ITMD 455/555 *Intelligent Device Applications* Lab 2

#### Temperature Converter (part deux) App- 50 points

**Introduction**. This lab continues with lab 1 logic but throws in a few more goodies namely a seekBar, checkbox, listview with an arrayAdapter and a Viewstub.

The code is well laid out replete with **comments**. Study over the code syntax, the logic (esp. the objects used), the import statements of the Java and XML files VERY carefully, line by line if necessary as that’s one way to learn big time!

**STEP 1 Creating a New Android Project**

Create a new project as an Empty Views Activity. When configuring your project, name it **TempConverter2** (Package name will be **com.example.tempconverter2**). Save your project to a desired location. For Language choice choose **Java**. Choose an appropriate API like API **28** (Android 9 Pie). Click Finish when complete.

**STEP 2 Completely override your activity\_main.xml file as follows:**

**\*\*\*Make sure to start your code at the very top line of your file.**

*<?***xml version="1.0" encoding="utf-8"***?>*<**RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"  
 xmlns:tools="http://schemas.android.com/tools"  
 xmlns:app="http://schemas.android.com/apk/res-auto"  
 android:id="@+id/activity\_main"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 app:layout\_constraintBottom\_toBottomOf="parent"  
 app:layout\_constraintLeft\_toLeftOf="parent"  
 app:layout\_constraintRight\_toRightOf="parent"  
 app:layout\_constraintTop\_toTopOf="parent"  
 tools:context="com.example.tempconverter2.MainActivity"**>  
  
 <**TextView android:id="@+id/textview"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"** />  
  
 *<!-- add View as gap for 4 blank lines -->* <**View  
 android:layout\_width="match\_parent"  
 android:layout\_height="60dp"  
 android:id="@+id/view"**>  
 </**View**>  
  
 <**SeekBar android:id="@+id/seekbar"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:max="100"  
 android:minWidth="250dp"  
 android:layout\_below="@+id/view"  
 android:layout\_alignParentLeft="true"  
 android:layout\_alignParentStart="true"** />  
  
 <**CheckBox  
 android:id="@+id/checkBox1"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:text="Show 5 Day forecast"  
 android:layout\_below="@+id/seekbar"  
 android:layout\_alignParentLeft="true"  
 android:layout\_alignParentStart="true"  
 android:layout\_marginTop="49dp"** />  
  
 <**ViewStub  
 android:id="@+id/viewStub1"  
 android:layout="@layout/stubview"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"** />  
  
</**RelativeLayout**>

Never mind the warning in red for the **ViewStub** tag, we’ll fix in the next step by adding the needed layout!

**STEP 3 Add an Android XML file in your layout folder call it stubview. Choose a LinearLayout as your Root element.**

To add in a layout file, right click on your **res/layout** folder and choose

**New > XML > Layout XML File**. Name the file **stubview**. You can put in LinearLayout as your ‘Root Tag’, if it is not given as a choice. Use the following code for the new file:

*<?***xml version="1.0" encoding="utf-8"***?>*<**LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"  
 android:orientation="vertical"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"**>  
  
 <**ListView  
 android:id="@+id/listView"  
 android:layout\_height="wrap\_content"  
 android:layout\_width="match\_parent"**/>  
  
</**LinearLayout**>

You can now observe in your stubview XML file, in Design view, a nice ListView layout destined to allow for a *series* of temperatures to fill each row of the listview. Snapshot follows.

A screenshot of a cell phone

Description automatically generated

**STEP 4 Finally add the following code to your MainActivity java file**

Include the entire code that follows after the following opening line of your class declaration,

up to, but NOT including the closing brace ( **}** ) of your class:

**public class** MainActivity **extends** AppCompatActivity {

SeekBar seekBar; //declare seekbar object  
TextView textView;  
//declare member variables for SeekBar  
int discrete = 0;  
int start = 50;  
int start\_position = 50; //progress tracker  
int temp = 0;  
//declare objects for ViewStub  
ViewStub stub;  
CheckBox checkBox;  
ListView lv; //declare Listview object

@SuppressLint("MissingInflatedId")

@Override  
public void onCreate(Bundle savedInstanceState) {  
 super.onCreate(savedInstanceState);  
 setContentView(R.layout.*activity\_main*);  
  
 //declare viewstub object  
 stub = findViewById(R.id.*viewStub1*);  
 @SuppressWarnings("unused")  
 View inflated = stub.inflate();  
 stub.setVisibility(View.*INVISIBLE*);  
  
 //ViewStub logic  
 checkBox = findViewById(R.id.*checkBox1*);  
  
 //handle checkbox click event  
 checkBox.setOnCheckedChangeListener(new CheckBox.OnCheckedChangeListener() {  
 public void onCheckedChanged(CompoundButton arg0, boolean isChecked) {  
 if (isChecked) {  
 //remove objs from parent view to allow for child view objs  
 checkBox.setVisibility(View.*GONE*);  
 seekBar.setVisibility(View.*GONE*);  
 textView.setVisibility(View.*GONE*);  
 stub.setVisibility(View.*VISIBLE*);  
 }  
 }  
 });  
  
 //seekbar logic  
 textView = findViewById(R.id.*textview*);  
 textView.setText(" Celsius at 0 degrees");  
 //set default view  
 seekBar = findViewById(R.id.*seekbar*);  
 seekBar.setProgress(start\_position);  
  
 //create event handler for SeekBar  
 seekBar.setOnSeekBarChangeListener(new OnSeekBarChangeListener() {  
 @Override  
 public void onStopTrackingTouch(SeekBar seekBar) {  
 if (temp == 0) //for initial view result  
 Toast.*makeText*(getBaseContext(), "Fahrenheit result: " +  
 "32 degrees", Toast.*LENGTH\_SHORT*).show();  
 else  
 Toast.*makeText*(getBaseContext(), "Fahrenheit result: "  
 + String.*valueOf*(discrete) +  
 " degrees", Toast.*LENGTH\_SHORT*).show();  
 }  
  
 @Override  
 public void onStartTrackingTouch(SeekBar seekBar) {  
 }  
  
 @Override  
 public void onProgressChanged(SeekBar seekBar,  
 int progress, boolean fromUser) {  
 // To convert progress passed as discrete (Fahrenheit) value  
 temp = progress - start;  
 discrete = (int) Math.*round*((((temp \* 9.0) /  
 5.0) + 32)); //convert C to F temp  
 textView.setText(" Celsius at " + temp + " degrees");  
 }  
 });  
 //Listview logic  
 String[] wkTemps = new String[]{"1", "-10", "0", "30", "10"};  
  
 lv = findViewById(R.id.*listView*);  
 @SuppressWarnings({"unchecked", "rawtypes"})  
 /\*  
 \* To use a basic ArrayAdapter, you just need to initialize the adapter and  
 \* attach the adapter to the ListView. First, initialize the adapter...:  
 \*  
 \*/  
 ArrayAdapter adapter = new ArrayAdapter(this,  
 android.R.layout.*simple\_list\_item\_1*,  
 android.R.id.*text1*, wkTemps);  
 // Assign adapter to ListView  
 lv.setAdapter(adapter);  
  
}//end onCreate method

}

Add the following import statements as well into your file:

**import** android.view.View;  
**import** android.view.ViewStub;  
**import** android.widget.ArrayAdapter;  
**import** android.widget.CheckBox;  
**import** android.widget.CompoundButton;  
**import** android.widget.ListView;  
**import** android.widget.SeekBar;  
**import** android.widget.SeekBar.OnSeekBarChangeListener;  
**import** android.widget.TextView;  
**import** android.widget.Toast;

**STEP 5 Test drive your app**

Run your app at this point and it should fly. Test first your SeekBar and see if you get the

results your supposed to,via the Toast message. Then *click* on the CheckBox and you should

see your Listview result.

\*Note- If ever your build errors out stating you have an older JDK version such as 1.8, you can choose from your Settings, a gradle build that can be set for version 11 as a possoble workable example. See snapshot that follows.

Graphical user interface, text, application, Teams

Description automatically generated

Other possible issues of late are compiler [dependency](https://stackoverflow.com/questions/73688486/android-studio-error-requires-libraries-and-applications-that-depend-on-it-to-c) ones that state the following

Text

Description automatically generated

To remedy this quickly and easily if necessary, do as stated above and adjust your build gradle settings file for your module. Snapshot that follows shows the file location as highlighted in the navigator area.

Graphical user interface, text, email, website

Description automatically generated

Change your settings for compileSdk and tarketSdk (snapshot below shows my version currently as 32) to 33.

android **{** namespace 'com.example.tempconverter2'  
 compileSdk 32  
  
 defaultConfig **{** applicationId "com.example.tempconverter2"  
 minSdk 26  
 targetSdk 32

Sync your changes to update your gradle build by clicking Sync Now located at the top area of your file tab as shown next.

Graphical user interface, text, application, chat or text message

Description automatically generated

**STEP 6 Modify your MainActivity code as follows:**

1. Search the web for “realistic” Chicago temps for the next 5 days and add them (update) to your string array, **wkTemps**. Likewise include the days of the week (i.e., Monday, Tuesday, etc.) so your temperatures actually reflect temps by day, into your array.

1. Add in a title at the top of your child view stating “5 Day Chicago Forecast”. Include a white font color and a light blue backcolor. See snapshot at end of this doc file as example.
2. In your MainActivity file, replace the Toast pop up message which renders the Fahrenheit temperature currently with a TextView that will serve to display the Fahrenheit equivalent to the chosen Celsius temperature by the user.

**Grads**

1. Include a “Back” Button to your second ‘simulated’ layout– with an appropriate icon for the button, that will ‘simulate’ the user going back to the original view to work the seek bar as depicted when the Activity first started. See snapshot at end of this doc file as example of the additional Back Button.

Include visible views once again once the Back Button has been clicked on, i.e., to show all the elements that were included at app start up for full UX functionality, namely, the textview, seekbar, checkbox and your newly added textview added in Step 6 part 3 above which displays Fahrenheit temperature equivalencies. Again, the visible views should become visible when the user presses your back button.

So basicallly your toggling between certain element ‘views’ or visbilities, upon the back button’s click event (Listview and back button should not be shown upon the action). Set your checkbox back to unchecked as well. Leave elements such as your textviews though in their current state (which should be still fresh with values).

**STEP 7 Submitting your assignment:**

For **full** credit turn in the following files:

1. A pdf file of your snapshots consisting of the following:

* Your Fahrenheit result in your textview with a chosen temperature of 22 degrees Celsius, chosen from the seekbar.
* Your full 5 day forecast by days view that’s rendered also.

**Grads**

Include also a snapshot result of your “Back” button in action that would show the Activity screen once again with it’s full UX functionality (i.e., presented in its lastest state).

1. A pdf file of your source code and XML markup files consisting of the following:

* Your MainActivity file code plus code for both your XML layout files.

1. A zip file of your project files.

Extra Credit (all). Include appropriate images for 5 day forecasts! Sample follows.

Forecast view for 5 days w. images (as extra credit):

