Queens University Belfast

Android Studio

CSC3054 / CSC7054

Exercises on Toast

Week 3 Book 1

Queens University Belfast

Exercises on Toast

A `Toast` provides simple feedback about an operation in a small popup. It only fills the amount of space required for the message and the current activity remains visible and interactive. For example, navigating away from an email before you send it triggers a "Draft saved" toast to let you know that you can continue editing later. Toasts automatically disappear after a timeout.

Exercise 1 – Basic Toast

Before You Begin

Open `Android Studio` and create a new project called `"ToastBasic"`. Refer to the `'Creating your first project'` tutorial to help you create a project. Once created your project should look like figure 1. Switch from `Design` view to Text `view` .

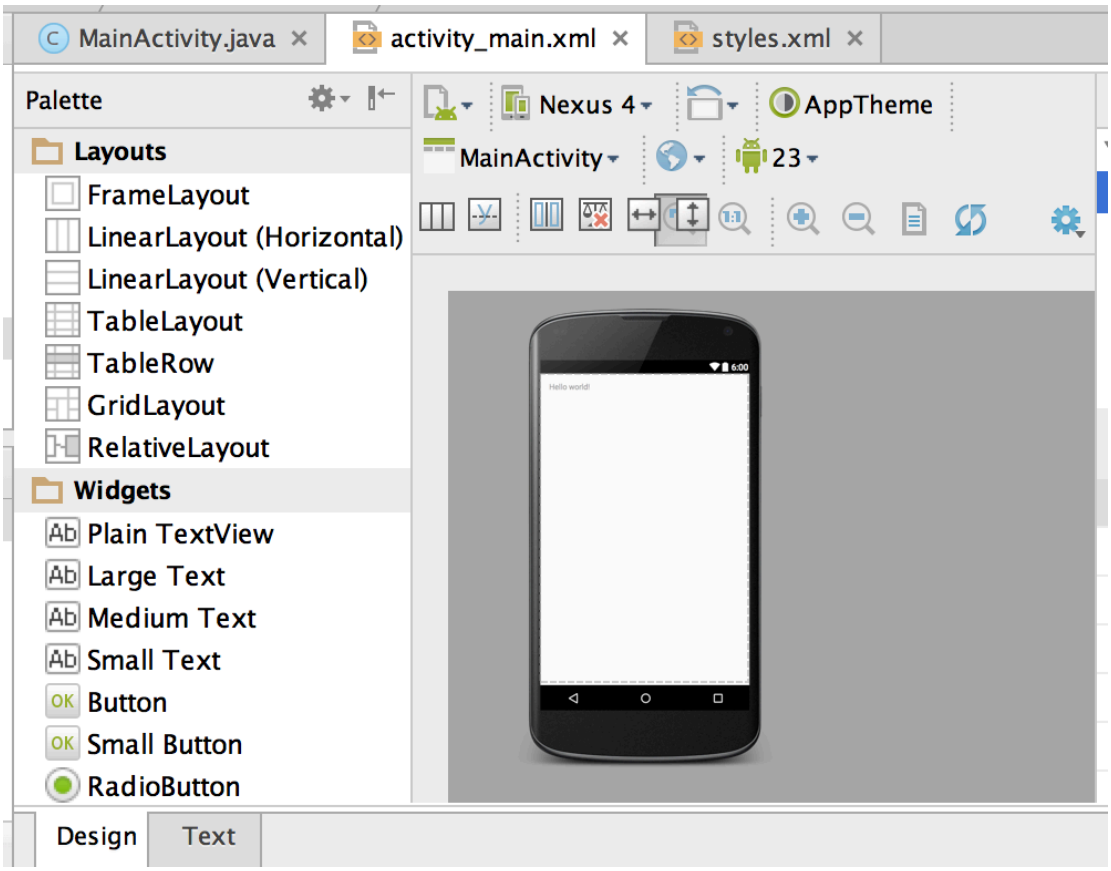

FIGURE 1 - OPEN PROJECT

What is a Basic `Toast`?

This is a plain, light `Toast` that contains a short text message. To create a basic `Toast` use the `makeText()` method. It has three parameters:

Parameters	Details
`Context`	Usually the application or activity context
`Message`	The text message to display. Use a `CharSequence` object or `String`
`Duration`	The time the pop-up is visible. There are two time options: `LENGTH_SHORT` – about 2 seconds `LENGTH_LONG` – about 4 seconds

Step 1 - Open `MainActivity.java`

Open MainActivity.java and enter in the following code into the `onCreate()` method.

```
Context context = getApplicationContext();
CharSequence message = "I like butter and jam on my toast";
int duration = Toast.LENGTH_SHORT;
Toast toastBasic = Toast.makeText(context,message,duration);
toastBasic.show();
```

Note the following:

Code	Details
`getApplicationContext()`	Gets the `context`. Most of the time you can use `this`
`message`	Use `CharSequence` although you can also use a `String`
`duration`	We're using `LENGTH_SHORT` so the `Toast` will appear for about 2 seconds

The method `show()`, will display your `Toast` on the screen as shown in Figure 2.

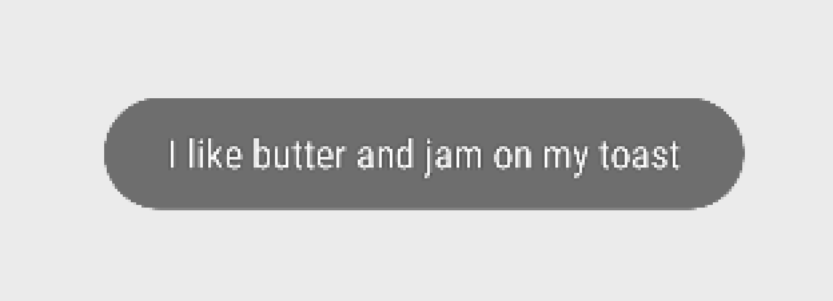

FIGURE 2 - TOAST ON APP

Step 2 Positioning the Toast

Where do you want your Toast, upstairs or downstairs? You can position the Toast anywhere on the screen. The default position for the standard `Toast` is centered near the bottom of the screen. Use the `setGravity()` method to change the position of the `Toast`. Update the `onCreate()` method so that it includes another `Toast` message:

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
    Context context = getApplicationContext();
    CharSequence message = "I like butter and jam on my toast";
    int duration = Toast.LENGTH_SHORT;

    Toast toastBasic = Toast.makeText(context,message,duration);

    int moveToastDown = 150;
    int moveToastRight = 150;
    Toast toastTop = Toast.makeText(context, message, duration);
    toastTop.setGravity(Gravity.TOP| Gravity.LEFT, moveToastDown, moveToastRight);
    toastBasic.show();
    toastTop.show();
}
```

When you run it on the emulator it should look like figure 3.

FIGURE 3 POSITIONING A TOAST

Note, both `Toasts` will be displayed onscreen.

Note the following:

Code	Detail
`getApplicationContext()`	Gets the context
`LENGTH_SHORT`	We want the `Toast` to display for about 2 seconds
`setGravity()`	Sets the position of the `Toast` on the screen. It has three parameters: • `Gravity` – specific placement of an object in a larger container. We use two constants, `TOP` and `LEFT`. This places the `Toast` in the top, left corner of the screen • `xOffset` – changing this value moves the Toast left and right • `yOffset` – changing this value moves the Toast up and down

Exercise 2 – Another `Toast` Example

Open `Android Studio` and create a new project called "`AndroidToastExample`". Switch from `Design` view to Text `view`.

Step 1 Create the XML

Open `res/layout/activity_main.xml` file as shown in figure 4:

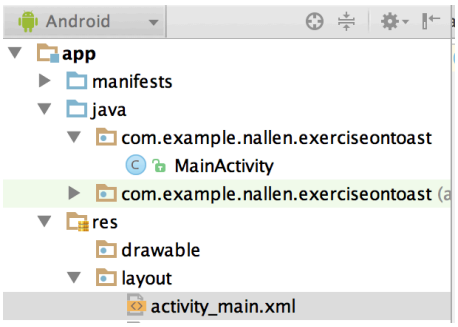

FIGURE 4 - NAVIGATING TO ACTIVITY_MAIN.XML

Set up the layout of `activity_main.xml` as follows:

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:orientation="vertical"
    tools:context=".MainActivity">
   <Button
       android:id="@+id/mainbutton"
       android:layout_width="wrap_content"
       android:layout_height="wrap_content"
       android:text="@string/button_label" />
</LinearLayout>
```

Step 2 Update strings.xml

Use the Package Explorer in Eclipse to navigate to `res/values/strings.xml` as shown in figure 5.

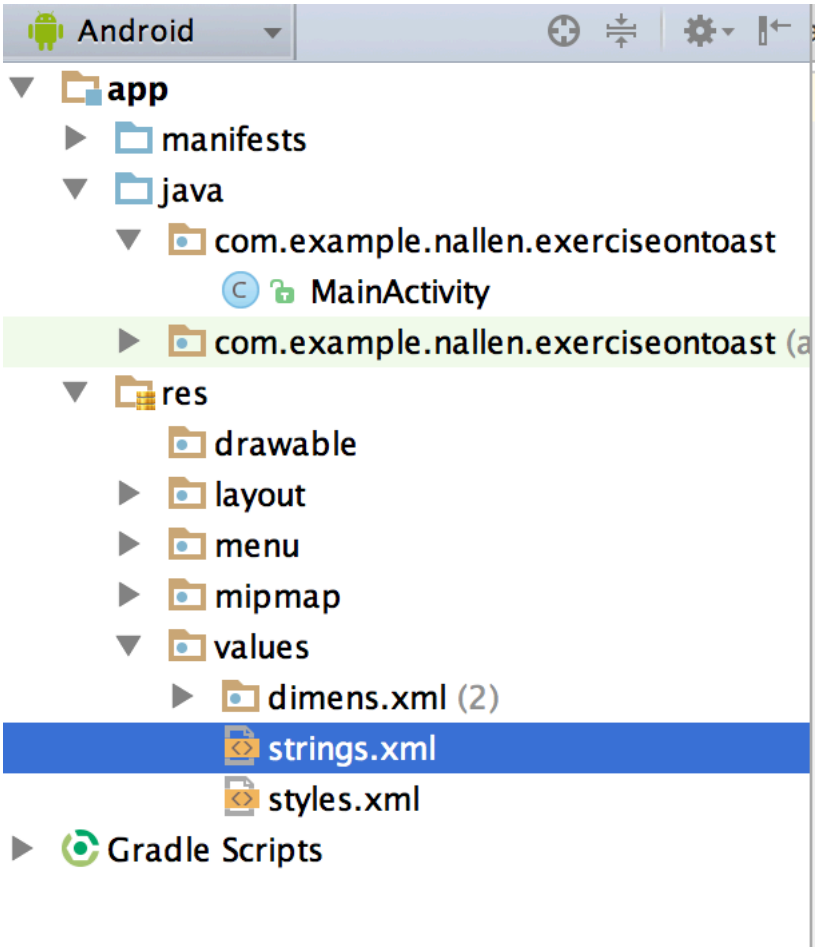

FIGURE 5 - NAVIGATING TO STRINGS.XML

When you open the `strings.xml` file ensure that the contents are as follows:

```
<resources>
    <string name="app_name">AndroidToastExample</string>
    <string name="hello_world">Hello world!</string>
    <string name="action_settings">Settings</string>
    <string name="menu_settings">Settings</string>
    <string name="button_label">Show message</string>
    <string name="image_content">image</string>
</resources>
```

Your canvas should look like figure 6.

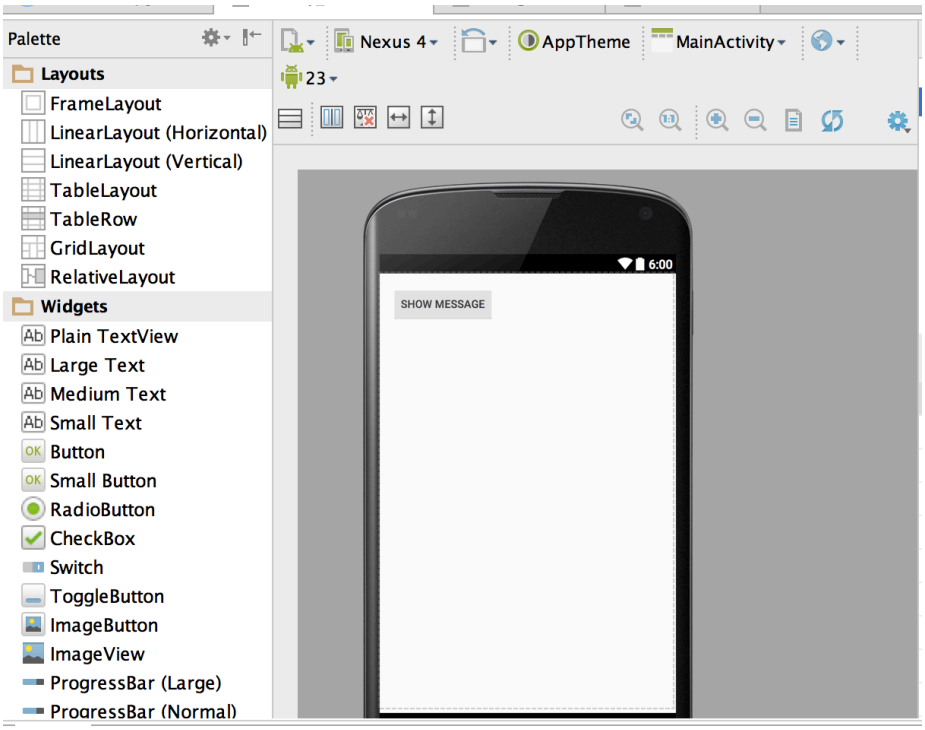

FIGURE 6 COMPLETED GUI

Step 3 Open `MainActivity.java`

Go to the java file that contains the code of the activity you've just created:

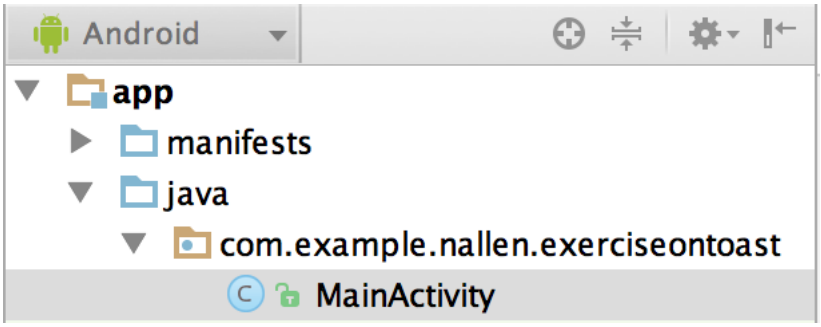

FIGURE 7 NAVIGATING TO MAINACTIVITY.JAVA

Ensure that this java file contains the following code:

```
import android.app.Activity;
import android.os.Bundle;
import android.view.Gravity;
import android.view.LayoutInflater;
import android.view.View;
import android.view.View.OnClickListener;
import android.view.ViewGroup;
import android.widget.Button;
import android.widget.TextView;
import android.widget.Toast;


public class MainActivity extends Activity {


    private Button button;

    public void onCreate(Bundle savedInstanceState) {

        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        button = (Button) findViewById(R.id.mainbutton);

        button.setOnClickListener(new OnClickListener() {

            @Override
            public void onClick(View arg0) {

                // code for first activity
                Toast.makeText(getApplicationContext(), "This is an Android Toast
                Message", Toast.LENGTH_LONG).show();


            }
        });
    }
}
```

Step 4 Run the application

When run, the application will be displayed as per figure 8. When the button is press the output will be as per figure 9. Remember that the layout of the main screen is described by `activity_main.xml`:

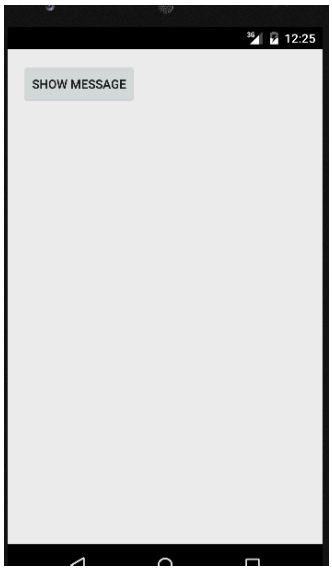

FIGURE 8 - RUNNING THE APP

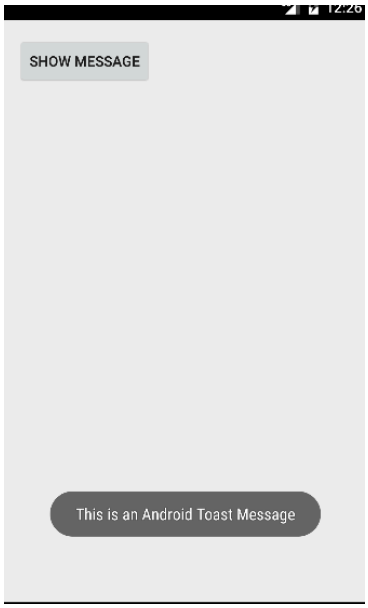

FIGURE 9 - OUTPUT ON BUTTON PRESS

Exercise 3 Creating a customized `Toast` message

There may be instances where you may want to create your own customized `Toast` message. For example you may want the `Toast` message to contain an image. To achieve this anew `layout xml` file need to be created. The `Toast` message will be created according to the layout specified in this xml file.

Step1 Create a new layout file

Within the `project folder` navigate to the `layout folder`. Right click on this folder and select `New` and then choose `Layout resource file` as shown in figure 10.

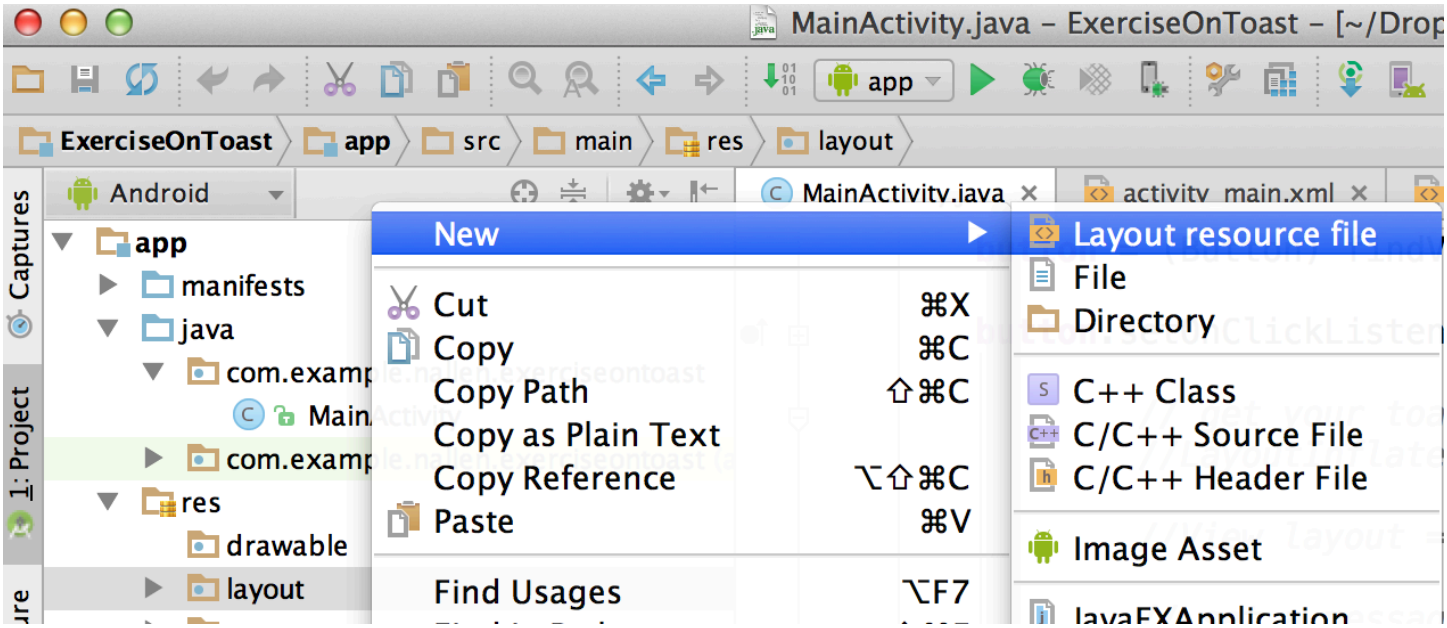

FIGURE 10 CREATING A NEW LAYOUT XML

Specify the name of the file and the Layout type as per figure 11 and click Finish:

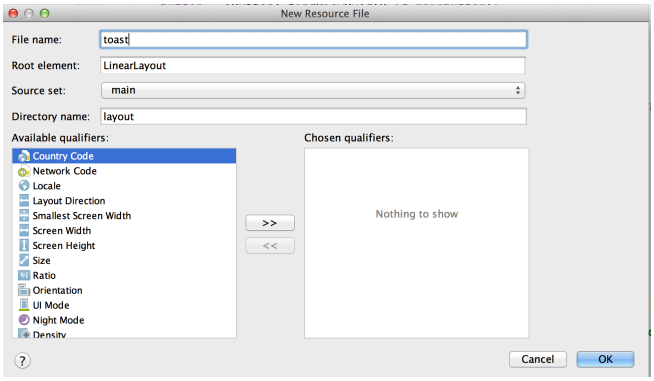

FIGURE 11 - ADDING NEW LAYOUT FILE

As you will see in the Package Explorer the new `/res/layout/toast.xml` file has been created as shown in figure 12.

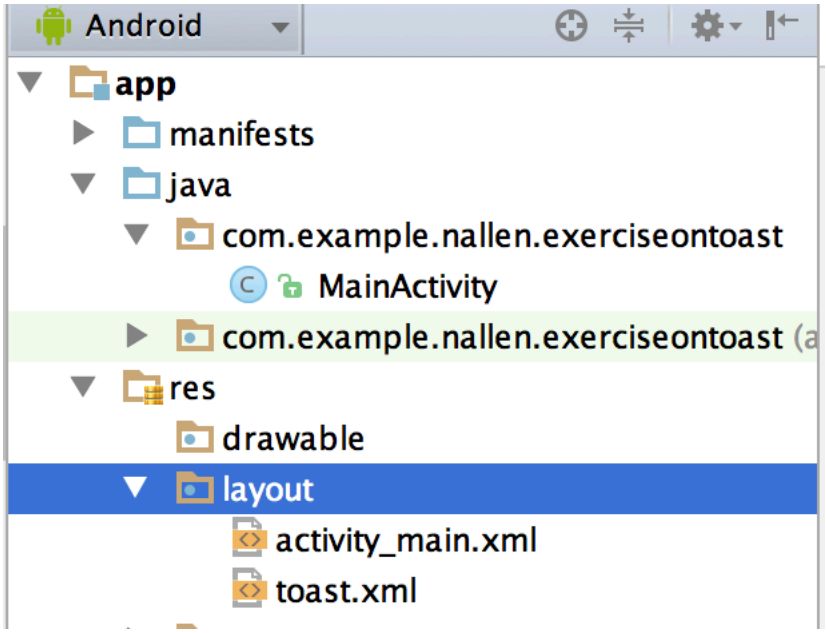

FIGURE 12 NEW LAYOUT FILE SUCCESSFULLY ADDED

Open `toast.xml` and update the XML so that it is as follows:

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/toast_layout_id"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:background="#000"
    android:orientation="horizontal"
    android:padding="5dp">

    <ImageView
        android:id="@+id/image"
        android:layout_width="wrap_content"
        android:layout_height="fill_parent"
        android:layout_marginRight="5dp"
        android:contentDescription="@string/image_content"
        android:src="@mipmap/ic_launcher" />

    <TextView
        android:id="@+id/text"
        android:layout_width="wrap_content"
        android:layout_height="fill_parent"
        android:textColor="#FFF" />

</LinearLayout>
```

Next go to the Design View of the XML and it should look like figure 13.

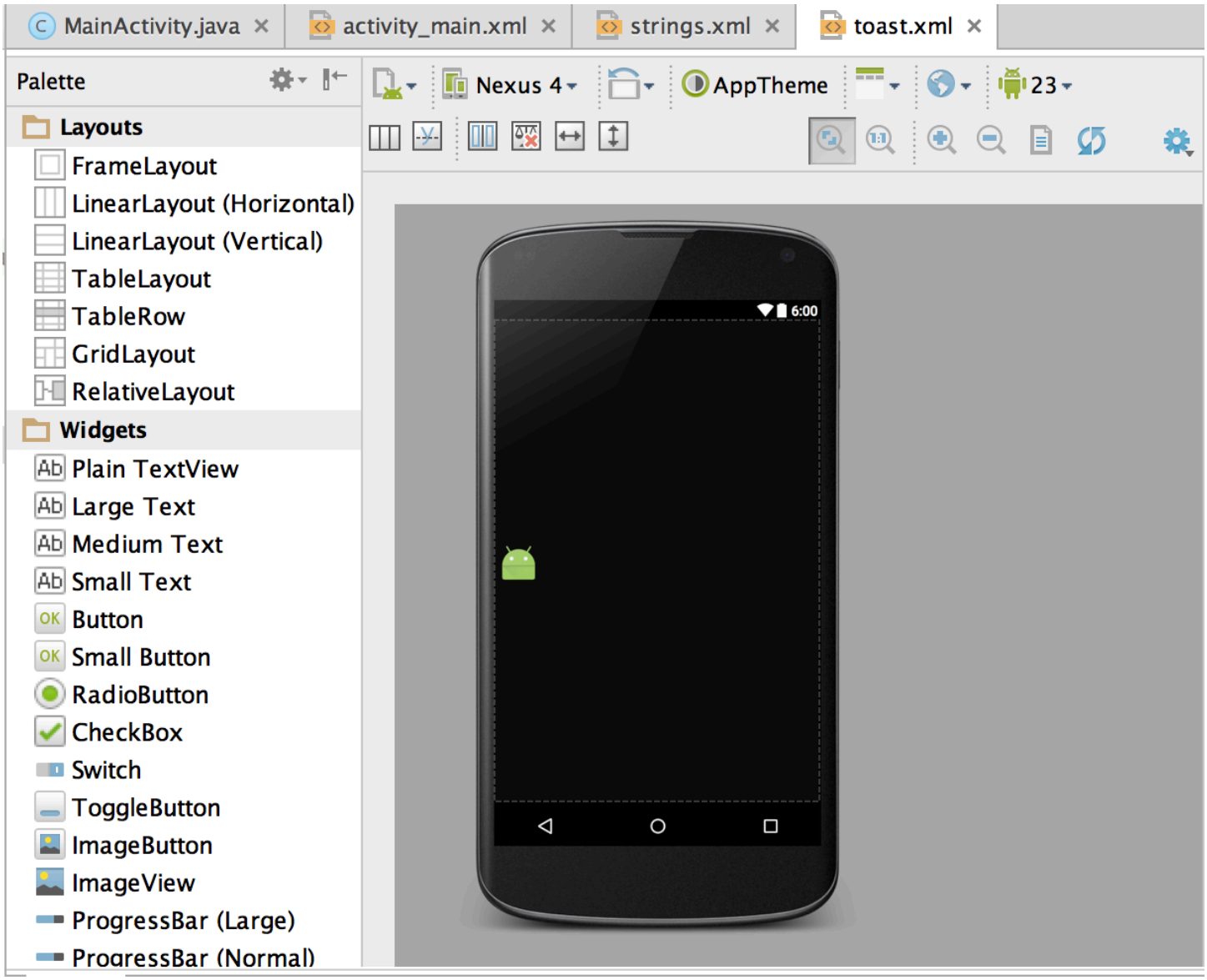

FIGURE 13 - TOAST.XML

This is the layout that will be used for the `Custom Toast Message`.

Step 2 Open `MainActivity.java`

The code of this tutorial is pretty much self-explanatory. The interesting part is how to tell the `Toast` component to use the new xml file as its layout description. Open `MainActivity.java` file as shown in the previous steps, and change the code to this:

```
import android.app.Activity;
import android.os.Bundle;
import android.view.Gravity;
import android.view.LayoutInflater;
import android.view.View;
import android.view.View.OnClickListener;
import android.view.ViewGroup;
import android.widget.Button;
import android.widget.TextView;
import android.widget.Toast;


public class MainActivity extends Activity {

    private Button button;

    public void onCreate(Bundle savedInstanceState) {

        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        button = (Button) findViewById(R.id.mainbutton);
        button.setOnClickListener(new OnClickListener() {

            @Override
            public void onClick(View arg0) {
                // get your toast.xml layout
                LayoutInflater inflater = getLayoutInflater();

                View layout = inflater.inflate(R.layout.toast, (ViewGroup)
                findViewById(R.id.toast_layout_id));

                // set a message

                TextView text = (TextView) layout.findViewById(R.id.text);
                text.setText("This is a Custom Toast Message");

                // Toast configuration
                Toast toast = new Toast(getApplicationContext());
                toast.setGravity(Gravity.CENTER_VERTICAL, 0, 0);
                toast.setDuration(Toast.LENGTH_LONG);
                toast.setView(layout);
                toast.show();

            }
        });
    }
}
```

The `LayoutInflater`

The `LayoutInflater` is used to create a new `View` that will use the `toast.xml` file as the layout description. When `toast.setView(layout)` is executed, the `Toast` is configured to use the newly created `View` as its User Interface.

Step 3 Run the updated application

When the application is run, `activity_main.xml` is used to inflate the GUI (figure 14). When the user presses the button, a custom `Toast` message will appear using the `toast.xml` file as the GUI.

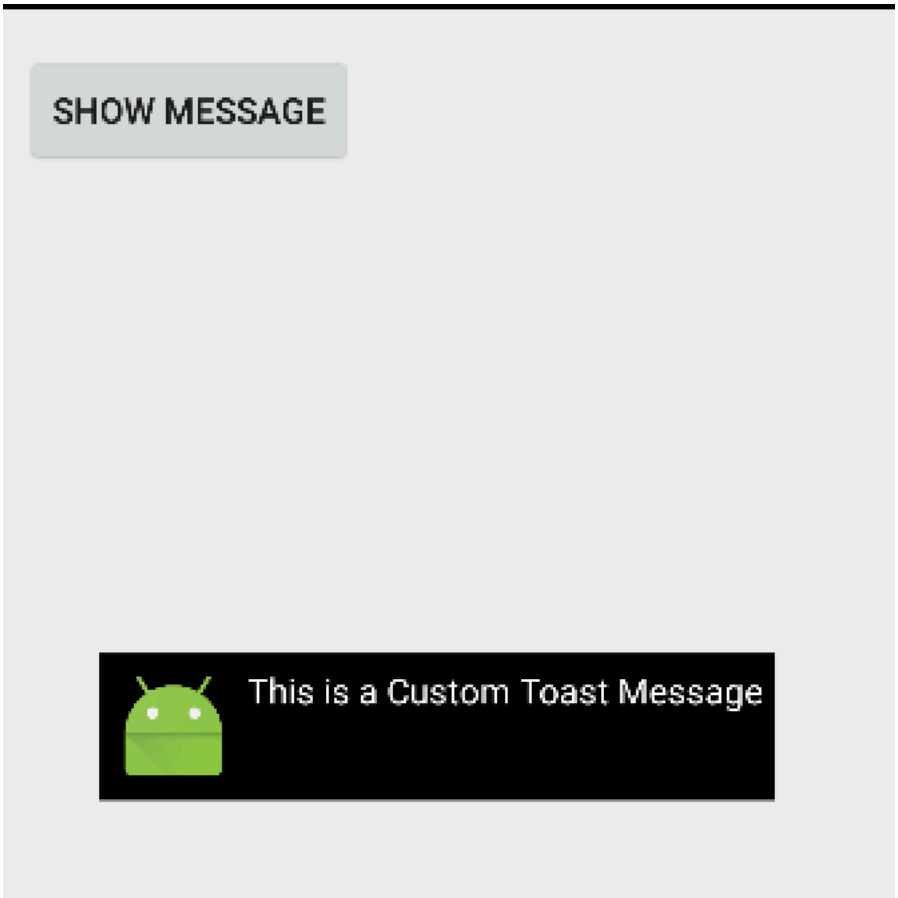

FIGURE 14 - CUSTOM TOAST

Exercise 4 Picking up information from an `EditText`

In this exercise you will learn how to pick information up from an `EditText`. Setup a new project and called it `ToastUserName`.

Step 1 Setup the XML

Navigate to activity_main.xml as show in previous steps and set up the XML as follows:

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:orientation="vertical"
    tools:context=".MainActivity">

    <EditText
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:id="@+id/firstNameEditText"
        android:hint="@string/FirstNameHint"
        android:inputType="text"/>

    <EditText
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:id="@+id/SurameEditText"
        android:hint="@string/SurnameHint"
        android:inputType="text"/>

    <Button
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="@string/ButtontText"
        android:id="@+id/button1"/>

</LinearLayout>
```

When complete, the GUI should look like figure 15.

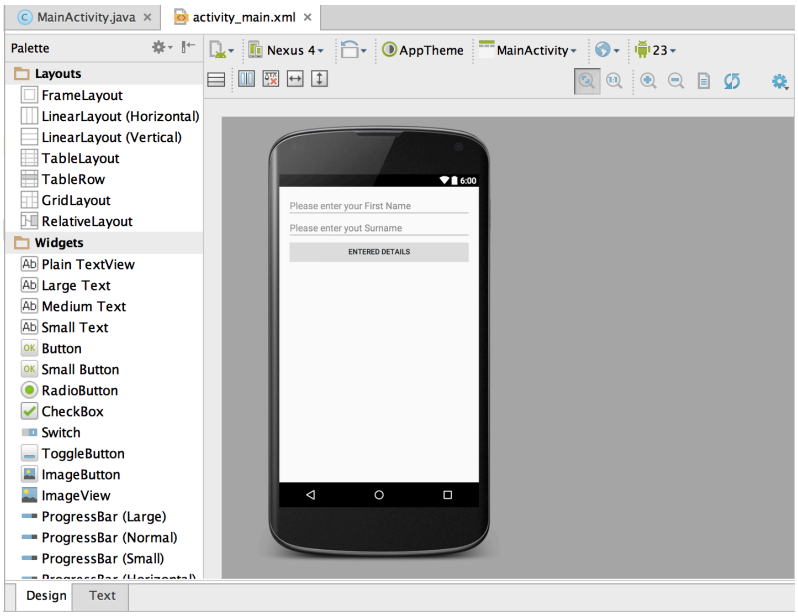

FIGURE 15 - COMPLETED GUI

Step 2 Update `strings.xml`

Navigate to the `strings.xml` file as shown in previous steps and ensure that the following resources have been added:

```
<resources>
    <string name="app_name">ToastUserName</string>
    <string name="hello_world">Hello world!</string>
    <string name="action_settings">Settings</string>
    <string name="FirstNameHint">Please enter your First Name</string>
    <string name="SurnameHint">Please enter yout Surname</string>
    <string name="ButtontText">Entered Details</string>
</resources>
```

Step 3 Add the Java code

Navigate to `MainActivity.java` and add the following code:

```
import android.app.Activity;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.Toast;

public class MainActivity extends Activity {


    @Override
    protected void onCreate(Bundle savedInstanceState) {

        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        Button b1 = (Button)findViewById(R.id.button1);

        b1.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                EditText firstname = (EditText)findViewById(R.id.firstNameEditText);
                EditText surname = (EditText) findViewById(R.id.SurameEditText);

                String output = firstname.getText() + " " + surname.getText();

                Toast.makeText(getApplicationContext(), output,
                Toast.LENGTH_SHORT).show();
            }
        });
    }
}
```

Step 4 Run your application

Run your application and your output should look like figures 16 and 17 respectively.

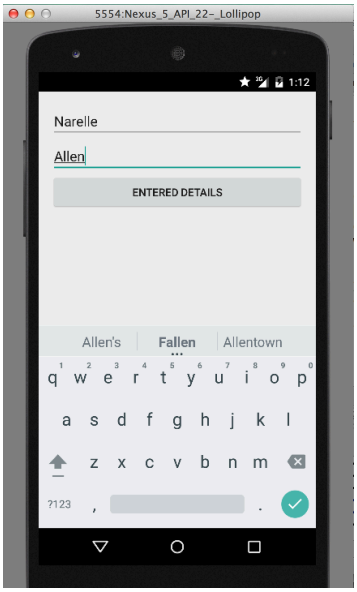

FIGURE 16 ENTERING DETAILS

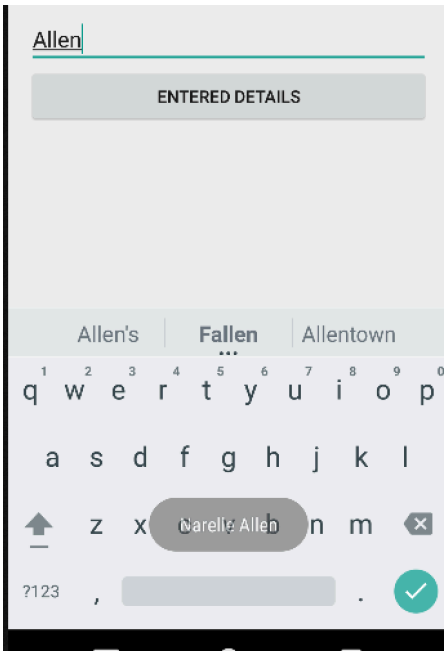

FIGURE 17 DISPLAYING THE INFORMATION