1. In the **WeatherForecast** activity, create a layout that shows:

* An ImageView for the current weather. Set the width and height to be 200dp.
* A TextVew for the current temperature.
* A TextView for the min temperature.
* A TextView for the max temperature.
* A TextView for the UV rating
* A normal progress bar. Set the initial visibility to “invisible”. To make the progress bar a horizontal bar instead of a spinning progress, add this parameter to the ProgressBar tag: ***style="?android:attr/progressBarStyleHorizontal"***

1. the onCreate() function, write code to set the progressBar’s visibility to View.Visible so that the progress bar will show.

1. Create an inner class in WeatherForecast, called ForecastQuery, which extends AsyncTask<String, Integer, String>. The class should have 4 string variables for the UV, min, max, and current temperature. There should also be a Bitmap variable to store the picture for the current weather.

1. We will be using a web server to tell us what the weather is in Delhi.

URL: <http://api.openweathermap.org/data/2.5/weather?q=delhi,in&APPID=7e943c97096a9784391a981c4d878b22&mode=xml&units=metric%22>);

The parameters are **q=**  … the city you want the weather for.

**APPID=** … an API key for measuring how many queries per hour. If we go over the limit for the free service level, then it will stop working.

**Mode=** … Can be JSON, XML, HTTP that gets returned

**Units=metric**  … we want units in Celsius, not Kelvin or Imperial

1. For getting the UV rating, use :

<http://api.openweathermap.org/data/2.5/uvi?appid=7e943c97096a9784391a981c4d878b22&lat=28.644800&lon=-77.216721>

The parameters are lat, and lon for latitude and longitude.

***doInBackground( )***

1. Write the protected String doInBackground(String …args) function for the ForecastQuery class. Look at the example code:

|  |
| --- |
| public String doInBackground(String ... args) |
|  | { |
|  | try { |
|  |  |
|  | //create a URL object of what server to contact: |
|  | URL url = new URL(args[0]); |
|  |  |
|  | //open the connection |
|  | HttpURLConnection urlConnection = (HttpURLConnection) url.openConnection(); |
|  |  |
|  | //wait for data: |
|  | InputStream response = urlConnection.getInputStream(); |
|  |  |
|  |  |
|  |
|  |
|  | XmlPullParserFactory factory = XmlPullParserFactory.newInstance(); |
|  | factory.setNamespaceAware(false); |
|  | XmlPullParser xpp = factory.newPullParser(); |
|  | xpp.setInput( response , "UTF-8"); |
|  |  |
|  |
|  |
|  | String parameter = null; |
|  |  |
|  | int eventType = xpp.getEventType(); //The parser is currently at START\_DOCUMENT |
|  |  |
|  | while(eventType != XmlPullParser.END\_DOCUMENT) |
|  | { |
|  |  |
|  | if(eventType == XmlPullParser.START\_TAG) |
|  | { |
|  | //If you get here, then you are pointing at a start tag |
|  | if(xpp.getName().equals("Weather")) |
|  | { |
|  | //If you get here, then you are pointing to a <Weather> start tag |
|  | String outlook = xpp.getAttributeValue(null, "outlook"); |
|  | String windy = xpp.getAttributeValue(null, "windy"); |
|  | } |
|  |  |
|  | else if(xpp.getName().equals("AMessage")) |
|  | { |
|  | parameter = xpp.getAttributeValue(null, "message"); // this will run for <AMessage message="parameter" > |
|  | } |
|  | else if(xpp.getName().equals("Weather")) |
|  | { |
|  | parameter = xpp.getAttributeValue(null, "outlook"); //this will run for <Weather outlook="parameter" |
|  | parameter = xpp.getAttributeValue(null, "windy"); //this will run for <Weather windy="paramter" > |
|  | } |
|  | else if(xpp.getName().equals("Temperature")) |
|  | { |
|  | xpp.next(); //move the pointer from the opening tag to the TEXT event |
|  | parameter = xpp.getText(); // this will return 20 |
|  | } |
|  | } |
|  | eventType = xpp.next(); //move to the next xml event and store it in a variable |
|  | } |
|  |  |
|  |  |
|  | } |
|  | catch (Exception e) |
|  | { |
|  | Log.e("Error", e.getMessage()); |
|  | } |
|  |  |
|  | return "Done"; |
|  | } |

You should change the starting tags that you are looking for. Each time, you should check if the current tag’s name is “temperature”. If it is, then you must get the value, min, and max parameters and save the text so that you can write the strings in a later step. XMLPullParser has a getAttributeValue(String namespace, String name) function which will give you those values. The namespace should be null, but the attributes you are looking for are “value”, “min”, “max”.

1. For each of the attributes, call publishProgress() with 25, 50, 75 as the parameters to show the progress of retrieving the data. You also need to look for the “weather” tag. The attribute you want to store is “icon”, representing the iconName to show. Run your application to verify that your web connection works properly, and your XMLPullParser is finding the values you need for later.

1. The OpenWeatherMap website also has icons for showing “Cloudy”, or “Sunny”, or “Raining”, etc. The URL you must build is "http://openweathermap.org/img/w/" + iconName + ".png". Here is some code to download an image:

*Bitmap image = null;*

*URL url = new URL(urlString);*

*connection = (HttpURLConnection) url.openConnection();*

*connection.connect();*

*int responseCode = connection.getResponseCode();*

*if (responseCode == 200) {*

*image = BitmapFactory.decodeStream(connection.getInputStream());*

*}*

You must build a Bitmap object, and then save it to the local storage. Once you have downloaded the image, call publishProgress() with 100 as the parameter to show that the progress is completed. Save the Bitmap object to the local application storage with the following code:

*FileOutputStream outputStream = openFileOutput( iconName + ".png", Context.MODE\_PRIVATE);*

*image.compress(Bitmap.CompressFormat.PNG, 80, outputStream);*

*outputStream.flush();*

*outputStream.close();*

1. Now, add code to check if your cloudy, sunny, raining images are already present in the local storage directory:

*public boolean fileExistance(String fname){*

*File file = getBaseContext().getFileStreamPath(fname);*

*return file.exists();   }*

1. If the Image file exists, then you don’t need to re-download it, just read it from your disk:

*FileInputStream fis = null;*

*try {    fis = openFileInput(imagefile);   }*

*catch (FileNotFoundException e) {    e.printStackTrace();  }*

*Bitmap bm = BitmapFactory.decodeStream(fis);*

1. Write Log.i() messages showing which filename you are looking for, and a message saying if you found the image locally, or if you need to download it.

1. For the UV rating, you will need to use the given URL, but use a JSON object.

JSONObject jObject = new JSONObject(result);

 From jObject, you want to retrieve the ***value*** parameters as floats:

float value = jObject.getDouble(“value”);

***onProgressUpdate( )***

1. Write the onProgressUpdate(Integer …value) function so that it sets the visibility of your progressBar to visible. Also, it should set the progress of the progressBar to the variable value[0] being passed in.

***onPostExecute( )***

1. In the onPostExecute function, update the GUI components with the min, max, and current temperature that you have read from the XML. Also update the ImageView with the Bitmap that you downloaded. Also, set the visibility of the progress bar to invisible, using the setVisibility(View.INVISIBLE ) function.
2. Create a new branch called watch. On your new branch, create two directories under /res, called layout-round and layout-notround. Copy your WeatherActivity layout xml from the layout directory into layout-round, and layout-notround. In the layout-round folder, set the layout\_gravity to “center” for each of the textViews so that the text is centered on your watch face.
3. Create both a virtual round and square watch emulator and test that your application still works in both emulators.
4. Create a virtual Android TV emulator and run your application. You must navigate the gui using the D-Pad in the emulator. Your weather forecast application should still work.