Lab 2: Temperature Converter App

Ref: http://www.vogella.com/tutorials/Android/article.html#tutorialtemperature (section 12)

Learning outcomes:

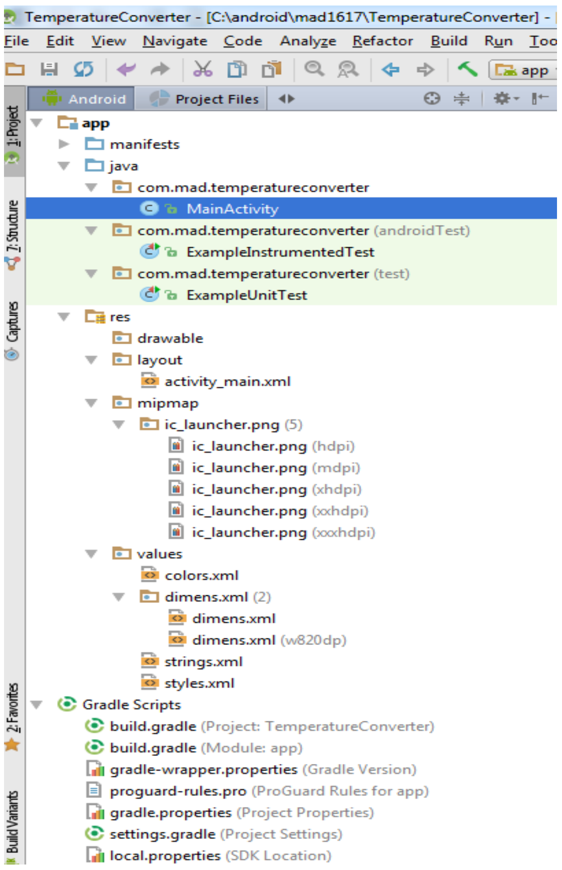
- Be able to use Android Studio to develop a basic GUI driven Android application to convert temperatures
- Know how to use the Interactive Layout Builder
- Understand basic event handling
- Know how to obtain user input data

Task:

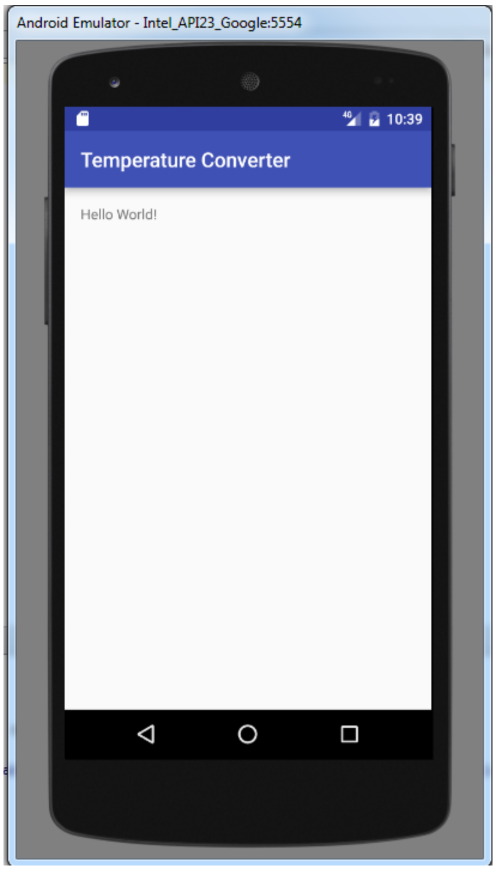
1. Create a new Android Project:

Property	Value
Application Name	Temperature Converter
Package name	com.mad.temperatureconverter
Form Factor	Phone and Tablet
API (Minimum, Target, Compile with)	API 15, Android 4.0.3(Icecream Sandwich), latest, latest
Template	Empty Activity
Activity	MainActivity (default)
Layout	activity_main (default)

2. Review the generated code and run on an emulator


TemperatureConverter - [C:\android\mad1617\TemperatureConverter] -
File Edit View Navigate Code Analyze Refactor Build Run Too
app
Android
Project Files
1: Project
7: Structure
Captures
2: Favorites
Build Variants

```
app
  manifests
  java
    com.mad.temperatureconverter
      MainActivity
    com.mad.temperatureconverter (androidTest)
      ExampleInstrumentedTest
    com.mad.temperatureconverter (test)
      ExampleUnitTest
  res
    drawable
    layout
      activity_main.xml
    mipmap
      ic_launcher.png (5)
        ic_launcher.png (hdpi)
        ic_launcher.png (mdpi)
        ic_launcher.png (xhdpi)
        ic_launcher.png (xxhdpi)
        ic_launcher.png (xxxhdpi)
    values
      colors.xml
      dimens.xml (2)
        dimens.xml
        dimens.xml (w820dp)
      strings.xml
      styles.xml
Gradle Scripts
  build.gradle (Project: TemperatureConverter)
  build.gradle (Module: app)
  gradle-wrapper.properties (Gradle Version)
  proguard-rules.pro (ProGuard Rules for app)
  gradle.properties (Project Properties)
  settings.gradle (Project Settings)
  local.properties (SDK Location)
```



3. **Create Attributes**
   Select the *res/values/strings.xml* file to open the editor for this file. Add the Color and String definitions to the file as shown below

Type	Name	Value
Color	myColor	#F5F5F5
String	celsius	to Celsius
String	fahrenheit	to Fahrenheit
String	calc	Calculate

The resultant strings.xml:

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

    <string name="app_name">Temperature Converter</string>
    <string name="action_settings">Settings</string>
    <string name="hello_world">Hello world!</string>
    <color name="myColor">#F5F5F5</color>
    <string name="celsius">to Celsius</string>
    <string name="fahrenheit">to Fahrenheit</string>
    <string name="calc">Calculate</string>

</resources>
```

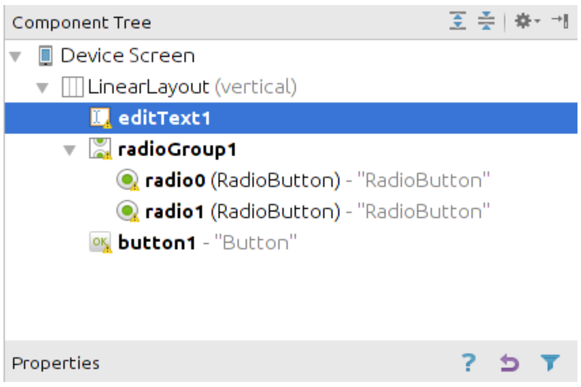
4. **Create the Layout**

Select the *res/layout/activity_main.xml* file and open the associated Android editor via a double-click on the file.

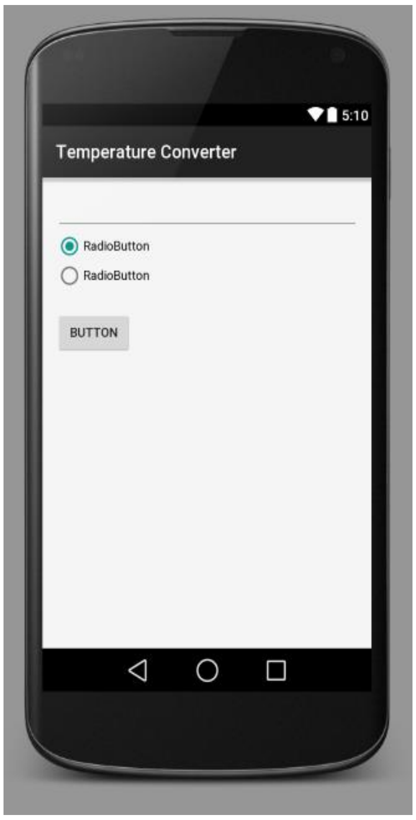
Remove any existing view from your layout, either directly from the XML source or via the graphical editor.

Afterwards add a LinearLayout, another LinearLayout with one ViewText, and EditText as children. Afterwards a RadioGroup with two radio buttons and a Button to your layout. Do this either directly in the XML file or via the graphical editor. A simple way of organizing the components is to drag and drop them onto the *Component Tree* view.

The result should look like the following screenshots. The first one shows the component view the second one the preview.


Component Tree
- Device Screen
  - LinearLayout (vertical)
    - editText1
    - radioGroup1
      - radio0 (RadioButton) - "RadioButton"
      - radio1 (RadioButton) - "RadioButton"
    - button1 - "Button"

Properties



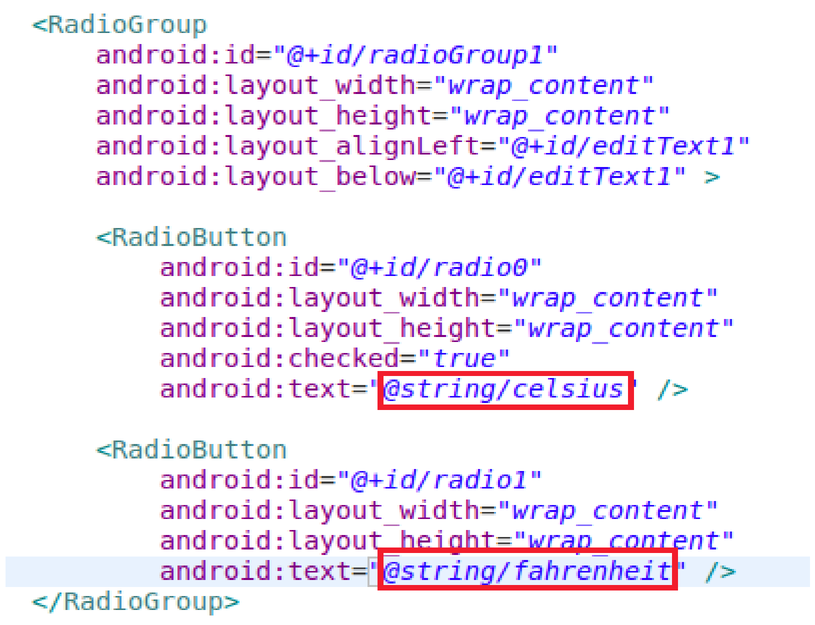
Switch to the XML tab of your layout file and verify that the file looks similar to the following listing. The Android tools team changes the generated code from time to time, so your XML might look slightly different.

```
<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:orientation="vertical"
  android:paddingBottom="@dimen/activity_vertical_margin"
  android:paddingLeft="@dimen/activity_horizontal_margin"
  android:paddingRight="@dimen/activity_horizontal_margin"
  android:paddingTop="@dimen/activity_vertical_margin"
  tools:context=".MainActivity"
  android:background="@color/myColor">

  <EditText
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:id="@+id/editText1" />

  <RadioGroup
    android:id="@+id/radioGroup1"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_alignStart="@+id/editText1"
    android:layout_below="@+id/editText1">
    <RadioButton
      android:id="@+id/radio0"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:checked="true"
      android:text="RadioButton" />

    <RadioButton
      android:id="@+id/radio1"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:text="RadioButton" />
  </RadioGroup>

  <Button
    android:id="@+id/button1"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_alignStart="@+id/radioGroup1"
    android:layout_below="@+id/radioGroup1"
    android:layout_marginTop="22dp"
    android:text="Button" />
</LinearLayout>
```

Note: You may see some warning messages. You'll fix these in the following section of this exercise.

5. **Edit view properties**

Switch to the XML representation of the file and assign the `@string/celsius` value to the *android:text* property of the first radio button. Assign the *fahrenheit* string attribute to the *text* property of the second radio button.



```
<RadioGroup
    android:id="@+id/radioGroup1"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_alignLeft="@+id/editText1"
    android:layout_below="@+id/editText1" >

    <RadioButton
        android:id="@+id/radio0"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:checked="true"
        android:text="@string/celsius" />

    <RadioButton
        android:id="@+id/radio1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/fahrenheit" />
</RadioGroup>
```

Ensure that the *Checked* property is set to `true` for the first `RadioButton`.

Assign *@string/calc* to the text property of your button and assign the value *onClick* to the *OnClick* property.

Set the *inputType* property to *numberSigned* and *numberDecimal* on the `EditText`. As an example you can use the last line in the following XML snippet. Also change its ID to "inputValue".

```
<EditText
  android:id="@+id/inputValue"
  android:layout_width="match_parent"
  android:layout_height="wrap_content"
  android:layout_alignParentEnd="true"
  android:layout_below="@+id/textView"
  android:ems="10"
  android:inputType="numberSigned|numberDecimal" />
```

All your user interface components are contained in a layout. Assign the background color to this `Layout`. Select *Color* and then select *myColor* in the dialog. As an example you can use the last line in the following XML snippet.

```
<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:orientation="vertical"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    tools:context=".MainActivity"
    android:background="@color/myColor">
```

Afterwards the background should change to the *whitesmoke* color. It might be difficult to see the difference.

Switch to the *activity_main.xml* tab and verify that the XML is correct.

```
<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:orientation="vertical"
android:paddingBottom="@dimen/activity_vertical_margin"
android:paddingLeft="@dimen/activity_horizontal_margin"
android:paddingRight="@dimen/activity_horizontal_margin"
android:paddingTop="@dimen/activity_vertical_margin"
tools:context=".MainActivity"
android:background="@color/myColor">
<EditText
   android:layout_width="match_parent"
   android:layout_height="wrap_content"
   android:id="@+id/inputValue"
   android:inputType="numberSigned|numberDecimal"/>
<RadioGroup
   android:id="@+id/radioGroup1"
   android:layout_width="wrap_content"
   android:layout_height="wrap_content"
   android:layout_alignStart="@+id/editText1"
   android:layout_below="@+id/editText1">
   <RadioButton
      android:id="@+id/radio0"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:checked="true"
      android:text="@string/celsius" />

   <RadioButton
      android:id="@+id/radio1"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:text="@string/fahrenheit" />
</RadioGroup>

<Button
   android:id="@+id/button1"
   android:layout_width="wrap_content"
   android:layout_height="wrap_content"
   android:layout_alignStart="@+id/radioGroup1"
   android:layout_below="@+id/radioGroup1"
```

```
        android:layout_marginTop="22dp"
        android:text="@string/calc"
        android:onClick="onClick"/>
</LinearLayout>
```

6. **Create utility class**

Create the following utility class to convert from celsius to fahrenheit and vice versa.

```
public class ConverterUtil {  // converts to celsius
  public static float convertFahrenheitToCelsius(float fahrenheit) {
    return ((fahrenheit - 32) * 5 / 9);
  }
  // converts to fahrenheit
  public static float convertCelsiusToFahrenheit(float celsius) {
    return ((celsius * 9) / 5) + 32;
  }
}
```

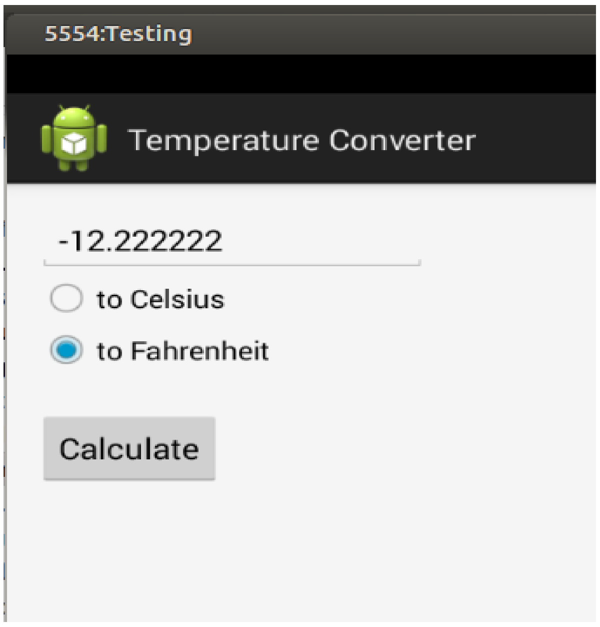
7. **Change the activity code**

The Android project wizard created the corresponding `MainActivity` class for your activity code. Adjust this class so that it is similar to the following code.

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

public class MainActivity extends Activity {
  private EditText text;

  @Override
  public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
    text = (EditText) findViewById(R.id.inputValue);

  }
  // this method is called at button click because we assigned the name to the
  // "OnClick" property of the button
  public void onClick(View view) {
    switch (view.getId()) {
    case R.id.button1:
      RadioButton celsiusButton = (RadioButton) findViewById(R.id.radio0);
      RadioButton fahrenheitButton = (RadioButton) findViewById(R.id.radio1);
      if (text.getText().length() == 0) {
        Toast.makeText(this, "Please enter a valid number",
            Toast.LENGTH_LONG).show();
        return;
      }
      float inputValue = Float.parseFloat(text.getText().toString());
      if (celsiusButton.isChecked()) {
        text.setText(String
            .valueOf(ConverterUtil.convertFahrenheitToCelsius(inputValue)));
        celsiusButton.setChecked(false);
        fahrenheitButton.setChecked(true);
      } else {
        text.setText(String
            .valueOf(ConverterUtil.convertCelsiusToFahrenheit(inputValue)));
        fahrenheitButton.setChecked(false);
        celsiusButton.setChecked(true);
      }
      break;
    }
  }
}
```

8. **Start the application**

Start your Android application and type in a number, select your conversion and press the button. The result should be displayed and the other option should get selected.



Deliverable: Upload your compressed project folder to Moodle