Location

Using location services in Android

Contents

CONTENTS 2

USING THE LOCATION METHODS..... 3

PRACTICAL EXERCISE 1 5

Using the location methods

For accessing the location systems we need to use *LocationManager*, which is obtained with *LOCATION_SERVICE*.

Firstly we have to create two attributes as follows:

```
LocationManager mylocManager;
LocationListener mylocListener;
```

LocationManager will allow to access the location service and LocationListener will be the class that will manage the different location events.

Then we have to write the following line of code in the "*onResume"* method.

```
mylocManager = (LocationManager)
getSystemService(Context.LOCATION_SERVICE);
```

Now we instantiate the MyLocationListener class, which will implement *LocationListener*, which will handle the different location events.

```
mylocListener = new MyLocationListener();
```

The following line of code will register the listener receiving the coordinates from the device.

```
mylocManager.requestLocationUpdates(LocationManager.NETWORK_PROVIDER,
0, 0, mylocListener);
```

The last line of code receives coordinates by using the network of the phone (GPRS, 3G, etc.). If we want to use the GPS, we have to use the following code:

```
mylocManager.requestLocationUpdates(LocationManager.GPS_PROVIDER, 0, 0,
mylocListener);
```

We could change between systems using the different parameters: *LocationManager.GPS_PROVIDER* and *LocationManager.NETWORK_PROVIDER*.

The "*onResume*" method should be as follows:

```
@Override
public void onResume()
{
	super.onResume();
	mylocManager = (LocationManager)
getSystemService(Context.LOCATION_SERVICE);
	mylocListener = new MyLocationListener();
	mylocManager.requestLocationUpdates(LocationManager.NETWORK_PROVI
DER, 0, 0, mylocListener);
}
```

On the other hand, in the "*onPause*" method we have to remove the listener because if the application is closed or the activity is in paused state, the listener

continues working. So if we do not do this, the battery of the device will run out quickly.

```
@Override
public void onPause()
{
	super.onPause();
	mylocManager.removeUpdates(mylocListener);
}
```

Finally, we have to create the class that implements the *LocationListener.* It will have the methods for receiving the coordinates.

```
public class MyLocationListener implements LocationListener {
@Override
public void onLocationChanged(Location loc) {
	String coordinates = "My coordinates are: " + "Latitude = "
	+ loc.getLatitude() + "- Longitude = " + loc.getLongitude();
	Toast.makeText(getApplicationContext(), coordinates,
				Toast.LENGTH_LONG).show();
	}

	@Override
	public void onProviderDisabled(String provider) {
		Toast.makeText( getApplicationContext(),"Gps
Disabled",Toast.LENGTH_SHORT ).show();
	}

	@Override
	public void onProviderEnabled(String arg0) {
		Toast.makeText( getApplicationContext(),"Gps
Enabled",Toast.LENGTH_SHORT ).show();
	}

	@Override
	public void onStatusChanged(String provider, int status, Bundle
extras) {
		// TODO Auto-generated method stub
	}
}
```

The "*onLocationChanged*" method allows receiving the location coordinates. It is important to highlight that the system uses the coordinates system called "geodesic decimal degrees", but many other information systems use other different location systems.

The "*getLatitude()*" and "*getLongitude()*" methods return the latitude and longitude, respectively. The first one is the distance from the Equator until the position of the coordinate. If the place is in the north this value will be positive, if instead it is in the south it will be a negative value. The longitude is the distance from the Greenwich meridian: if the place is to the left, it will be a negative value; and positive if the position is to the right of the meridian.

```
loc.getLatitude();
loc.getLongitude();
```

The "*onProviderEnabled*" *and* "*onProviderDisabled*" methods detect if the location system is enabled or disabled.

The "*onStatusChanged*" method allows knowing the state of the GPS, according to the following values:

```
public static final int OUT_OF_SERVICE = 0;
public static final int TEMPORARILY_UNAVAILABLE = 1;
public static final int AVAILABLE = 2;
```

Finally, we have to add this line to the manifest file. It is the permission for accessing the location systems of the device. We will add this line in the "*AndroidManifest.xml*" file, before the *"application"* node.

```
<uses-permission
     android:name="android.permission.ACCESS_FINE_LOCATION" />
```

Practical exercise 1

In the DDMS perspective (*Window -> Open Perspective -> Other… -> DDMS* and click "OK") open the Emulator Control view (*Window -> Show View -> Other… -> Emulator Control* and click "OK") and use the Location Controls section to send different positions to the emulator and test that the operation is correct.