Εισαγωγή στην Ανάπτυξη Android Εφαρμογών

Ενότητα 5 – GPS, Hardware & Αισθητήρες

5.1 Location-Based Services (LBS)

Οι υπηρεσίες βάσει τοποθεσίας (Location-Based Services) αποτελούν μία από τις πιο ενδιαφέρουσες δυνατότητες της πλατφόρμας, δεν είναι τυχαίο άλλωστε πως γύρω από αυτήν έχουν χτιστεί οι πιο δημοφιλείς εφαρμογές για Android. Η δυνατότητα αυτή σε συνδυασμό με την έμφυτη υποστήριξη της κορυφαίας εφαρμογής χαρτών της Google (Google Maps) από την πλατφόρμα αποτελεί ένα από τα δυνατά σημεία των συσκευών Android έναντι των ανταγωνιστών τους. Οι υπηρεσίες βάσει τοποθεσίας μπορούν να ενσωματωθούν σε πολλές λειτουργίες όπως για παράδειγμα σε μία αναζήτηση στο web, στη λήψη φωτογραφιών, την κοινωνική δικτύωση κ.α. Στην υποενότητα αυτή θα δούμε πως μπορούμε να λάβουμε πληροφορίες σχετικά με την τρέχουσα τοποθεσία μιας συσκευής ώστε να τις χρησιμοποιήσουμε στις εφαρμογές μας.
Η πρόσβαση στις υπηρεσίες τοποθεσίας γίνεται μέσω των περιεχομένων του πακέτου `android.location`, βασικότερες από τις οποίες είναι οι εξής τρεις:

- `LocationManager`: Η κεντρική κλάση που παρέχει πρόσβαση στις εν λόγω υπηρεσίες
- `LocationListener`: Interface που παρέχει ενημερώσεις μέσω του `LocationManager` όταν η θέση της συσκευής αλλάζει
- `Location`: Η κλάση αυτή αναπαριστά μία γεωγραφική τοποθεσία σε δεδομένη χρονική στιγμή

Για να αποκτήσουμε πρόσβαση στην υπηρεσία παροχής πληροφοριών τοποθεσίας θα πρέπει να αρχικοποιήσουμε μία αναφορά τύπου `LocationManager` ώστε να δείχνει στο υπάρχον στιγμιότυπο της υπηρεσίας συστήματος `LOCATION_SERVICE`, ως εξής:

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

Έχοντας αρχικοποιήσει έναν `LocationManager` θα πρέπει επιλεγεί ο πάροχος εντοπισμού θέσης που θα χρησιμοποιηθεί. Υπάρχουν διαφορετικοί πάροχοι κάθε ένας από τους οποίους χρησιμοποιεί διαφορετική τεχνολογία για τον εντοπισμό της θέσης της συσκευής. Έτσι λοιπόν, οι διαθέσιμοι πάροχοι είναι το AGPS (Assisted Global Positioning System) με το οποίο είναι εφοδιασμένες σχεδόν όλες οι νέες συσκευές Android, η εύρεση θέσης βάσει της κεραίας του παρόχου κινητής τηλεφωνίας που εξυπηρετεί τη συσκευή (triangulation) και τέλος βάσει των συντεταγμένων γνωστών Wi-Fi hotspots που μπορεί να χρησιμοποιούνται. Κάθε μία από τις μεθόδους αυτές παρέχει διαφορετικό επίπεδο ακρίβειας καθώς και αντίστοιχα διαφορετικές απαιτήσεις κατανάλωσης ενέργειας. Μπορούμε να θέσουμε τον `LocationManager` να επιλέξει τον κατάλληλο πάροχο βάσει κριτηρίων που έχουμε τη δυνατότητα να ορίσουμε, ή να ορίσουμε ρητά συγκεκριμένο πάροχο.
Στην πρώτη περίπτωση κάνουμε χρήση ενός αντικειμένου της κλάσης `android.location.Criteria` η οποία μας επιτρέπει να ορίσουμε τις επιθυμητές παραμέτρους που στη συνέχεια θα χρησιμοποιηθούν από τον `LocationManager` για να επιλεγεί ο πιο κατάλληλος διαθέσιμος πάροχος. Η εφαρμογή που ακολουθεί κάνει χρήση των παραπάνω και προβάλλει στην οθόνη της συσκευής τις

συντεταγμένες της τελευταίας θέσης της συσκευής. Το UI της αποτελείται από ένα μόνο TextView το οποίο προβάλλει τη συγκεκριμένη πληροφορία. Το XML κείμενο που ορίζει το interface είναι το ακόλουθο:

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
   android:orientation="vertical" android:layout_width="fill_parent"
   android:layout_height="fill_parent">
   <TextView android:id="@+id/textView" android:layout_width="fill_parent"
       android:layout_height="wrap_content"/>
</LinearLayout>
```

Ο κώδικας του κεντρικού Activity είναι ο εξής:

```
package elearning.location;

import android.app.Activity;
import android.content.Context;
import android.location.Criteria;
import android.location.Location;
import android.location.LocationManager;
import android.os.Bundle;
import android.widget.TextView;

public class LastLocationActivity extends Activity {
   private LocationManager lm;
   private TextView tv;

   /** Called when the activity is first created. */
   @Override
   public void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.main);

      tv = (TextView) findViewById(R.id.textView);
      lm = (LocationManager) getSystemService(Context.LOCATION_SERVICE);
      Criteria criteria = new Criteria();
      criteria.setAccuracy(Criteria.ACCURACY_FINE);
      criteria.setPowerRequirement(Criteria.POWER_LOW);
      criteria.setSpeedRequired(false);
      criteria.setBearingRequired(false);
      criteria.setAltitudeRequired(false);
      String provider = lm.getBestProvider(criteria, true);
      Location location = lm.getLastKnownLocation(provider);
      tv.setText("Last location latitude: " + location.getLatitude() +
                  " - longitude: " + location.getLongitude() + "\n" + provider);
   }
}
```

Για να λειτουργήσει σωστά η εφαρμογή θα πρέπει στο αρχείο *AndroidManifest.xml* να ορίσουμε το κατάλληλο `<uses-permission>` tag για τα δικαιώματα πρόσβασης `ACCESS_FINE_LOCATION`. Εκτελώντας τη σε μία πραγματική συσκευή θα πάρετε έξοδο σαν αυτήν που ακολουθεί:

Last location latitude: 37.0675722 - longitude: 22.4301269
gps

Στον παραπάνω κώδικα έχοντας αρχικοποιήσει τη μεταβλητή-μέλος τύπου `LocationManager` με το υπάρχον στιγμιότυπο της υπηρεσίας δημιουργούμε ένα αντικείμενο τύπου `Criteria` όπου θέτουμε τις παραμέτρους που θα χρησιμοποιηθούν για την επιλογή του κατάλληλου διαθέσιμου παρόχου. Συγκεκριμένα επιθυμούμε να έχουμε μεγάλη ακρίβεια και τη μικρότερη δυνατή κατανάλωση, ενώ δεν μας ενδιαφέρουν στοιχεία όπως η ταχύτητα, το ύψος ή η πορεία.
Η `getBestProvider()` θα επιστρέψει το όνομα του κατάλληλου διαθέσιμου παρόχου με τη μορφή αλφαριθμητικού, το οποίο και περνάμε ως παράμετρο στην `getLastKnownLocation()`. Η μέθοδος αυτή επιστρέφει την τελευταία θέση που έχει καταγράψει ο πάροχος που λαμβάνει ως παράμετρο και είναι χρήσιμη γιατί αποφεύγεται η διαδικασία εντοπισμού της τρέχουσας θέσης της συσκευής που μπορεί να είναι χρονοβόρα. Η `getLastKnownLocation()` θα επιστρέψει ένα αντικείμενο τύπου `Location` που όπως αναφέρθηκε νωρίτερα αποθηκεύει τις συντεταγμένες της θέσης της συσκευής τη δεδομένη ημερομηνία/ώρα με τη μορφή UTC timestamp. Στη συνέχεια οι `getLatitude()` και `getLongitude()` θα επιστρέψουν τις γεωγραφικές συντεταγμένες της τελευταίας τοποθεσίας της συσκευής, οι οποίες και προβάλλονται στο TextView της εφαρμογής, μαζί με το όνομα του παρόχου που επιλέχθηκε για τον εντοπισμό και που στη συγκεκριμένη περίπτωση είναι το GPS.
Αυτό είναι φυσιολογικό, μιας και στα κριτήρια επιλογής παρόχου έχουμε ορίσει πως επιθυμούμε να έχουμε μεγάλη ακρίβεια στον εντοπισμό της θέσης, περνώντας στην `setAccuracy()` τη σταθερά `ACCURACY_FINE`. Για να λειτουργήσει σωστά η εφαρμογή και χωρίς προβλήματα, θα πρέπει το GPS να έχει αποθηκευμένη την τελευταία θέση της συσκευής διαφορετικά θα προκληθεί εξαίρεση τύπου `NullPointerException`. Σε περίπτωση που δε μας ενδιέφερε να έχουμε μεγάλη ακρίβεια θα μπορούσαμε να περάσουμε στην `setAccuracy()` ως παράμετρο τη σταθερά `ACCURACY_COARSE` και να ορίσουμε δικαιώματα πρόσβασης `ACCESS_COARSE_LOCATION` στο αρχείο *AndroidManifest.xml*. Με αυτά τα κριτήρια ο πάροχος εντοπισμού που θα επιλεγόταν θα ήταν το δίκτυο κινητής τηλεφωνίας, οπότε στην οθόνη της εφαρμογής το όνομα του παρόχου που θα εμφανιζόταν θα ήταν το network.
Μπορούμε να βελτιώσουμε την παραπάνω εφαρμογή ώστε να ενημερώνεται ανά τακτά χρονικά διαστήματα με τη νέα θέση της συσκευής. Αυτό μπορεί να γίνει εύκολα υλοποιώντας το `LocationListener` interface, οπότε ο κώδικας του κεντρικού Activity θα πάρει την ακόλουθη μορφή:

```
package elearning.location;

import android.app.Activity;
import android.content.Context;
import android.location.Criteria;
import android.location.Location;
import android.location.LocationListener;
import android.location.LocationManager;
import android.os.Bundle;
import android.widget.TextView;

public class LocationUpdateActivity extends Activity implements LocationListener {
   private LocationManager lm;
   private TextView tv;
   private Location location;

   @Override
   public void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.main);

      tv = (TextView) findViewById(R.id.textView);
      lm = (LocationManager) getSystemService(Context.LOCATION_SERVICE);
```

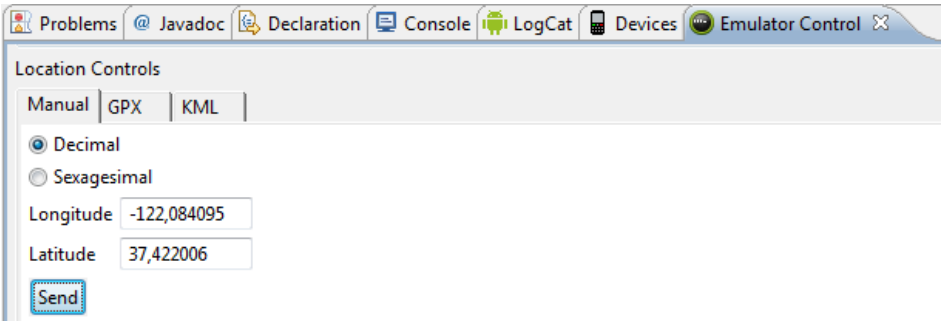
```
        Criteria criteria = new Criteria();
        criteria.setAccuracy(Criteria.ACCURACY_FINE);
        criteria.setPowerRequirement(Criteria.POWER_LOW);
        criteria.setSpeedRequired(false);
        criteria.setBearingRequired(false);
        criteria.setAltitudeRequired(false);
        String provider = lm.getBestProvider(criteria, true);
        location = lm.getLastKnownLocation(provider);
        lm.requestLocationUpdates(provider, 5000, Float.parseFloat("2.0"), this);
    }

    @Override
    public void onLocationChanged(Location loc) {
    location = loc;
    tv.setText("Last location latitude: " + location.getLatitude() +
                    " - longitude: " + location.getLongitude());
    }

    @Override
    public void onProviderDisabled(String provider) { }

    @Override
    public void onProviderEnabled(String provider) { }

    @Override
    public void onStatusChanged(String a, int b, Bundle c) { }
}
```

Η βασικότερη αλλαγή στην κλάση είναι πως έχει τεθεί να υλοποιεί το interface `LocationListener` όπως προαναφέρθηκε. Θα μπορούσαμε να έχουμε υλοποιήσει το `LocationListener` με τη μορφή εσωτερικής κλάσης όπως έχουμε κάνει σε προηγούμενες ενότητες με το ίδιο ακριβώς αποτέλεσμα. Το interface αυτό λαμβάνει ενημερώσεις για τη νέα θέση της συσκευής και υλοποιώντας τη μέθοδο `onLocationChange()`, γράφουμε κώδικα που θα εκτελείται σε κάθε φορά που λαμβάνει χώρα αυτό το συμβάν. Στη συγκεκριμένη μέθοδο προτείνεται να μη γράφουμε κώδικα που απαιτεί μεγάλο επεξεργαστικό φόρτο, παρά να περνάμε τα δεδομένα για επεξεργασία σε ένα νέο νήμα. Στο συγκεκριμένο παράδειγμα απλά προβάλλουμε στο TextView της εφαρμογής τις συντεταγμένες της νέας θέσης.

Εκτός από την `onLocationChange()`, για να έχουμε συμπαγή κλάση θα πρέπει επίσης να υλοποιήσουμε και τις μεθόδους `onProviderDisabled()`, `onProviderEnabled()` και `onStatus Changed()`, έστω κι αν αυτές δεν περιέχουν καθόλου κώδικα στο σώμα του όπως για παράδειγμα στην περίπτωση της εφαρμογής μας. Η πρώτη καλείται αυτόματα όταν ο τρέχων πάροχος απενεργοποιηθεί από τον χρήστη, η δεύτερη όταν ενεργοποιηθεί από τον χρήστη, ενώ η τρίτη όταν μεταβληθεί η κατάστασή του (π.χ. γίνει ξανά διαθέσιμος μετά από μικρό διάστημα μη διαθεσιμότητας) κι επομένως μπορούμε να γράψουμε κώδικα που θα εκτελείται ώς αντίδραση σε αυτά τα συμβάντα, αν το επιθυμούμε.

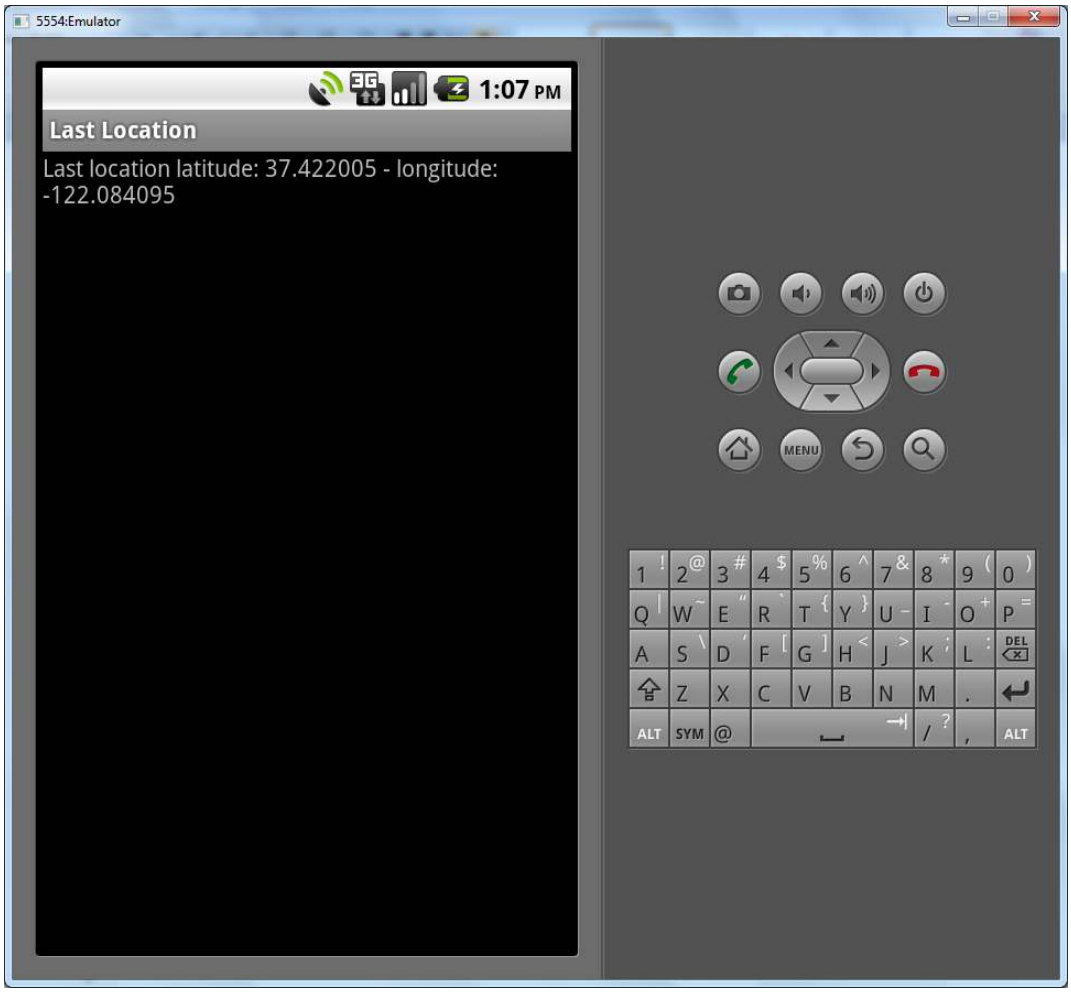
Μία ακόμη αλλαγή σε σχέση με την προηγούμενη εφαρμογή είναι πως έχουμε θέσει ως μεταβλητή-μέλος της κλάσης μία αναφορά τύπου `Location`. Έχοντας αρχικοποιήσει έναν πάροχο εντοπισμού θέσης καλούμε μέσω του αντικειμένου τύπου `LocationManager` την `requestLocationUpdates()` η οποία ορίζει το κάθε πότε θα στέλνονται ενημερώσεις στον Listener. Η πρώτη παράμετρος που λαμβάνει είναι ο τρέχων πάροχος εντοπισμού θέσης, η δεύτερη το ελάχιστο χρονικό διάστημα σε milliseconds που θα μεσολαβεί μεταξύ των ενημερώσεων, η τρίτη την ελάχιστη διαφορά απόστασης μεταξύ των δύο θέσεων σε μέτρα και τέλος, ο Listener που θα λαμβάνει τις ενημερώσεις αυτές. Στο

συγκεκριμένο παράδειγμα έχουμε ορίσει χρονικό διάστημα 5 sec και 2 μέτρα διαφορά μεταξύ των ενημερώσεων. Σε μία πραγματική εφαρμογή θα έπρεπε να χρησιμοποιούσαμε μεγαλύτερες τιμές για να περιορίσουμε την κατανάλωση σε ενέργεια.



Σχήμα 5.1

Μπορούμε να ελέγξουμε την εφαρμογή αυτή στον emulator στέλνοντάς του εικονικές συντεταγμένες μέσω του εργαλείου Emulator Control που χρησιμοποιήσαμε και στην προηγούμενη ενότητα, όπως φαίνεται στο σχήμα 5.1. Πληκτρολογώντας χειρωνακτικά τιμές για γεωγραφικό πλάτος και μήκος και πατώντας **Send**, τις στέλνουμε στον emulator προσομειώνοντας την αντίστοιχη λειτουργία του GPS.



Σχήμα 5.2

Στο σχήμα 5.2 βλέπετε την έξοδο της εφαρμογής στον emulator, έχοντας αποστείλει μέσω του Emulator Control το ζευγάρι γεωγραφικών συντεταγμένων του σχήματος 5.1.

5.2 Geocoding

Ο όρος geocoding αναφέρεται στη διαδικασία αντιστοίχισης μιας διεύθυνσης στο ζεύγος γεωγραφικών συντεταγμένων όπου υπάγεται και που μπορεί να φανεί χρήσιμη για τις εφαρμογές μας. Για να λειτουργήσει σωστά η διαδικασία απαιτείται να γίνει χρήση μιας client Geocoder υπηρεσίας μιας και το Android SDK δεν παρέχει έμφυτη υποστήριξη για τη συγκεκριμένη λειτουργία. Παρόλα αυτά, το Google Maps API την παρέχει και μπορούμε να την χρησιμοποιήσουμε αρκεί η συσκευή μας να έχει σύνδεση στο διαδίκτυο είτε μέσω Wi-Fi, είτε μέσω 3G. Η υποτυπώδης εφαρμογή που ακολουθεί στέλνει ένα ερώτημα για τη διεύθυνση της πόλης της Σπάρτης και προβάλλει στην οθόνη της συσκευής το ζεύγος των γεωγραφικών συντεταγμένων που επέστρεψε η υπηρεσία Geocoder.

```
package elearning.geocoding;

import java.io.IOException;
import java.util.List;
import android.app.Activity;
import android.location.Address;
import android.location.Geocoder;
import android.os.Bundle;
import android.widget.TextView;

public class GeocodingActivity extends Activity {
   private TextView tv;

   @Override
   public void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.main);

      tv = (TextView) findViewById(R.id.textView);
      List<Address> addresses;

      String myAddress = "Sparta, Greece";
      Geocoder gc = new Geocoder(this);
      try {
         addresses = gc.getFromLocationName(myAddress, 1);
         if(addresses != null){
            Address x = addresses.get(0);
            StringBuilder sb = new StringBuilder("Address: \n");
            sb.append("latitude: ").append(x.getLatitude());
            sb.append("\nlongitude: ").append(x.getLongitude());
            tv.setText(sb.toString());
         }
      } catch(IOException ioe) {
         tv.setText(ioe.getMessage());
      }
   }
}
```

Εκτελώντας την εφαρμογή σε μία συσκευή με σύνδεση στο διαδίκτυο θα πάρουμε την έξοδο:

Address:
latitude: 37.074301
longitude: 22.432584

Σε περίπτωση που η συσκευή δε διαθέτει σύνδεση στο διαδίκτυο ούτε μέσω Wi-Fi ούτε μέσω 3G, το TextView θα προβάλλει το μήνυμα:

Unable to parse response from server

Στην εφαρμογή αυτή ορίζουμε hard-coded τη διεύθυνση "Sparta, Greece", η οποία θα αποτελέσει το αντικείμενο του ερωτήματος που θα στείλουμε στην υπηρεσία `Geocoder`. Βασικό συστατικό για τη διαδικασία geocoding αλλά και reverse geocoding όπως θα δούμε στην αμέσως επόμενη υποενότητα είναι η κλάση `Geocoder` του πακέτου `android.location`. Κατά τη δημιουργία ενός αντικειμένου τύπου Geocoder περνάμε ως παράμετρο το τρέχον context και προαιρετικά ένα locale. Στην εφαρμογή μας χρησιμοποιείται το locale της συσκευής.
Η έκδοση της υπερφορτωμένης μεθόδου `getFromLocation()` που χρησιμοποιούμε λαμβάνει ως πρώτη παράμετρο τη διεύθυνση που επιθυμούμε να μεταφράσουμε σε συντεταγμένες και ως δεύτερη το μέγιστο πλήθος αποτελεσμάτων που επιθυμούμε να μας επιστρέψει. Η `getFromLocation()` θα επιστρέψει μία λίστα από αντικείμενα τύπου `android.location.Address` ή `null` αν η μετάφραση αποτύχει. Δεδομένου του ότι έχουμε ορίσει πως επιθυμούμε να μας επιστραφεί ένα και μόνο αποτέλεσμα, λαμβάνουμε το μοναδικό αποτέλεσμα αυτό από τη λίστα και προβάλλουμε το γεωγραφικό του πλάτος και μήκος στο TextView.

5.3 Αντίστροφο Geocoding

Η αντίστροφη διαδικασία, η αντιστοιχία δηλαδή ενός ζεύγους γεωγραφικών συντεταγμένων σε μία διεύθυνση ονομάζεται reverse geocoding και πρόκειται για μία εξίσου χρήσιμη λειτουργία για τις εφαρμογές μας. Όπως στην περίπτωση του απλού geocoding, έτσι και στο αντίστροφο είναι απαραίτητη η χρήση μιας υπηρεσίας Geocoder η οποία αναλαμβάνει να φέρει σε πέρας την όλη διαδικασία. Στην εφαρμογή που ακολουθεί γίνεται επίδειξη της λειτουργίας αυτής. Για το ερώτημα χρησιμοποιούνται οι συντεταγμένες που επιστράφηκαν από την υπηρεσία Geocoder κατά την εκτέλεση της εφαρμογής της προηγούμενης υποενότητας. Ο κώδικας του κεντρικού Activity είναι ο ακόλουθος:

```
package elearning.geocoding;

import java.io.IOException;
import java.util.List;
import android.app.Activity;
import android.location.Address;
import android.location.Geocoder;
import android.os.Bundle;
import android.widget.TextView;
```

```
public class RevGeocodingActivity extends Activity {
   private TextView tv;

   /** Called when the activity is first created. */
   @Override
   public void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.main);

      tv = (TextView) findViewById(R.id.textView);
      List<Address> addresses;
      Geocoder gc = new Geocoder(this);
      try {
         addresses = gc.getFromLocation(37.074301, 22.432584, 1);
         if(addresses != null){
            Address current = addresses.get(0);
            StringBuilder sb = new StringBuilder("Address: \n");
            for(int i = 0; i < current.getMaxAddressLineIndex(); i++){
               sb.append(current.getAddressLine(i)).append("\n");
            }
            tv.setText(sb.toString());
         }
      } catch(IOException ioe) {
         tv.setText(ioe.getMessage());
      }
   }
}
```

Εκτελώντας την εφαρμογή σε μία Android συσκευή θα πάρουμε την ακόλουθη έξοδο:

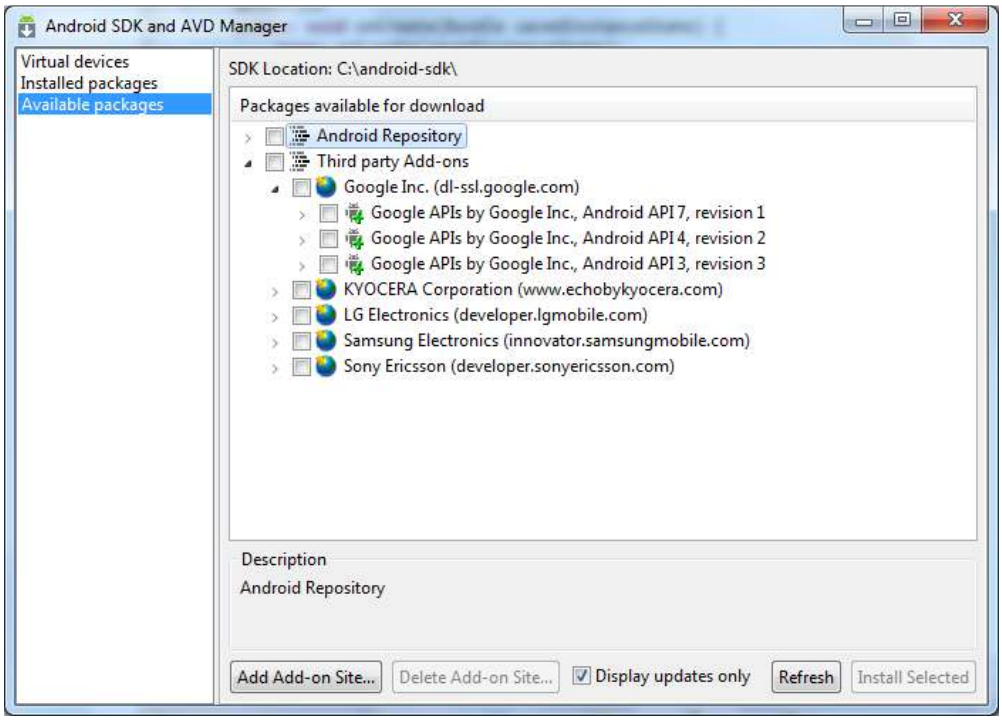
Address:
Χείλωνος 2-14
Sparta 23100

Η μόνη διαφορά της εφαρμογής αυτής από την προηγούμενη είναι πως γίνεται χρήση της μεθόδου `getFromLocation()` της κλάσης `Geocoder`, η οποία λαμβάνει ως παραμέτρους ένα ζεύγος γεωγραφικών συντεταγμένων και τον μέγιστο αριθμό αποτελεσμάτων που επιθυμούμε να επιστρέψει. Η `getFromLocation()` θα πραγματοποιήσει τη μετάφραση και θα επιστρέψει μία λίστα αντικειμένων τύπου `Address`. Στον κώδικα της εφαρμογής που προηγήθηκε η θα επιστραφεί ένα μόλις αποτέλεσμα που όμως μιας και πρόκειται για διεύθυνση μπορεί να αποτελείται από περισσότερες της μιας γραμμές. Η δομή `for` που ακολουθεί έχει ως σκοπό να προβάλλει στο TextView του user interface όλες τις γραμμές από τις οποίες αποτελείται η εν λόγω διεύθυνση.

5.4 Χρήση Google Maps

Η εφαρμογή Maps της Google είναι αναμφισβήτητα μια από τις πιο ενδιαφέρουσες και πιο εντυπωσιακές μαζί με τις Earth, Sky κ.α. Δεδομένου του ότι και η ίδια η πλατφόρμα είναι προϊόν της Google, δε θα μπρούσε παρά να παρέχεται υποστήριξη για τις εφαρμογές αυτές στις συσκευές Android. Αν και το ίδιο το Android SDK δεν παρέχει έμφυτη υποστήριξη για τις εφαρμογές της Google, μπορούμε να κατεβάσουμε και να εγκαταστήσουμε το Google API ως Third Party πακέτο αν δεν το

έχουμε ήδη κάνει μέσω του εργαλείου AVD Manager που χρησιμοποιήσαμε για την εγκατάσταση του πρώτου στην ενότητα 1. Στο σχήμα 5.3 βλέπετε την οθόνη του AVD Manager από όπου μπορούμε να επιλέξουμε να εγκαταστήσουμε τις διαφορετικές εκδόσεις του Google API (υπάρχει ξεχωριστή έκδοση του Google API για κάθε έκδοση του Android).



Σχήμα 5.3

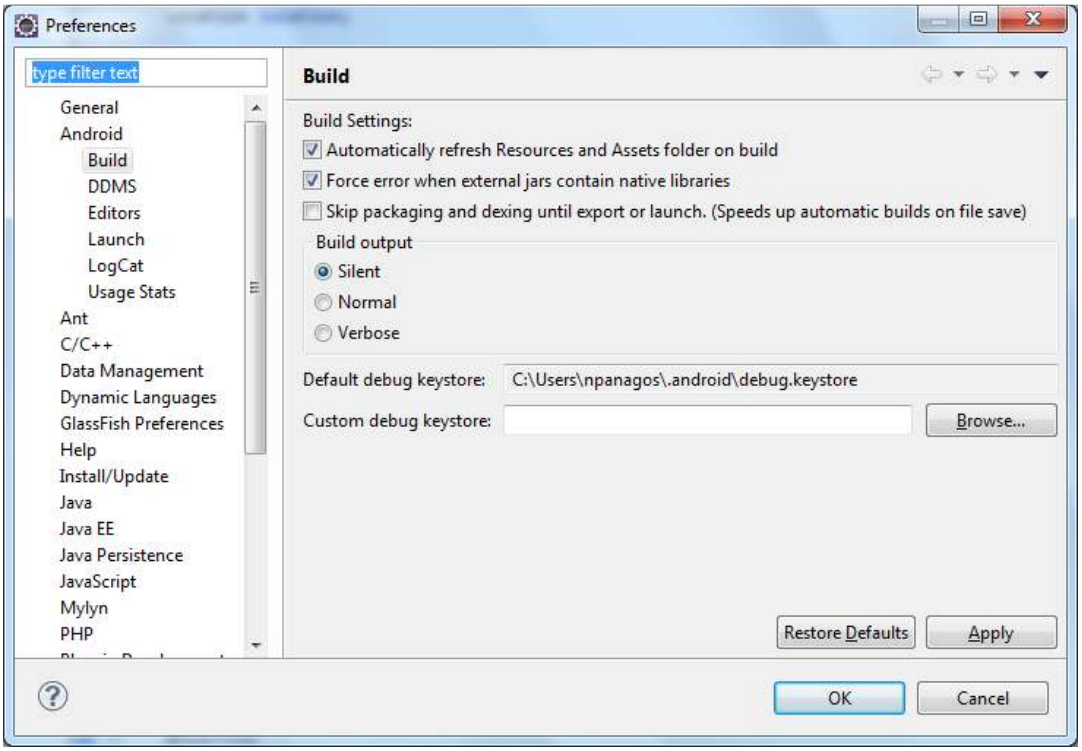
Στον διάλογο του σχήματος βλέπετε ως διαθέσιμα μόνο τα πακέτα για τις εκδόσεις 1.5, 1.6 και 2.1, μιας και τα υπόλοιπα είναι ήδη εγκατεστημένα στον υπολογιστή του γράφοντος. Από τον διάλογο αυτόν επιλέγετε τα πακέτα που θέλετε να εγκαταστήσετε στον υπολογιστή σας, τυπικά όλα από την έκδοση 2.2 και πάνω (API8) και στην συνέχεια κάνετε κλικ στο **Install Selected**.

Μπορούμε να ενσωματώσουμε την τεχνολογία Google Maps στις εφαρμογές μας είτε χρησιμοποιώντας το αντίστοιχο στοιχείο UI που είναι το MapView, είτε χρησιμοποιώντας απ' ευθείας το API παρόλα αυτά, για να λειτουργήσει σωστά το Google Maps API απαιτείται η χρήση ενός έγκυρου κλειδιού (key) το οποίο μπορούμε να προμηθευτούμε από την Google. Η διαδικασία έκδοσης ενός έγκυρου κλειδιού περιγράφεται λεπτομερώς στη σελίδα που βρίσκεται στη διεύθυνση:
http://code.google.com/android/add-ons/google-apis/mapkey.html

Για να ολοκληρωθεί επιτυχώς θα πρέπει πρώτα να δημιουργήσουμε ένα MD5 πιστοποιητικό αποτύπωμα για το κλειδί που χρησιμοποιούμε για την υπογραφή των εφαρμογών μας. Θυμηθείτε πως η ηλεκτρονική υπογραφή των εφαρμογών μας με ένα κλειδί απαιτείται κάθε φορά που κάνουμε export ένα project ώστε να δημιουργηθεί το .apk αρχείο. Η διαδικασία πακεταρίσματος μιας εφαρμογής αναφέρθηκε στην ενότητα 1 και περιγράφεται λεπτομερώς στο κεφάλαιο 14 του βιβλίου σας. Αν λοιπόν έχετε ήδη δημιουργήσει ένα κλειδί, θα πρέπει χρησιμοποιώντας το εργαλείο keytool να παράξετε το MD5 πιστοποιητικό αποτύπωμά του.

Στις γραμμές που ακολουθούν θα περιγραφεί η διαδικασία αυτή για το default κλειδί που έχει δημιουργήσει αυτόματα και χρησιμοποιεί το Eclipse ώστε να μπορούμε να εκτελούμε και να τεστάρουμε τις εφαρμογές μας κατά την υλοποίησή τους στο περιβάλλον. Μπορείτε να βρείτε σε

ποια τοποθεσία βρίσκεται το κλειδί αυτό στον υπολογιστή σας επιλέγοντας **Windows → Preferences** μέσα από το Eclipse και στον διάλογο που θα ανοίξει αναπτύσσοντας την επιλογή **Android** στη λίστα αριστερά και επιλέγοντας **Build**. Στο σχήμα 5.4 και στο πεδίο με ετικέτα **Default debug keystore** βλέπετε τη διαδρομή προς το default κλειδί που χρησιμοποιεί το Eclipse στον υπολογιστή του γράφοντος.



Σχήμα 5.4

Γνωρίζοντας τη διαδρομή προς το default κλειδί, τη χρησιμοποιείτε σε συνδυασμό με το εργαλείο keytool ώστε να δημιουργήσετε το MD5 ηλεκτρονικό αποτύπωμα, όπως φαίνεται στο σχήμα 5.5. Για το default κλειδί όταν σας ζητηθεί να εισάγετε το συνθηματικό, πατάτε απλά Enter.

Σχήμα 5.5

Αυτό που απομένει είναι να χρησιμοποιήσετε το ηλεκτρονικό αποτύπωμα που παράχθηκε για να δημιουργήσετε το κλειδί που θα χρησιμοποιείτε στις εφαρμογές που κάνουν χρήση του Google Maps API μέσω της σελίδας που βρίσκεται στη διεύθυνση:
http://code.google.com/android/maps-api-signup.html
Για να ολοκληρωθεί η διαδικασία και να παραχθεί το κλειδί σας θα πρέπει να έχετε έναν λογαριασμό με την Google, συνεπώς, αν δεν έχετε θα πρέπει να δημιουργήσετε.
Για να δούμε με ποιον τρόπο μπορούμε να κάνουμε χρήση της τεχνολογίας Google Maps, θα αναπτύξουμε μία απλή εφαρμογή η οποία όταν εκτελείται θα χρησιμοποιεί το GPS για να μας υποδείξει την τρέχουσα θέση της συσκευής στον παγκόσμιο χάρτη. Για το συγκεκριμένο project κι επειδή κάνουμε χρήση του Google API θα πρέπει να τσεκάρουμε από τη λίστα **Build Target** του διαλόγου δημιουργίας νέου Android Project την επιλογή **Google APIs** για την πλατφόρμα 2.2 αντί της **Android 2.2** που χρησιμοποιούσαμε μέχρι τώρα.
Στο αρχείο *AndroidManifest.xml* που ακολουθεί έχουν οριστεί όπως μπορείτε να δείτε τα δικαιώματα πρόσβασης `INTERNET` και `ACCESS_FINE_LOCATION`. Το πρώτο είναι απαραίτητο μιας και για να προβληθεί ο χάρτης απαιτείται σύνδεση στο διαδίκτυο, είτε μέσω Wi-Fi είτε μέσω δικτύου 3G, ενώ το δεύτερο απαιτείται για τη χρήση του GPS. Επιπλέον, υπάρχει το tag `<uses-library>` που ενημερώνει την πλατφόρμα πως η εφαρμογή κάνει χρήση τη βιβλιοθήκη `com.google.android.maps` του Google API.

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
      package="elearning.maps"
      android:versionCode="1"
      android:versionName="1.0">
   <uses-sdk android:minSdkVersion="8" />
   <uses-permission android:name="android.permission.INTERNET" />
   <uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
   <application android:icon="@drawable/icon" android:label="@string/app_name">
      <activity android:name=".MapsActivity" android:label="@string/app_name">
         <intent-filter>
            <action android:name="android.intent.action.MAIN" />
            <category android:name="android.intent.category.LAUNCHER" />
         </intent-filter>
      </activity>
   <uses-library android:name="com.google.android.maps" />
   </application>
</manifest>
```

Το user interface της εφαρμογής είναι πολύ απλό και αποτελείται από ένα TextView και ένα MapView:

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
   android:orientation="vertical"
   android:layout_width="fill_parent"
   android:layout_height="fill_parent"
   >
   <TextView android:id="@+id/textView" android:layout_width="fill_parent"
       android:layout_height="wrap_content"
   />
   <com.google.android.maps.MapView android:id="@+id/mapView"
      android:layout_width="fill_parent" android:layout_height="fill_parent"
```

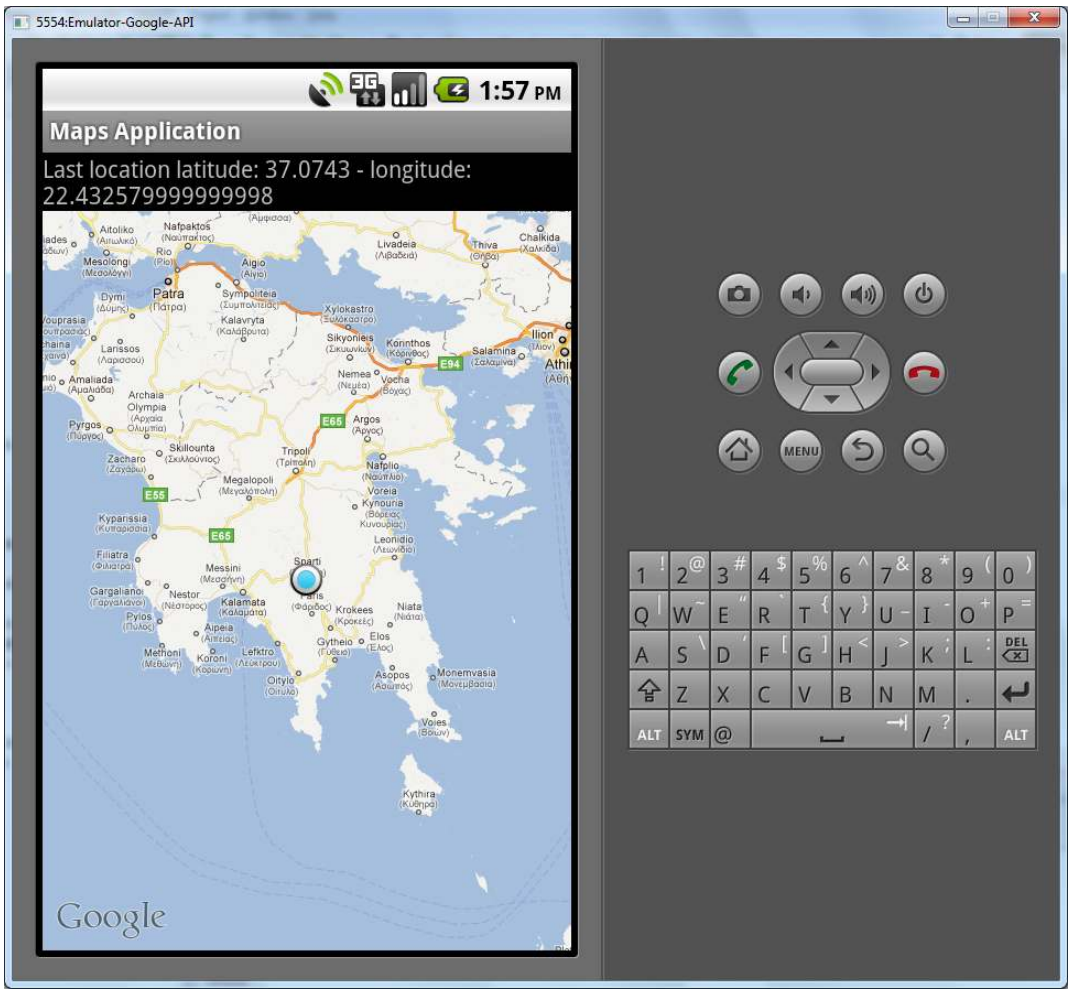
```
      android:clickable="true" android:apiKey="0Cl1FccGJvmUdKZyvtRNmnWH0RvpX-
WN_wLns1w"
   />
</LinearLayout>
```

Στο attribute apiKey του MapView θέτετε ως τιμή το κλειδί που δημιουργήσατε προηγουμένως. Ακολουθεί ο κώδικας του κεντρικού Activity της εφαρμογής:

```
package elearning.maps;

import android.content.Context;
import android.location.Criteria;
import android.location.Location;
import android.location.LocationManager;
import android.os.Bundle;
import android.widget.TextView;
import com.google.android.maps.MapActivity;
import com.google.android.maps.MapController;
import com.google.android.maps.MapView;
import com.google.android.maps.MyLocationOverlay;

public class MapsActivity extends MapActivity {
   private LocationManager lm;
   TextView tv;
   Location location;

   /** Called when the activity is first created. */
   @Override
   public void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.main);

      tv = (TextView) findViewById(R.id.textView);
      MapView mapView = (MapView) findViewById(R.id.mapView);

      mapView.setBuiltInZoomControls(true);
      final MapController controller = mapView.getController();

      lm = (LocationManager) getSystemService(Context.LOCATION_SERVICE);
      Criteria criteria = new Criteria();
      criteria.setAccuracy(Criteria.ACCURACY_FINE);
      criteria.setPowerRequirement(Criteria.POWER_LOW);
      String provider = lm.getBestProvider(criteria, true);
      location = lm.getLastKnownLocation(provider);
      final MyLocationOverlay myLoc = new MyLocationOverlay(this, mapView);
      myLoc.enableMyLocation();
      myLoc.runOnFirstFix(new Runnable() {
         public void run(){
            controller.setZoom(9);
            controller.animateTo(myLoc.getMyLocation());
         }
      });
      mapView.getOverlays().add(myLoc);
      tv.setText("Last location latitude: " + location.getLatitude() +
                     " - longitude: " + location.getLongitude());
   }

   @Override
   protected boolean isRouteDisplayed() {
```

```
        return false;
    }
}
```

Το πρώτο πράγμα που θα πρέπει να παρατηρήσετε στον κώδικα είναι πως το Activity της εφαρμογής μας κλάση επεκτείνει την κλάση `MapActivity` αντί της `Activity`. Πρόκειται για μια αφηρημένη κλάση που έχει υλοποιηθεί ώστε να χειρίζεται τα διάφορα θέματα που σχετίζονται με την προβολή ενός MapView. Είναι υποχρεωτική η υλοποίηση της μεθόδου `isRouteDisplayed()` η οποία σύμφωνα με την Google θα πρέπει να υποδεικνύει αν γίνεται χρήση του API για την προβολή διαδρομών οπότε τη θέτουμε να επιστρέφει `true` η `false` ανάλογα με την περίπτωση.
Κατά τα άλλα, ο κώδικας της εφαρμογής είναι παρόμοιος με αυτόν της πρώτης που υλοποιήθηκε στην τρέχουσα ενότητα και που μας επέστρεφε την τελευταία θέση της συσκευής με την επιπλέον προσθήκη ενός συστατικού MapView. Έτσι λοιπόν, αφότου αρχικοποιηθεί η μεταβλητή-μέλος τύπου `LocationManager` με έναν πάροχο εντοπισμού θέσης και που βάσει των κριτηρίων που έχουμε θέσει θα είναι το GPS, δηλώνεται μία αναφορά τύπου `MapController` η οποία αρχικοποιείται με τον controller του MapView της εφαρμογής μας μέσω της κλήσης της `getController()`. Η κλάση `MapController` χρησιμοποιείται για τον έλεγχο του χάρτη παρέχοντας μεθόδους για την εκτέλεση ενεργειών όπως το zoom-in και zoom-out, η προβολή συγκεκριμένης τοποθεσίας κλπ.
Μία από τις πολλές δυνατότητες που προσφέρει το Google Maps API είναι αυτή της προσθήκης και προβολής δεικτών (markers) που υποδεικνύουν σημεία ενδιαφέροντος μαζί με πληροφορίες. Οι δείκτες αυτοί τοποθετούνται σε εικονικά στρώματα (overlays) που επικάθονται επάνω στο MapView. Εκτός από τα custom overlays που μπορεί να δημιουργήσει ο προγραμματιστής, το Google API περιλαμβάνει και ένα πολύ χρήσιμο προκαθορισμένο overlay, το `MyLocationOverlay`, το οποίο προβάλλει αυτόματα την τρέχουσα τοποθεσία της συσκευής στο χάρτη ως μια μπλε κουκίδα. Παράλληλα μπορεί να εμφανίσει στοιχεία σχετικά με την ακρίβεια της θέσης καθώς και την πορεία του χρήστη.
Στην εφαρμογή μας δημιουργείται ένα αντικείμενο τύπου `MyLocationOverlay` που αρχικοποιείται να χρησιμοποιηθεί σε συνδυασμό με το MapView της εφαρμογής μας. Η κλάση `MyLocationOverlay` παρέχει αρκετές χρήσιμες μεθόδους, όπως για παράδειγμα η `enableMyLocation()` η οποία επιστρέφει ένα αντικείμενο τύπου `GeoPoint` με τις συντεταγμένες της τρέχουσας θέσης της συσκευής. Η `runOnFirstFix()` δημιουργεί ένα νέο νήμα, το σώμα του οποίου θα εκελεστεί αμέσως όταν εντοπιστεί η τρέχουσα θέση της συσκευής. Στο παράδειγμά μας θέτουμε το επίπεδο ζουμ στο 9 για το MapView με την `setZoom()` και κεντράρουμε τον χάρτη ώστε να προβάλλεται η γεωγραφική περιοχή που περιβάλλει την τρέχουσα τοποθεσία της συσκευής. Στις τελευταίες γραμμές κώδικα προστίθεται το αντικείμενο `MyLocationOverlay` στη λίστα με τα overlays του MapView μας και προβάλλονται οι συντεταγμένες της τρέχουσας θέσης στο TextView.
Για να εκτελέσετε την εφαρμογή στον emulator θα πρέπει αυτός να έχει οριστεί να υποστηρίζει το Google API, διαφορετικά η εκτέλεση θα αποτύχει. Μπορείτε να δημιουργήσετε μία νέα εικονική συσκευή με τον τρόπο που περιγράφηκε στην ενότητα 1, επιλέγοντας όμως από τη λίστα με ετικέτα **Target** την τιμή **Google APIs (Google Inc.) - API Level 8**. Εκτελώντας την εφαρμογή στον emulator και έχοντας στείλει ένα εικονικό ζεύγος γεωγραφικών συντεταγμένων μέσω του Emulator Control θα πάρουμε έξοδο σαν αυτήν του σχήματος 5.6. Όπως μπορείτε να δείτε από το σχήμα, δόθηκαν οι συντεταγμένες που φαίνονται στο TextView που βρίσκεται στο επάνω τμήμα της οθόνης και που αντιστοιχούν στην πόλη της Σπάρτης, οπότε ο χάρτης έχει κεντραριστεί έχοντας ως σημείο αναφοράς την εικονική τρέχουσα θέση της συσκευής και το επίπεδο ζουμ όπως έχει οριστεί στην εφαρμογή. Η εικονική τρέχουσα θέση σημειώνεται επάνω στον χάρτη με μία μπλε κουκίδα.



Σχήμα 5.6

Μπορούμε επίσης να εκτελέσουμε την εφαρμογή σε μία πραγματική συσκευή με τον τρόπο που περιγράφεται στην ενότητα 7, οπότε θα μαρκαριστεί στον χάρτη η πραγματική τρέχουσα θέση της συσκευής σας. Φυσικά, για να λειτουργήσει σωστά η εφαρμογή θα πρέπει να μη βρίσκεστε σε κλειστό χώρο ώστε να μπορεί να γίνει λήψη σήματος από δορυφόρο και η συσκευή σας να έχει πρόσβαση στο internet είτε μέσω Wi-Fi είτε μέσω 3G. Την ίδια ακριβώς λειτουργία βέβαια προσφέρει και η εφαρμογή Google Maps εξ' ορισμού.

Οι υπηρεσίες βάσει τοποθεσίας αποτελούν μία άκρως ενδιαφέρουσα δυνατότητα για τις εφαρμογές μας και στην ενότητα αυτή εξετάσαμε κάποια από τα πιο βασικά χαρακτηριστικά τους. Δεδομένου του ότι βρισκόμαστε σε εισαγωγικό επίπεδο, δεν συνυπολογίστηκαν παράγοντες που αυξάνουν την πολυπλοκότητά τους και που σαφώς θα πρέπει να λαμβάνονται υπ' όψιν κατά την υλοποίηση ολοκληρωμένων εφαρμογών. Ένας τέτοιος παράγοντας για παράδειγμα είναι η κατανάλωση ενέργειας, η οποία θα πρέπει να είναι όσο το δυνατόν χαμηλότερη. Εφαρμογές που χρησιμοποιούν λειτουργίες εντοπισμού θέσης όπως αυτές που υλοποιήθηκαν και δη αυτές που ενημερώνουν για την τρέχουσα θέση ανά τακτά χρονικά διαστήματα είναι ενεργοβόρες και θα πρέπει ο προγραμματιστής να προσπαθεί όποτε του δίνεται η ευκαιρία να ελαττώνει την κατανάλωση. Αποτελεί λοιπόν κοινή πρακτική να αποδεσμεύουμε τους πόρους που εμπλέκονται στον εντοπισμό της θέσης όταν η εφαρμογή μας μεταβαίνει στο background και να τους επαναδεσμεύουμε όταν η εφαρμογή επιστρέφει στο προσκήνιο υπερκαλύπτοντας τις `onPause()` και `onResume()` με τον κατάλληλο κώδικα.

5.5 Φωτογραφική Μηχανή

Η φωτογραφική μηχανή αποτελεί τον πιο προφανή αισθητήρα σε μία συσκευή Android και πιθανώς αυτόν με την περισσότερη χρήση. Για αρκετούς μάλιστα υποψήφιους αγοραστές, τα τεχνικά χαρακτηριστικά της φωτογραφικής μηχανής παίζουν πρωταρχικό ρόλο κατά την επιλογή συσκευής, ενώ οι προδιαγραφές τους βελτιώνονται διαρκώς σε κάθε νέα γενιά συσκευών.
Μπορούμε να χρησιμοποιήσουμε τη φωτογραφική μηχανή μέσα από τις εφαρμογές μας με δύο τρόπους, μέσω του προκαθορισμένου Intent ή προγραμματιστικά μέσω του Android SDK. Ο πρώτος είναι σαφώς πιο εύκολος και θα σας καλύψει στις περισσότερες των περιπτώσεων, ενώ ο δεύτερος είναι αρκετά πιο πολύπλοκος και γίνεται μέσω της κλάσης `android.hardware.Camera` σε συνδυασμό με την `android.view.SurfaceView` που προβάλλει μία προεπισκόπηση στην οθόνη της συσκευής. Και στις δύο περιπτώσεις για να αποκτήσει πρόσβαση η εφαρμογή μας στη φωτογραφική μηχανή απαιτούνται τα κατάλληλα δικαιώματα (`android.permission.CAMERA`).
Η υποτυπώδης εφαρμογή που ακολουθεί κάνει χρήση της πρώτης μεθόδου, δηλαδή του προκαθορισμένου Intent. Το user interface της αποτελείται από ένα μόνο κουμπί, που όταν πατηθεί από τον χρήστη θα ενεργοποιηθεί η λειτουργία της φωτογραφική μηχανής και ο χρήστης θα μεταβεί στην αντίστοιχη οθόνη. Το XML κείμενο του UI είναι το εξής:

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
   android:orientation="vertical"
   android:layout_width="fill_parent"
   android:layout_height="fill_parent"
   android:gravity="center">
   <Button android:id="@+id/button" android:layout_width="wrap_content"
      android:layout_height="wrap_content" android:text="Take photo!"/>
</LinearLayout>
```

Ο κώδικας του κεντρικού Activity της εφαρμογής μας είναι ο ακόλουθος:

```
package elearning.camera;

import android.app.Activity;
import android.content.Intent;
import android.os.Bundle;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;

public class CameraActivity extends Activity {
   /** Called when the activity is first created. */
   @Override
   public void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.main);

      Button button = (Button) findViewById(R.id.button);
      button.setOnClickListener(new OnClickListener() {
         @Override
         public void onClick(View view) {
            takePhoto();
```

```
        }
      });
   }

   public void takePhoto(){
      Intent intent = new Intent("android.media.action.IMAGE_CAPTURE");
      startActivity(intent);
   }
}
```

Δεδομένου του ότι ο emulator προς το παρόν δεν παρέχει προεπισκόπηση, είναι προτιμότερο να εκτελέσετε και να ελέγξετε την έξοδο της εφαρμογής σε μία πραγματική συσκευή είτε κάνοντάς την export και εγκαθιστώντας τη, είτε μέσα από το περιβάλλον του Eclipse όπως περιγράφεται στην ενότητα 7.

5.6 Αισθητήρες

Η βελτίωση και εξάπλωση των μικρών σε μέγεθος και χαμηλής ενεργειακής κατανάλωσης ηλεκτρονικών τύπου MEMS (Micro-Electro-Mechanical Systems) είναι ολοένα μεγαλύτερη τα τελευταία χρόνια. Ένα τυπικό smartphone τύπου Android για παράδειγμα περιλαμβάνει τουλάχιστον ένα ζευγάρι αισθητήρων της τεχνολογίας αυτής, ενώ πιο ακριβές συσκευές χρησιμοποιούν περισσότερους και πιο εξεζητημένους. Αυτό με τη σειρά του αποτελεί ένα θέμα που θα πρέπει να λαμβάνεται υπ' όψιν κατά την υλοποίηση εφαρμογών που εκμεταλλεύονται κάποιον αισθητήρα μιας και ο συγκεκριμένος είναι πιθανόν να μην είναι διαθέσιμος σε όλες τις συσκευές.
Αυτοί που θεωρούνται ως στάνταρντ είναι το επιταχυνσιόμετρο τριών αξόνων (three-axis accelerometer) που προσδιορίζει την κλίση της συσκευής και το μαγνητόμετρο τριών αξόνων (three-axis magnetometer) για τον προσδιορισμό του προσανατολισμού. Εκτός από αυτούς, σε πιο ακριβές συσκευές θα βρείτε και πιο "εξωτικούς" αισθητήρες όπως μέτρησης θερμοκρασίας (temperature sensor), προσδιορισμού εγγύτητας (proximity sensor), μέτρησης φωτός περιβάλλοντος (light sensor), μέτρησης ατμοσφαιρικής πίεσης (pressure sensor), και γυροσκοπίου (gyroscope) κ.α. Κάθε ένας από τους αισθητήρες αυτούς περιγράφεται από την αντίστοιχη σταθερά, που για την τελευταία έκδοση Android κατά τη σύνταξη των σημειώσεων συνοψίζονται στον πίνακα 5.1.

Τύπος αισθητήρα	Περιγραφή
TYPE_ACCELEROMETER	Μετράει την επιτάχυνση σε m/sec^2
TYPE_ALL	Περιγράφει όλους τους αισθητήρες
TYPE_GYROSCOPE	Γυροσκόπιο που μετράει το ρυθμό περιστροφής από τους άξονες X, Y και Z σε rad/sec.
TYPE_LIGHT	Μετράει το φως της ατμόσφαιρας σε lux
TYPE_MAGNETIC_FIELD	Μετράει το μαγνητικό πεδίο σε micro-Tesla
TYPE_PRESSURE	Μετράει την ατμοσφαιρική πίεση
TYPE_PROXIMITY	Μετράει την απόσταση από ένα αντικείμενο σε εκατοστά
TYPE_TEMPERATURE	Μετράει τη θερμοκρασία σε βαθμούς Κελσίου
TYPE_GRAVITY	Μετράει την κατεύθυνση και την τιμή της βαρύτητας σε 3 άξονες. Η μονάδα μέτρησης είναι τα m/sec^2.
TYPE_LINEAR_ACCELERATION	Μετράει την επιτάχυνση στους τρεις άξονες σε

	m/sec^2 χωρίς να συνυπολογίζει την επιτάχυνση βαρύτητας.
TYPE_ROTATION_VECTOR	Αναπαριστά τον προσανατολισμό της συσκευής ως συνδυασμό γωνίας με άξονα.

Πίνακας 5.1

Η πλατφόρμα μας δίνει τη δυνατότητα να χρησιμοποιήσουμε όλους τους υποστηριζόμενους αισθητήρες και να κάνουμε πιο ενδιαφέρουσες τις εφαρμογές μας απευθείας μέσω του SDK, χωρίς να απαιτούνται δικαιώματα πρόσβασης. Η βασική κλάση του SDK που διαχειρίζεται όλους τους αισθητήρες μιας συσκευής είναι η `SensorManager`. Επίσης, παρέχει μια σειρά listeners οι οποίοι χρησιμοποιούνται για να τροφοδοτήσουν τις εφαρμογές μας με τιμές που προέρχονται απευθείας από κάποιον αισθητήρα χρησιμοποιώντας πρωτίστως την callback μέθοδο `onSensorChanged()` και ενδεχομένως την `onAccuracyChanged()`. Για να ελέγξουμε με ποιους αισθητήρες είναι εφοδιασμένη μία συσκευή μπορούμε να χρησιμοποιήσουμε τη μέθοδο `getSensorList()` η οποία επιστρέφει μία λίστα με τους υποστηριζόμενους αισθητήρες. Η συγκεκριμένη μέθοδος χρησιμοποιείται στην εφαρμογή που ακολουθεί όπου επιπλέον γίνεται παρακολούθηση των μετρήσεων του αισθητήρα φωτός της συσκευής, οι οποίες και προβάλλονται στην οθόνη ανά τακτά χρονικά διαστήματα.
Το UI της εφαρμογής αποτελείται από ένα μόνο TextView που καλύπτει ολόκληρη την οθόνη και προβάλλει τις πληροφορίες. Ο κώδικας του κεντρικού Activity είναι ο ακόλουθος:

```
package elearning.metering;

import android.app.Activity;
import android.hardware.Sensor;
import android.hardware.SensorEvent;
import android.hardware.SensorEventListener;
import android.hardware.SensorManager;
import android.os.Bundle;
import android.widget.TextView;
import java.util.List;

public class LightMeterActivity extends Activity implements SensorEventListener {
   private SensorManager sm;
   private List<Sensor> sensors;
   private TextView tv;
   private StringBuilder sensorNames;

   /** Called when the activity is first created. */
   @Override
   public void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.main);

      tv = (TextView) findViewById(R.id.textView);
      sm = (SensorManager) getSystemService(SENSOR_SERVICE);
      sensors = sm.getSensorList(Sensor.TYPE_ALL);
      sensorNames = new StringBuilder("Supported sensors: \n");
      for (Sensor s : sensors) {
         sensorNames.append(s.getName());
         sensorNames.append("\n");
      }
   }
```

```
    @Override
    public void onAccuracyChanged(Sensor sensor, int acc) { }

    @Override
    public void onSensorChanged(SensorEvent se) {
        if(se.sensor.getType() == Sensor.TYPE_LIGHT)
            tv.setText(sensorNames + "\n\nLight:\n" + se.values[0] + " lux");
    }

    @Override
    protected void onPause() {
        super.onPause();
        sm.unregisterListener(this);
    }

    @Override
    protected void onResume() {
        super.onResume();
        sm.registerListener(LightMeterActivity.this,
                    sm.getDefaultSensor(Sensor.TYPE_LIGHT),
                    SensorManager.SENSOR_DELAY_GAME);
      }
}
```

Το Activity της εφαρμογής μας έχει τεθεί να υλοποιεί το interface `SensorEventListener` το οποίο περιέχει τις προαναφερθείσες callback μεθόδους `onSensorChanged()` και `onAccuracyChanged()`. Η πρώτη καλείται αυτόματα όταν ο αισθητήρας λάβει μία νέα μέτρηση ενώ η δεύτερη όταν υπάρξει κάποια μεταβολή στην ακρίβεια του μετρίσιμου μεγέθους. Αρχικά λαμβάνεται το υπάρχον στιγμιότυπο της υπηρεσίας διαχείρισης αισθητήρων και εκχωρείται στη μεταβλητή-μέλος τύπου `SensorManager`. Καλώντας στη συνέχεια τη μέθοδο `getSensorList()` θα επιστραφεί η λίστα με τους αισθητήρες με τους οποίους είναι εφοδιασμένη η συσκευή και θα αποθηκευτεί στη μεταβλητή-μέλος `sensors`. Με μία δομή for προσπελαύνονται ένας ένας οι αισθητήρες της λίστας και δημιουργείται ένα λεκτικό που αποτελείται από τα ονόματά τους, το οποίο αποθηκεύεται στη μεταβλητή-μέλος τύπου `StringBuilder`. Το λεκτικό αυτό θα χρησιμοποιηθεί στη συνέχεια για να προβληθεί στην οθόνη.

Από τις δύο callback μεθόδους, η μέθοδος `onAccuracyChanged()` περιέχει κενή υλοποίηση ενώ αυτή που μας ενδιαφέρει είναι η `onSensorChanged()` η οποία περιέχει τον κώδικα που θα εκτελείται κάθε φορά που θα έχουμε νέα μέτρηση. Αρχικά ελέγχεται αν η νέα μέτρηση προέρχεται από τον συγκεκριμένο αισθητήρα και αν ισχύει αυτό, τίθεται ως λεκτικό προς προβολή στο TextView η λίστα των αισθητήρων της συσκευής ακολουθούμενη από την νέα τιμή του αισθητήρα φωτός. Αν προσπαθήσετε να εκτελέσετε την εφαρμογή στον emulator θα πάρετε ως έξοδο μία κενή οθόνη μιας και ο συγκεκριμένος αισθητήρας όχι μόνο δεν προσομοιώνεται αλλά δεν υποστηρίζεται καν. Εκτελώντας την σε μία πραγματική συσκευή που διαθέτει τέτοιον αισθητήρα, θα πάρετε έξοδο σαν την ακόλουθη:

Supported sensors:
AK8973 Compass
AK8973 Compass RAW
Lis302dl
AK8973 Magnetic Field
Ambient Light

proximity

Light:
32.0 lux

Η συγκεκριμένη έξοδος προήλθε από την εκτέλεση της εφαρμογής σε μία συσκευή ZTE Blade. Όπως μπορείτε να δείτε η συσκευή διαθέτει ένα μαγνητόμετρο (η εγγραφή AK8973 που επαναλαμβάνεται 3 φορές), ένα επιταχυνσιόμετρο (Lis302dl), έναν αισθητήρα φωτός (Ambient Light) και τέλος έναν αισθητήρα προσδιορισμού εγγύτητας (proximity). Ακολουθεί η μέτρηση φωτός που κατέγραψε ο αισθητήρας τη δεδομένη χρονική στιγμή.
Η χρήση των αισθητήρων μπορεί να κάνει τις εφαρμογές σας πολύ πιο ενδιαφέρουσες και σε πολλές περιπτώσεις πρωτότυπες και ίσως διασκεδαστικές, παρόλα αυτά θα πρέπει να είσαστε ιδιαίτερα προσεκτικοί κατά τη χρήση τους, μιας και υπάρχουν πολλά θέματα που θα πρέπει να λαμβάνονται υπ' όψιν. Ένα από αυτά που έχει ήδη αναφερθεί είναι το γεγονός πως δεν υποστηρίζονται όλων των ειδών οι αισθητήρες από όλες τις συσκευές. Έτσι λοιπόν, θα πρέπει να γνωρίζετε εκ των προτέρων πως η εφαρμογή σας που χρησιμοποιεί κάποιον από αυτούς δεν θα προορίζεται για το σύνολο των συσκευών Android. Στις περιπτώσεις αυτές καλό θα είναι να αναφέρεται ρητά στην περιγραφή της εφαρμογής ώστε οι χρήστες να είναι ενήμεροι.
Ένα άλλο σημαντικό θέμα έχει να κάνει με την έμφυτη δυσκολία στην κατανόηση της λειτουργίας ορισμένων εξ' αυτών που οφείλεται στις γνώσεις Φυσικής που απαιτούνται και που ως συνέπεια έχουν την αυξημένη πιθανότητα σύνταξης κώδικα που περιέχει bugs. Πριν προχωρήσετε στην υλοποίηση εφαρμογής που χρησιμοποιεί έναν τέτοιον αισθητήρα θα πρέπει να είστε σίγουροι πως γνωρίζετε ακριβώς τη λειτουργία του και τον τρόπο με τον οποίο μετράται το συγκεκριμένο μέγεθος ή που υπολογίζεται ένα δεύτερο μέσω κάποιου φυσικού τύπου. Η τεκμηρίωση του Android SDK περιγράφει λεπτομερώς τη λειτουργία όλων σχεδόν των αισθητήρων, σε κάθε περίπτωση όμως θα πρέπει να κάνετε εξαντλητικά τεστ στις εφαρμογές σας πριν τις διαθέσετε στο Android Market.
Τέλος, όπως και στην περίπτωση του GPS που αναφέρθηκε προηγουμένως, έτσι και οι λοιποί αισθητήρες καταναλώνουν αρκετή ενέργεια και θα πρέπει να χρησιμοποιούνται με φειδώ. Στην εφαρμογή που προηγήθηκε υλοποιείται η οδηγία που αποτελεί κανόνα σωστής πρακτικής κατά τη χρήση αισθητήρων η του GPS στις εφαρμογές και που αφορά στη δέσμευσή τους όταν η εφαρμογή βρίσκεται στο προσκήνιο και την αποδέσμευσή τους όταν η εφαρμογή μεταβαίνει στο παρασκήνιο. Όπως μπορείτε να δείτε από τον κώδικα, στην `onResume()` έχει τοποθετηθεί ο κώδικας που δεσμεύει τον αισθητήρα κάνοντας register τον listener ενώ τον αποδεσμεύει μέσα στο σώμα της `onPause()` καλώντας την `unregisterListener()`. Κατά την εγγραφή του listener στην κλήση της `registerListener()` ορίζουμε μέσω της τρίτης παραμέτρου τον επιθυμητό ρυθμό ανανέωσης των μετρήσεων, ο οποίος μπορεί να λάβει τις εξής 4 τιμές:

- `SENSOR_DELAY_FASTEST`: Ο κατά το δυνατόν πιο ταχύς ρυθμός ανανέωσης μετρήσεων που μπορεί να είναι από 8ms έως περίπου 30ms ανάλογα με τη συσκευή
- `SENSOR_DELAY_GAME`: Ρυθμός ανανέωσης κατάλληλος για παιχνίδια, ο οποίος κυμαίνεται γύρω στα 40ms
- `SENSOR_DELAY_NORMAL`: Ο εξ' ορισμού ρυθμός ανανέωσης. Κυμαίνεται στο επίπεδο των 200 περίπου ms που είναι κατάλληλος για μεταβολές στον προσανατολισμό της οθόνης
- `SENSOR_DELAY_UI`: Ρυθμός ανανέωσης κατάλληλος για μεταβολές στο user interface της εφαρμογής (περίπου 350ms)

5.7 Μηχανισμός Δόνησης

Το τελευταίο θέμα που θα μας απασχολήσει στην ενότητα αυτή είναι η χρήση του μηχανισμού δόνησης. Η δόνηση αποτελεί βασικό χαρακτηριστικό όλων των συσκευών κινητής τηλεφωνίας και οι συσκευές Android δεν αποτελούν εξαίρεση. Η πλατφόρμα μας επιτρέπει να χρησιμοποιήσουμε τον μηχανισμό δόνησης κάνοντας χρήση της κλάσης `Vibrator` η οποία βρίσκεται μέσα στο πακέτο `android.os`. Τα απαιτούμενα δικαιώματα πρόσβασης που θα πρέπει να έχει μία εφαρμογή για να κάνει χρήση της συγκεκριμένης λειτουργίας ορίζονται από τη σταθερά `VIBRATE`.
Κάνοντας χρήση της κλάσης `Vibrator` και δηλώνοντας μία αναφορά, μπορούμε να λάβουμε το υπάρχον στιγμιότυπο της υπηρεσίας δόνησης ως εξής:

```
Vibrator vib = (Vibrator) getSystemService(Context.VIBRATOR_SERVICE);
```

Καλώντας στη συνέχεια τη μέθοδο `vibrate()` της `Vibrator`, η συσκευή θα αρχίσει να δονείται, π.χ.:

```
vib.vibrate(5000);
```

Η `vibrate()` λαμβάνει ως παράμετρο τη διάρκεια δόνησης σε milliseconds, άρα η παραπάνω γραμμή θα έχει ως αποτέλεσμα η συσκευή να δονηθεί για 5 δευτερόλεπτα. Μπορούμε αν επιθυμούμε να διακόψουμε τη δόνηση πριν αυτή ολοκληρωθεί καλώντας απλά τη μέθοδο `cancel()`.
Τέλος, είναι δυνατόν να ορίσουμε ρυθμική δόνηση της συσκευής ορίζοντας μία αλληλουχία που αποτελείται από δονήσεις και παύσεις, όπως φαίνεται στο ακόλουθο παράδειγμα:

```
long[] pattern = {1000, 2000, 5000};
vib.vibrate(pattern, 1);
```

Οι παραπάνω γραμμές θα έχουν ως αποτέλεσμα η συσκευή να δονείται με μία ρυθμική ακολουθία που αποτελείται από μία παύση ενός δευτερολέπτου, μία δόνηση 2 δευτερολέπτων και μία παύση 5 δευτερολέπτων. Η έκδοση της `vibrate()` του παραδείγματος λαμβάνει ως παράμετρο μία ρυθμική ακολουθία με τη μορφή ενός πίνακα τύπου `long`, ενώ η δεύτερη παράμετρος ορίζει το αν η ακολουθία αυτή θα επαναλαμβάνεται ή όχι. Σε περίπτωση που δεν επιθυμούσαμε την επανάληψη της ακολουθίας θα έπρεπε να περάσουμε ως δεύτερη παράμετρο την τιμή -1.