Mobile App Design Project
GPS Location Display

Description:

This App takes the Location Values from the GPS receiver on the device and prints these values to TextView objects on the screen. Once GPS Latitude and Longitude values are harvested from the LocationListener, they can be deployed to other parts of the Application such as calculating distance traveled, speed, waypoints, altitude, and mapping.

Phase 1: Create the App Project

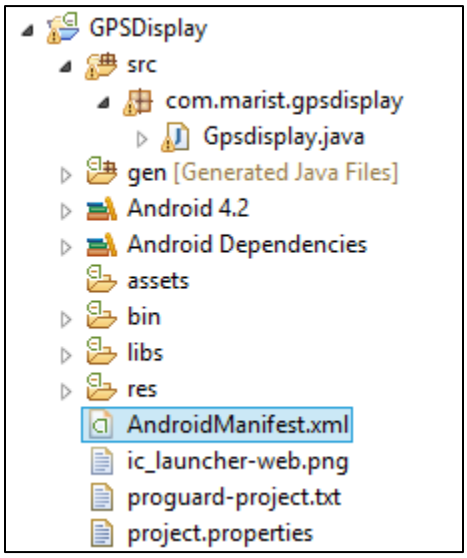
1. Start Eclipse and select New Project -> "Android Application Project"
2. Fill out the fields with the following:
    a. Application Name: GPSDisplay
    b. Project Name: GPSDisplay
    c. Package name: com.example.gpsdisplay
3. Click Next
4. Click Next at the Configure Project Screen
5. Click Next at the Configure Launcher Icon Screen
6. Click Next at the Create Activity Screen
7. Fill out the following fields in the New Blank Activity Screen
    a. Activity Name: Gpsdisplay
    b. LayoutName: activity_gpsdisplay
    c. Navigation Type: None
8. Click "Finish"

Phase 2: Manifest and User Interface and XML Design:

We need to access the GPS sensor in the Device, therefore we will need to modify the AndroidManifest.xml to set the permission. We then will build the User Interface in the activity_gpsdisplay.xml.

Process:

1. Open the AndroidManifest.xml file from the App resource tree:

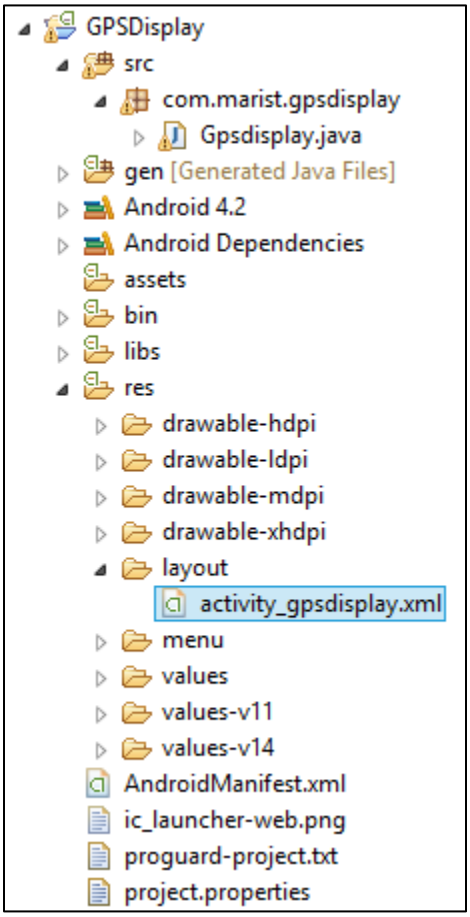


2. Add the following lines to the AndroidManifest.xml file between the <uses-sdk /> and the <application /> tags:

```
10
11      <uses-permission
12          android:name = "android.permission.ACCESS_FINE_LOCATION"
13          />
14
```

This will give the App permission to use the GPS service.

3. Go to the activity_gpsdisplay.xml file located in res/layout:



4. Delete the existing text and begin by establishing a LinearLayout:

```
<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"
    tools:context=".Gpsdisplay" >

```

5. We will now add 2 TextView boxes for Latitude display
    a. labelLat -> a TextView displaying the Text “Latitude”
    b. textLat -> a TextView that will display the output from the program.

    Add the following XML code:

```
 7
 8      <TextView
 9          android:id = "@+id/labelLat"
10          android:layout_width="wrap_content"
11          android:layout_height="wrap_content"
12          android:text="Latitude:" />
13
14      <TextView
15          android:id = "@+id/textLat"
16          android:layout_width="wrap_content"
17          android:layout_height="wrap_content"
18          android:text="---" />
19
```

6. We will now add 2 TextView objects to display the Longitude:
    a. labelLong -> TextView displaying the Text Longitude
    b. textLong -> TextView that will display the output from the program

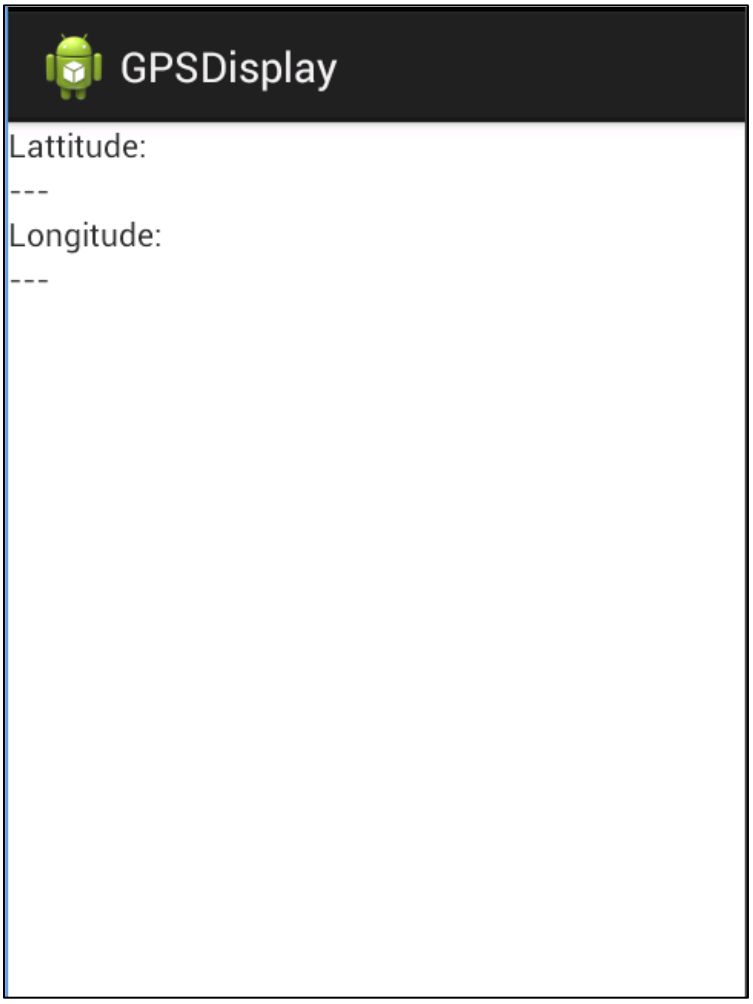
    Add the following XML code:

```
19
20      <TextView
21          android:id = "@+id/labelLong"
22          android:layout_width="wrap_content"
23          android:layout_height="wrap_content"
24          android:text="Longitude:" />
25
26      <TextView
27          android:id = "@+id/textLong"
28          android:layout_width="wrap_content"
29          android:layout_height="wrap_content"
30          android:text="---" />
31
```

7. Finish the XML with a </LinearLayout> end tag:

```
31
32  </LinearLayout>
33
```

8. The finished User Interface should look like this:



Phase 2: Writing the java code for the Gpsdisplay class

In the Gpsdisplay class we will create the fields representing the TextView objects that will display the latitude and longitude. Then we will create the LocationManager and Location Listener that will read the GPS data from the device and write these values to the screen.

Process:

1. Write the import statements for the Android Classes required for the project:

```
1  package com.example.gpsdisplay;
2
3  import android.location.Location;
4  import android.location.LocationListener;
5  import android.location.LocationManager;
6  import android.os.Bundle;
7  import android.app.Activity;
8  import android.content.Context;
9  import android.view.Menu;
10 import android.widget.TextView;
11
```

2. Write the class declaration and the two fields we will use to display the Latitude and Longitude.

```
11
12 public class Gpsdisplay extends Activity {
13
14     // Text Fields
15     private TextView textLat;
16     private TextView textLong;
17
```

3. Write the onCreate() method to connect the TextView fields to the User Interface XML.

```
17
18     @Override
19     protected void onCreate(Bundle savedInstanceState) {
20         super.onCreate(savedInstanceState);
21         setContentView(R.layout.activity_gpsdisplay);
22
23         // Local Text Fields
24         textLat = (TextView) findViewById(R.id.textLat);
25         textLong = (TextView) findViewById(R.id.textLong);
26
```

4. We now need to create the LocationManager instance and LocationListener instance to fetch the GPS data on location changed events. Add these objects to the onCreate() method and close the method.

```
26
27          // Use the Location Manager to obtain GPS locations
28          LocationManager locManager = (LocationManager)getSystemService(Context.LOCATION_SERVICE);
29          LocationListener locListener = new MyLocationListener();
30          locManager.requestLocationUpdates(LocationManager.GPS_PROVIDER, 0, 0, locListener);
31
32      } // end method onCreate()
33
```

5. Keep the onCreateOptionsMenu code intact. We will not change this code in this Application.

```
33
34      @Override
35      public boolean onCreateOptionsMenu(Menu menu) {
36          // Inflate the menu; this adds items to the action bar if it is present.
37          getMenuInflater().inflate(R.menu.activity_gpsdisplay, menu);
38          return true;
39      }
40
```

6. We will now write an inner class that will 'listen' to the Location and call the events that display the Latitude and Longitude. Start with the class declaration:

```
40
41      public class MyLocationListener implements LocationListener
42      {
43
```

7. The onLocationChanged method will do the work. We first will declare two doubles that receive the Latitude and Longitude data. We then will have the textLat and textLong objects display this data.

```
43
44          public void onLocationChanged(Location loc) {
45              // Get Latitude and Longitude
46              double lat = loc.getLatitude();
47              double lon = loc.getLongitude();
48
49              textLat.setText(String.valueOf(lat));
50              textLong.setText(String.valueOf(lon));
51
52          } // end onLocationChanged
```

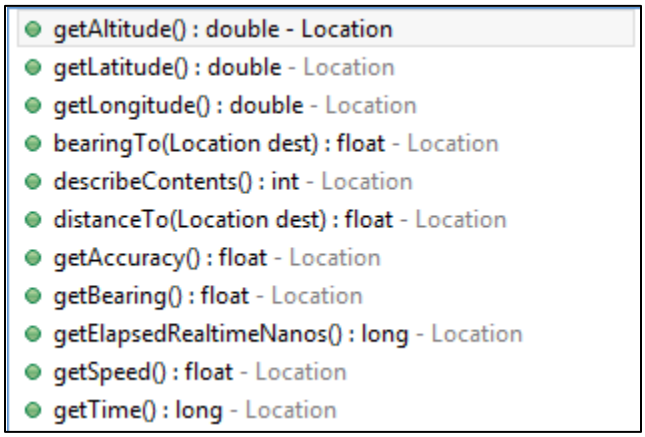
8. Now add the three additional required methods for this class. We will not use these methods (though they are required).

```
53
54          public void onProviderDisabled(String provider) {
55              // Not used in this App
56          }
57
58          public void onProviderEnabled(String provider) {
59              // Not used in this App
60          }
61
62          public void onStatusChanged(String provider, int status, Bundle extras) {
63              // Not used in this App
64          }
65
66      } // end Class MyLocationListener
67
```

9. Finally, close the class with a curly bracket

```
67
68  } // end Class Gpsdisplay
69
```

10. Save and run the App. The Location class provides several different data points. Experiment by adding more TextView objects to display this additional data. Several options are shown below:



11. Grading:
    a. GPS App Completed according to directions: 8.5 Points (Out of 10)
    b. Adding TextViews to Display / Calculate other GPS Data (.5 Points per view/Data)
    c. Summary – to earn a 10 on this assignment, complete the instructions and add 3 additional Text Views with different GPS Data.

12. NOTE!!! This App does not have an onPause() or onStop() method to turn off the GPS. Thus, the App will run down the battery with the because the GPS sensor will keep running even if you exit the App. To switch off the GPS for now, you will need to restart your device.