ATLS 4120/5120: Mobile Application Development
Week 12: Android UI

Continue with Feelings app

EditText
Add a horizontal linear layout under the textView

```
<LinearLayout
   android:orientation="horizontal"
   android:layout_width="match_parent"
   android:layout_height="wrap_content"
   android:layout_below="@+id/textView"
   android:id="@+id/linear1">
</LinearLayout>
```

Note that you want the height to be wrap_content so it's only as tall as needed. Match_parent would take up the whole view.
In the linear layout add a EditText
We're using a linear layout because we want these two controls to be next to each other.
In strings.xml add <**string name="name_edit"**>Name</**string**>
In activity_main.xml update the editView.

```
<EditText
   android:layout_width="wrap_content"
   android:layout_height="wrap_content"
   android:id="@+id/name_editText"
   android:hint="@string/name_edit"
  android:layout_weight="1" />
```

Now let's update MainActivity.java so our message is personalized with our name. Update findMood()
EditText name = (EditText) findViewById(R.id.***name_editText***);
String nameValue = name.getText().toString();

feeling.setText(nameValue + **" is in a "** + moodValue + **" mood"**);
We create a name object so we have a reference to our EditText.
Then we create a string and use getText() to get the text in the EditText and toString() to cast it to a String so we can use it in our TextView.

Run your app.

In a larger app where you need access to the UI components throughout the class you can make the global by adding TextView feeling; right under the class definition.

Toggle Button
Add a toggle button next to the name editText
You can double click in the design view to get the most common properties.
Add the strings
<**string name="toggle_on"**>positive</**string**>
<**string name="toggle_off"**>negative</**string**>

Update the xml

```
<ToggleButton
   android:layout_width="wrap_content"
   android:layout_height="wrap_content"
   android:textOn="@string/toggle_on"
   android:textOff="@string/toggle_off"
   android:id="@+id/energy_toggle"/>
```

[cmd B will take you to where this is defined in the strings.xml file]
The toggle is on the right and the name takes up most of the line because editText has
android:layout_weight="1"
See if you can figure out how to move the edittext and the toggle button down a little bit.

Add logic to MainActivity.java

```
ToggleButton toggle = (ToggleButton) findViewById(R.id.energy_toggle);
boolean energy = toggle.isChecked();
String energyString;
if(energy) {
   energyString = "positive";
}
   else {
   energyString="negative";
}
```

Update textView

```
feeling.setText(nameValue + " is a " + energyString + " person");
```

Radio Buttons

Add another linear layout for the next horizontal row

```
<LinearLayout
   android:orientation="horizontal"
   android:layout_width="match_parent"
   android:layout_height="wrap_content"
   android:layout_below="@+id/linear1"
   android:id="@+id/linear2"
   android:layout_marginTop="5dp">
```

In it put a textView that says yoga and a radio group with 3 radio buttons for the types of yoga.
In strings.xml add

```
<string name="yoga_text">Yoga</string>
<string name="yoga1_radio">Yin</string>
<string name="yoga2_radio">Bikram</string>
<string name="yoga3_radio">Hatha</string>
```

Then update activity_main.xml update

```
<TextView
   android:layout_width="wrap_content"
   android:layout_height="wrap_content"
   android:text="@string/yoga_text"/>
```

```
<RadioGroup
   android:layout_width="wrap_content"
   android:layout_height="wrap_content"
   android:id="@+id/yoga_type"
   android:orientation="horizontal"
   android:layout_weight="1">

 <RadioButton
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="@string/yoga1_radio"
    android:id="@+id/radioButton1"/>

 <RadioButton
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="@string/yoga2_radio"
    android:id="@+id/radioButton2"
    android:layout_weight="1"/>

 <RadioButton
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="@string/yoga3_radio"
    android:id="@+id/radioButton3"/>
</RadioGroup>
</LinearLayout>
```

To get them all evenly spaced in the row add for each **android:layout_weight="1**

Add logic to MainActivity.java

```
RadioGroup yoga = (RadioGroup) findViewById(R.id.yoga_type);
String yogatype;
int yoga_id = yoga.getCheckedRadioButtonId();
switch(yoga_id){
   case -1:
      yogatype="no";
      break;
   case R.id.radioButton1:
      yogatype="Yin";
      break;
   case R.id.radioButton2:
      yogatype="Bikram";
      break;
   case R.id.radioButton3:
      yogatype="Hatha";
      break;
```

```
    default:
        yogatype="no";
}
```

Update textView
feeling.setText(nameValue + " is a " + energyString + " person" + " that does " + yogatype + " yoga");

Check boxes
For our 4 checkoxes we need 2 rows. So add 2 linear layouts with 2 checkboxes in each.
Add string resources

```
<string name="enlightened_check">enlightened</string>
<string name="conservative_check">conservative</string>
<string name="sarcastic_check">sarcastic</string>
<string name="secretive_check">secretive</string>
```

Update activity_main.xml

```
<LinearLayout
    android:orientation="horizontal"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_marginTop="5dp"
    android:layout_below="@+id/linear2"
    android:id="@+id/linear3">

<CheckBox
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="@string/sarcastic_check"
    android:id="@+id/checkBox1"
    android:layout_weight="1" />

<CheckBox
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="@string/conservative_check"
    android:id="@+id/checkBox2"
    android:layout_weight="1" />
</LinearLayout>

<LinearLayout
    android:orientation="horizontal"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_below="@+id/linear3"
    android:id="@+id/linear4">

<CheckBox
    android:layout_width="wrap_content"
```

```
    android:layout_height="wrap_content"
    android:text="@string/secretive_check"
    android:id="@+id/checkBox3"
    android:layout_weight="1" />

    <CheckBox
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/enlightened_check"
        android:id="@+id/checkBox4"
        android:layout_weight="1" />
</LinearLayout>
```

We are again using **android:layout_weight="1"** to get them evenly spaced

Add logic to MainActivity.java

```
String checkbox_string = "";
CheckBox check1 = (CheckBox) findViewById(R.id.checkBox1);
boolean checked1 = check1.isChecked();
if(checked1){
    checkbox_string += " sarcastic";
}

CheckBox check2 = (CheckBox) findViewById(R.id.checkBox2);
boolean checked2 = check2.isChecked();
if(checked2){
    checkbox_string += " conservative";
}

CheckBox check3 = (CheckBox) findViewById(R.id.checkBox3);
boolean checked3 = check3.isChecked();
if(checked3){
    checkbox_string += " secretive";
}

CheckBox check4 = (CheckBox) findViewById(R.id.checkBox4);
boolean checked4 = check4.isChecked();
if(checked4){
    checkbox_string += " enlightened";
}
```

Update textView

```
feeling.setText(nameValue + " is a " + energyString + checkbox_string + " person" + " that does " +
yogatype + " yoga");
```

Switch

The spinner is in a linear layout. Add a switch next to it.

Add the string resource <**string name="meditate_switch"**>Meditate</**string**>

Update the xml.

```
<LinearLayout
   android:orientation="horizontal"
   android:layout_width="match_parent"
   android:layout_height="wrap_content"
   android:layout_below="@+id/linear4"
   android:id="@+id/linear5">

   <Spinner
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:id="@+id/spinner"
      android:entries="@array/moods" />

   <Switch
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:paddingLeft="110dp"
      android:text="@string/meditate_switch"
      android:id="@+id/switch1" />
</LinearLayout>
```

Give the spinner **android:layout_weight="1"** and the switch **android:paddingLeft=110dp"** to space them in the row.

Add logic to MainActivity.java

```
String meditate_string = "";
Switch meditate_switch = (Switch) findViewById(R.id.switch1);
boolean meditate = meditate_switch.isChecked();
if (meditate){
   meditate_string = " and meditates";
}
```

Update textView

```
feeling.setText(nameValue + " is a " + energyString + checkbox_string + " person" + " that does " + yogatype + " yoga" + meditate_string);
```

ImageView

Copy and paste your images into the drawables folder.
Add an imageView after the button and before the textView.
Add the string resource <**string name="mood_image"**>Mood Image</**string**>
Update the xml

```
<ImageView
   android:layout_width="100dp"
   android:layout_height="100dp"
   android:id="@+id/imageView"
   android:layout_below="@+id/button"
   android:layout_centerHorizontal="true"
```

```
   android:layout_marginTop="5dp"
   android:contentDescription="@string/mood_image"/>
```

The textView should be the last item in the relative layout:

```
<TextView
   android:layout_width="wrap_content"
   android:layout_height="wrap_content"
   android:text="@string/feeling"
   android:id="@+id/feelingText"
   android:layout_below="@+id/imageView"
   android:layout_centerVertical="true"
   android:layout_centerHorizontal="true"
   android:layout_marginTop="5dp"
   android:textSize="20sp"/>
```

Add logic to MainActivity.java

```
ImageView emotion = (ImageView) findViewById(R.id.imageView);
int image;
if (moodValue.equals("happy")) {
   image = R.drawable.emotion_happy;
} else if (moodValue.equals("sad")) {
   image = R.drawable.emotion_sad;
} else if (moodValue.equals("confused")) {
   image = R.drawable.emotion_confused;
} else if (moodValue.equals("angry")) {
   image = R.drawable.emotion_angry;
} else image = R.drawable.emotion_happy;
emotion.setImageResource(image);
```

App icons

I used Android Asset Studio to create all my launcher images.
http://romannurik.github.io/AndroidAssetStudio/index.html
Go to your apps app/src/main/res folder and replace all the mipmap folders with the new ones generated.
Exit Android Studio and start it to see your new launcher icon.
OR
File | New | Image Asset
Browse to the highest res launcher image (192 x 192)
Next
When you chose Finish Android Studio will generate all the other launcher images for you overwriting the default images.

App name

If you look in the Android_manifest.xml file you will see an application label that holds the string app_name. To change this go into the strings.xml file and change that resource to change your app name.

```
<string name="app_name">Karma</string>
```