

13954 - COMP 11062: Mobile Networks And Smartphone Applications

Lab 6: My Location App

Learning Objectives

- Get permission and show GPS location.
- Create Dialog Builder.

MyLocation App

- In this exercise, you will program an app to get your current location.
- When you push a button, the app will try to get your current latitude and longitude using GPS signal.
- If the app detects that GPS is disabled, it will pop up a dialog to the user, asking to enable this functionality.

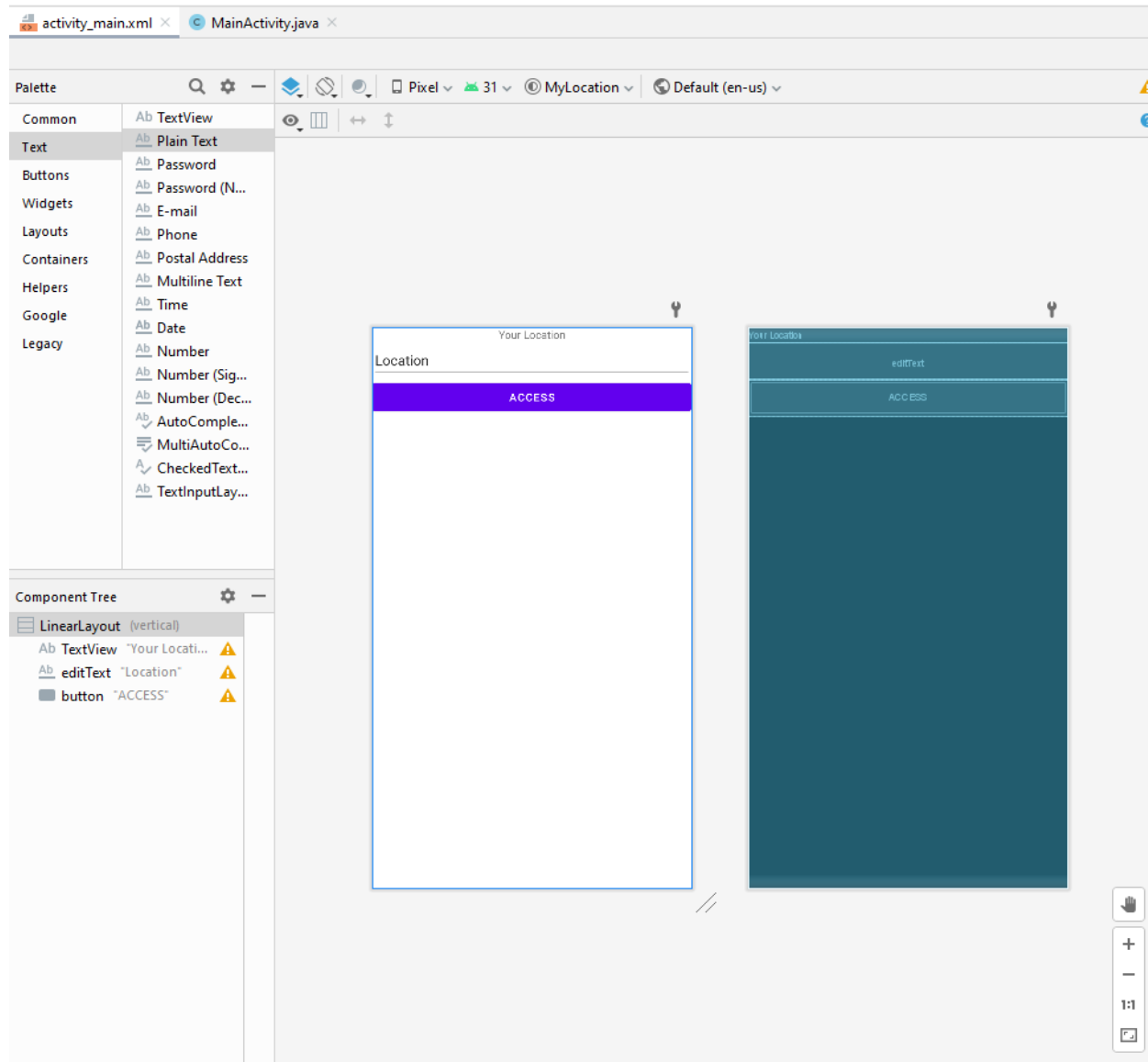
Task 1

Create the GUI

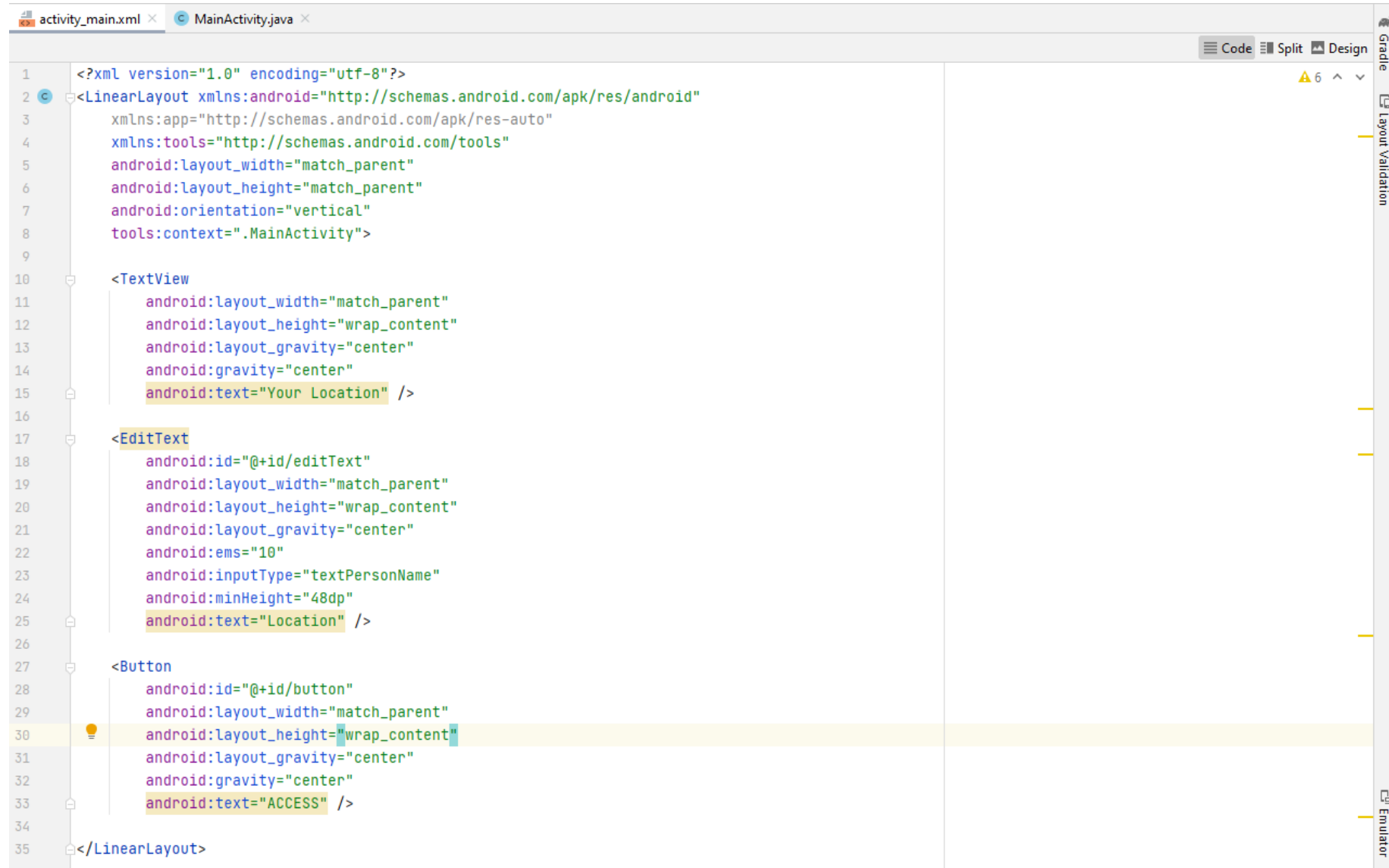
- In the Project tab, go to app->res->layout, then double click on file activity_main.xml to open the GUI Designer tool.
- Create a Vertical Layout, then put inside a TextView, and EditText (PlainText) and a Button.
- Change the text of the TextView to “Your Location”. Make the EditText non-editable.
- Run the app to check that everything works.

Note: Right click on Constraint Layout in the component tree to convert from Constraint layout to Linear layout. Then convert Linear layout to vertical layout in the similar way. Additionally, set android:layout_gravity and android:gravity=“center” either in the code or in the design view.

View of the App



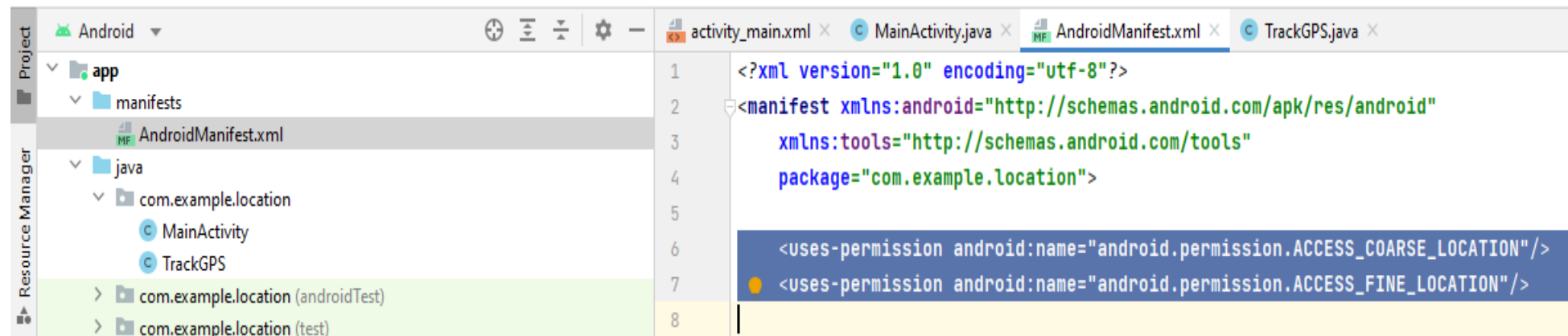
View of the Code



```
1 <?xml version="1.0" encoding="utf-8"?>
2 <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
3     xmlns:app="http://schemas.android.com/apk/res-auto"
4     xmlns:tools="http://schemas.android.com/tools"
5     android:layout_width="match_parent"
6     android:layout_height="match_parent"
7     android:orientation="vertical"
8     tools:context=".MainActivity">
9
10    <TextView
11        android:layout_width="match_parent"
12        android:layout_height="wrap_content"
13        android:layout_gravity="center"
14        android:gravity="center"
15        android:text="Your Location" />
16
17    <EditText
18        android:id="@+id/editText"
19        android:layout_width="match_parent"
20        android:layout_height="wrap_content"
21        android:layout_gravity="center"
22        android:ems="10"
23        android:inputType="textPersonName"
24        android:minHeight="48dp"
25        android:text="Location" />
26
27    <Button
28        android:id="@+id/button"
29        android:layout_width="match_parent"
30        android:layout_height="wrap_content"
31        android:layout_gravity="center"
32        android:gravity="center"
33        android:text="ACCESS" />
34
35 </LinearLayout>
```

Add App Permissions

- In the Project Tab, go to app -> manifests, and open the file AndroidManifest.xml.
- In the manifest section, before the application tag, add the following permissions.



Task 2

Create TrackGPS Class

- 1. On the Project tab:
go to app-> java -> com.example.location (your package name could be different) and right click, then choose new -> java class.
- 2. Assign to it the name TrackGPS.java
- 3. Inside TrackGPS.java, define the class as:

```
public class TrackGPS extends Service implements LocationListener
```
- 4. Use Alt+Enter on Service and LocationListener to import the required classes.
- 5. Create TrackGPS() constructor, getLatitude(), getLongitude() and canGetLocation() methods. See next slide.

TrackGPS.java

```
public class TrackGPS extends Service implements LocationListener {

    private final Context ctxt; // reference to current Activity
    boolean checkGPS = false; // check if GPS is available

    Location mylocation; // variable to store current location
    protected LocationManager locationManager;
    double latitude;
    double longitude;

    private static final long MINDELAY = 1000 * 60; // minimum time between updates
    private static final long MINDISTANCE = 10; // minimum distance between updates

    public TrackGPS(Context ctxt){
        this.ctxt = ctxt;
    }

    public double getLatitude(){
        if (mylocation != null) return mylocation.getLatitude();
        return latitude;
    }

    public double getLongitude(){
        if (mylocation != null) return mylocation.getLongitude();
        return longitude;
    }

    public boolean canGetLocation() { return this.checkGPS; }
```

Task 3

Add a method to get current Location

- Create a method called getLocation() that returns a Location.
- Add a line on the public creation method of TrackGPS to call getLocation on object creation:

```
public TrackGPS(Context ctxt) {  
    this.ctxt = ctxt;  
    getLocation();  
}
```

- Fill the method getLocation with the following code:

Building Function getLocation()

```
private Location getLocation(){
    try{
        locationManager = (LocationManager) ctxt.getSystemService(LOCATION_SERVICE);
        // get GPS status
        checkGPS = locationManager.isProviderEnabled(LocationManager.GPS_PROVIDER);
        if (checkGPS){
            try {
                locationManager.requestLocationUpdates(LocationManager.GPS_PROVIDER, MINDELAY, MINDISTANCE, listener: this);
                if (locationManager != null){
                    mylocation = locationManager.getLastKnownLocation(LocationManager.GPS_PROVIDER);
                    if (mylocation != null){
                        latitude = mylocation.getLatitude();
                        longitude = mylocation.getLongitude();
                    }
                }
            } catch (SecurityException e) {
                Toast.makeText(ctxt, text: "No permission to access GPS", Toast.LENGTH_SHORT).show();
            }
        }
        else {
            Toast.makeText(ctxt, text: "No service provider available", Toast.LENGTH_SHORT).show();
        }
    } catch (Exception e) {
        e.printStackTrace();
    }
    return mylocation;
}
```

Task 4

Program the buttons

- In MainActivity.class, define the following:

```
public class MainActivity extends AppCompatActivity {  
    private Button button;  
    private EditText editText;  
    private TrackGPS gps;
```

- Inside onCreate, add an OnClickListener to button, containing the following code

```
@Override  
protected void onCreate(Bundle savedInstanceState) {  
    super.onCreate(savedInstanceState);  
    setContentView(R.layout.activity_main);  
  
    button=(Button) findViewById(R.id.button);  
    button.setOnClickListener(new View.OnClickListener() {  
        @Override  
        public void onClick(View v) {  
  
            gps=new TrackGPS( ctxt: MainActivity.this);  
            if (gps.canGetLocation()){  
                editText=(EditText)findViewById(R.id.editText);  
                editText.setText("Latitude: "+gps.getLatitude()+"Longitude: "+gps.getLongitude());  
            }  
        }  
    });  
}
```

Issues with higher APIs

-It is likely to face location permission issues with higher APIs, specially from API 23 onwards. In that case, we need to add extra code that will check if permission is granted and request location permission if it is not already granted.

```
public class MainActivity extends AppCompatActivity {

    private Button button;
    private EditText editText;
    private TrackGPS gps;

    private static final int REQUEST_CODE_PERMISSION = 1;
    String[] mPermission = {Manifest.permission.ACCESS_FINE_LOCATION};

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        button=(Button) findViewById(R.id.button);
        button.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                if(Build.VERSION.SDK_INT>= 23) {

                    if (checkSelfPermission(mPermission[0]) != PackageManager.PERMISSION_GRANTED) {
                        ActivityCompat.requestPermissions( MainActivity.this, mPermission, REQUEST_CODE_PERMISSION);
                        return;
                    }
                }

                gps=new TrackGPS( MainActivity.this);
                if (gps.canGetLocation()){
                    editText=(EditText)findViewById(R.id.editText);
                    editText.setText("Latitude: "+gps.getLatitude()+"Longitude: "+gps.getLongitude());
                }
            }
        });
    }
}
```

Creating Dialogs with Dialog.Builder

- The class Dialog.Builder is a helper class that generates any type of dialog that our apps might need. It follows the Builder java pattern.
- We first instantiate an object of type Dialog.Builder, then add to it the parameters that we want for our dialog, like the title, number of buttons, and more.
- We then add labels and onClick listeners to each button to implement the logic we need.

Task 5

Add options to enable/disable GPS

- Open TrackGPS.java and create a method called showAlert that displays the user a Dialog. Set the title to “GPS not enabled” and the text “Do you want to turn on GPS?”.
- Follow the next slide.

Task 5 Add options to enable/disable GPS

```
//cetxt is the variable that stores the reference to the activity because this is a pure java class  
//if dialog is created inside and Activity, replace cetxt with MainActivity.this  
  
public void showAlert() {  
    AlertDialog.Builder dialog = new AlertDialog.Builder(ctxt);  
    dialog.setTitle("GPS disabled");  
    dialog.setMessage("Do you want to turn on GPS?");  
    dialog.setPositiveButton( text: "YES", new DialogInterface.OnClickListener() {  
  
        @Override  
        public void onClick(DialogInterface dialog, int which) {  
            Intent intent = new Intent(Settings.ACTION_LOCATION_SOURCE_SETTINGS);  
            ctxt.startActivity(intent);  
        }  
    });  
    dialog.setNegativeButton( text: "NO", new DialogInterface.OnClickListener() {  
  
        @Override  
        public void onClick(DialogInterface dialog, int which) {  
            dialog.cancel();  
        }  
    });  
    dialog.show();  
}
```


Disable GPS when user closes app

- Open TrackGPS.java and create a method called stopGPS to stop listening to the GPS sensor.
- Add the following code to the method:

```
public void stopGPS() {  
    if (locationManager != null) {  
        try {  
            locationManager.removeUpdates(TrackGPS.this);  
        } catch (SecurityException e) {  
            Toast.makeText(ctxt, "No permission to access GPS", Toast.LENGTH_SHORT).show();  
        }  
    }  
}
```

Update Activity to handle GPS

- Open MainActivity.java and update the OnClickListener of button:

```
if (gps.canGetLocation()) {  
    editText = (EditText)findViewById(R.id.editText);  
    editText.setText("Latitude: "+gps.getLatitude()+" Longitude: "+gps.getLongitude());  
} else {  
    gps.showAlert();  
}
```

- Override the method called onDestroy() to stop the GPS.

```
@Override  
protected void onDestroy() {  
    super.onDestroy();  
    gps.stopGPS();  
}
```

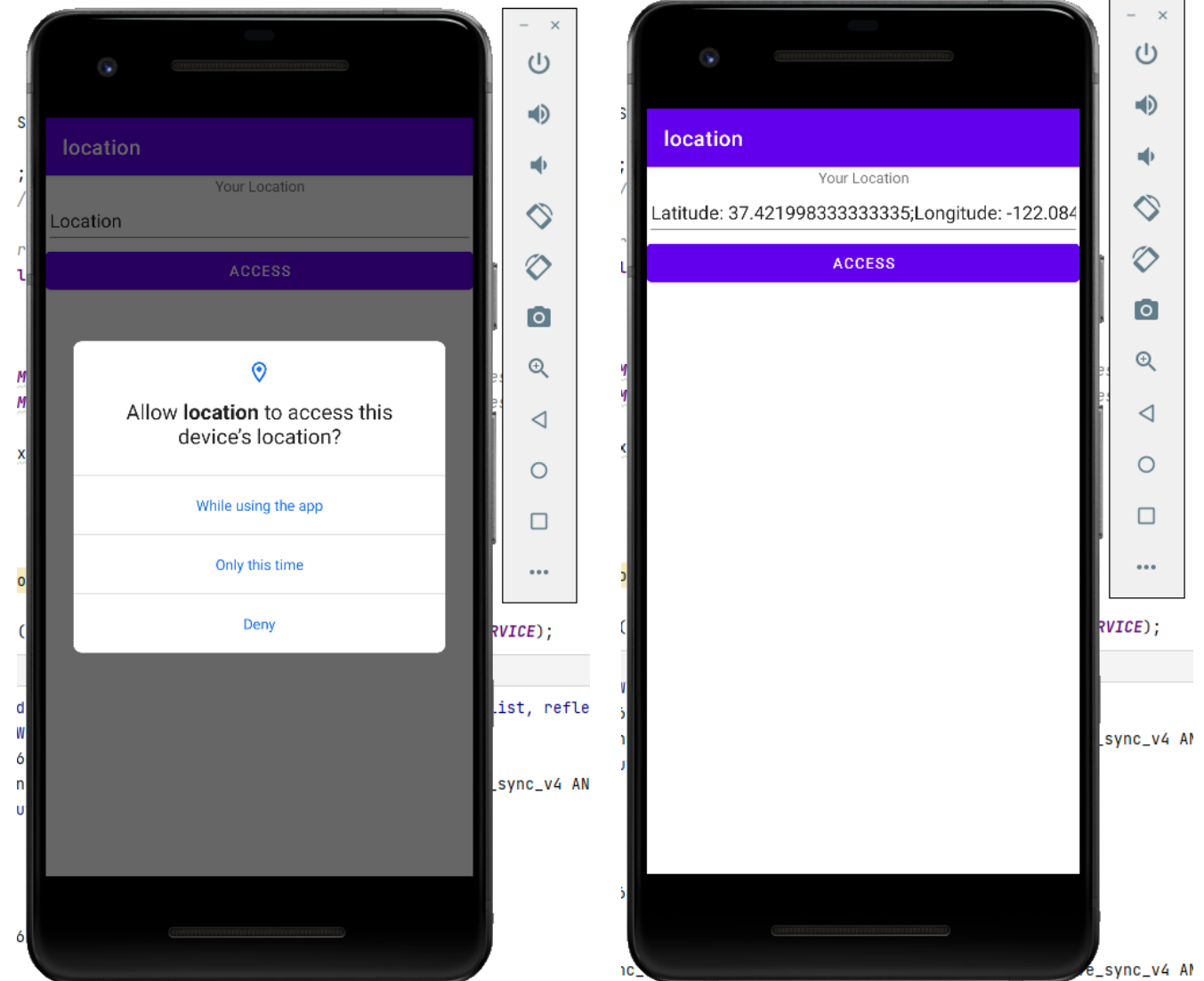
At last....

Add these methods, if you do not have them, in your TrackGPS.java file and import any necessary classes.

```
| @Nullable  
| @Override  
| public IBinder onBind(Intent intent) {  
|     return null;  
| }  
  
| @Override  
| public void onLocationChanged(@NonNull Location location) {  
  
| }  
| public void onStatusChanged(String provider, int status, Bundle extras){  
  
| }  
| }
```

Emulator Output

- After running the program, the output should like in the screenshots.
- Make sure you switch on the location service by going to the settings.
- You will see by default the output as a Los Angeles location, because we take data from the Emulator's GPS.



Exercise:

- Add two extra EditText widgets to activity_main.xml. Use one EditText to display latitude, one for longitude and one for altitude. (You will need to call setText() method on every EditText)
- Modify TrackGPS.java and MainActivity.java to also display altitude.
- Hints:
 - Inside TrackGPS.java declare a **double altitude** variable similar to latitude and longitude.
 - Inside TrackGPS.java create a method **getAltitude()** similar to getLatitude() and getLongitude().
- **Submission:** Submit screenshots of your emulator showing your Banner ID and output of the exercise; Submit screenshots of code snippets you wrote for exercise. You can create word document and must submit as single PDF combined with other labs.