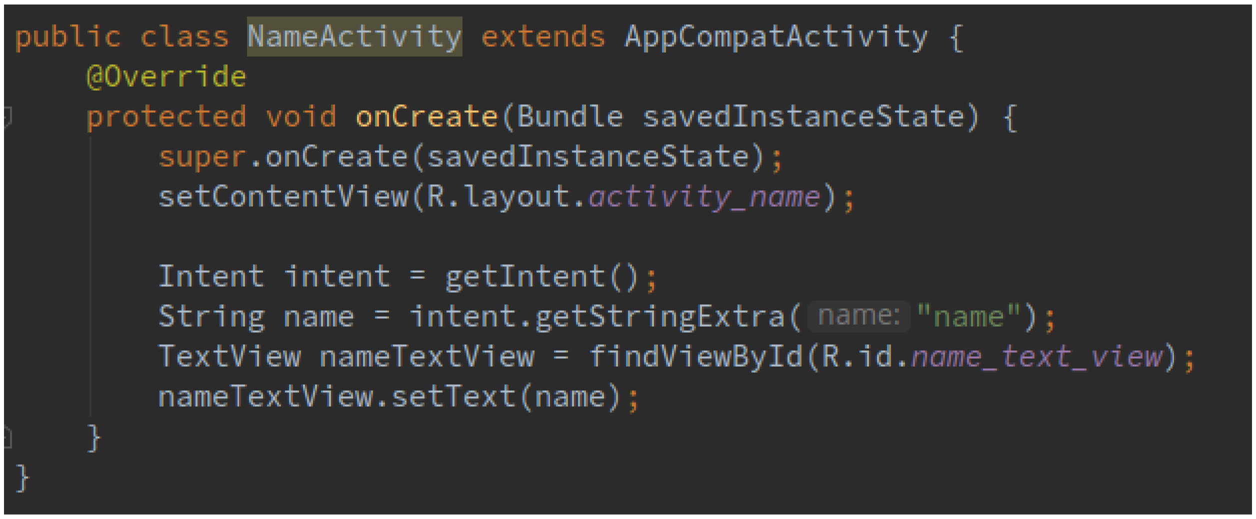
```xml
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:orientation="vertical"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    <TextView
        android:id="@+id/name_text_view"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="center"
        tools:text="Name" />

</LinearLayout>
```

In MainActivity, update the onClick method to start the new activity, sending in the name.



```java
nameButton.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        // Get the name in the EditText and display it on the TextView
        String name = nameEditText.getText().toString();
        nameTextView.setText(name);

        // Start the next activity using an Intent, and send in the name from the EditText
        Intent intent = new Intent( packageContext: MainActivity.this, NameActivity.class);
        intent.putExtra( name: "name", name);
        startActivity(intent);
    }
});
```

In the NameActivity, get the intent that the Activity was started with, and get the name out of it. Set the TextView's text to be the name.



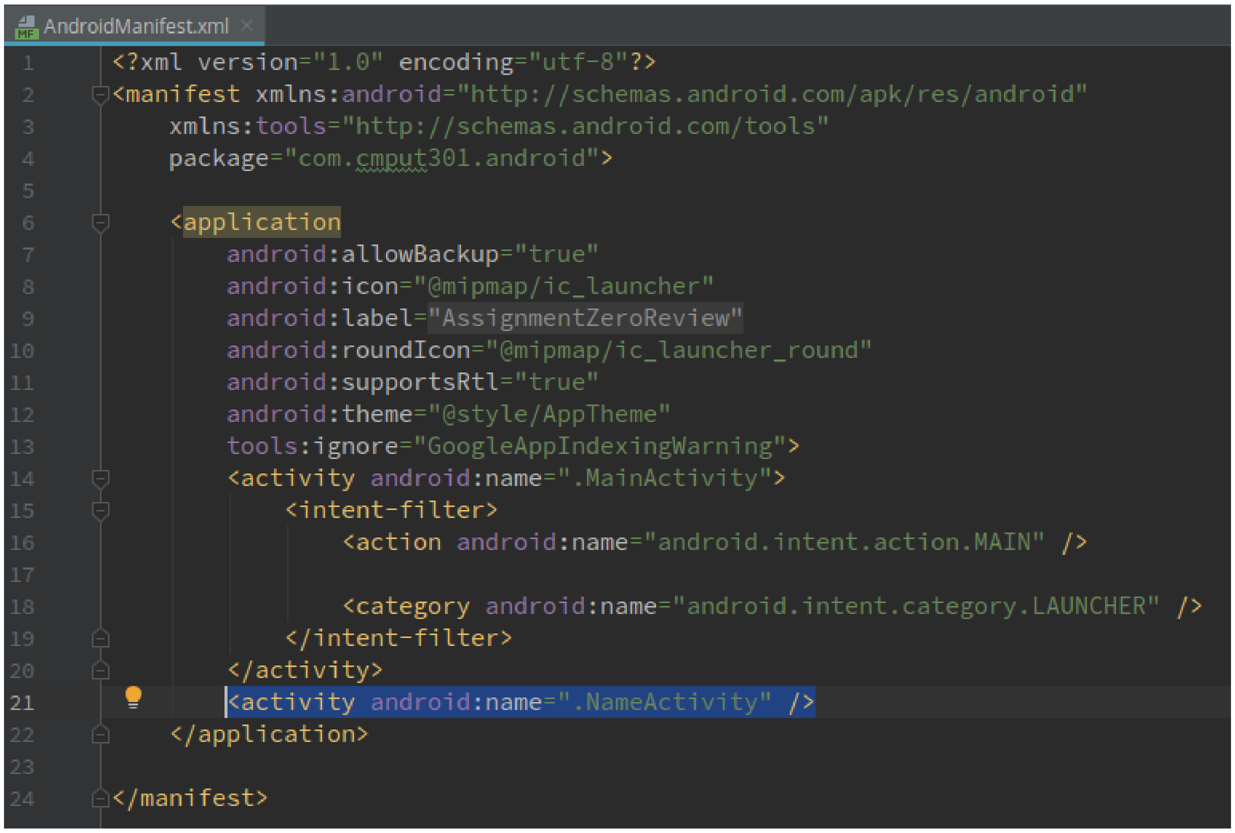
```java
public class NameActivity extends AppCompatActivity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_name);

        Intent intent = getIntent();
        String name = intent.getStringExtra( name: "name");
        TextView nameTextView = findViewById(R.id.name_text_view);
        nameTextView.setText(name);
    }
}
```

Run the app → **the app SHOULD CRASH when pressing the button!** We haven't registered the activity in the manifest.xml, so the Android OS doesn't even know about the new activity.

You can see the error log in the “Run” tab
“ActivityNotFoundException: Unable to find explicit activity class {com.cmput301.android/com.cmput301.android.NameActivity}; **have you declared this activity in your AndroidManifest.xml?”**

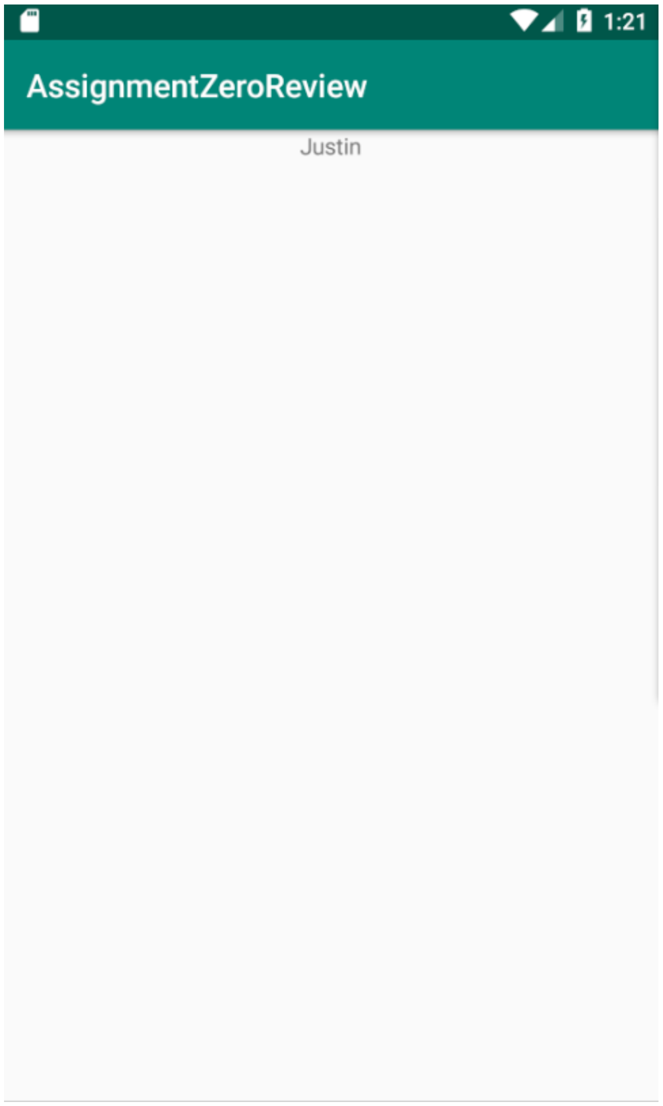
The Android manifest is where all the meta-data about your app goes. For example, if your app requires camera/internet permissions from the user, you would have to define it here. Likewise, if our app uses multiple Activities, they must be defined here so the Android OS knows about it and can run it!

Add NameActivity to the manifest.



```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    package="com.cmput301.android">

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="AssignmentZeroReview"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/AppTheme"
        tools:ignore="GoogleAppIndexingWarning">
        <activity android:name=".MainActivity">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

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

</manifest>
```

Run the app and make sure the name shows on the second screen!



Next time you make a new Activity, you can simply just right click the folder → New → Activity. It will make both the **Java class and layout file for you and add it to the manifest.**