Location Based Services and the Location API

F. Ricci
2010/2011
Content

- Location-based services
- Context-based service
- What is a location
- How to obtain location data
  - Satellites
  - Cellular network
  - Short range positioning beacons
- JSR 179 – Location API
  - LocationProvider
  - ProximityListener
  - LocationListener
  - Landmark and LandmarkStore
What location-based services do?

- **Location Based Services (LBS) answer three questions:**
  - Where am I?
  - What’s around me?
  - How do I get there?
Where am I?

- I’m in Walter square; I’m in room 308; I’m near exit 203 ... all these location information need:
  - Localization Services
  - They determine the location of the user using one among several positioning technologies
    - GPS; Wi-Fi; Cellular Network; RFID
  - Position is combined with other information (context) to provide personalized applications and services
    - E.g.: Fleet Management Service
      - The current location of the transport means of a fleet are collected from the GPS installed in the means (trucks) and sent to the headquarters to show the position of all the means.
What is around me?

- Service enabling to search for point of interests (hotel, restaurants, bus stations, etc) that are around the user

- **Search by categories:**

  - All categories
  - Apparel - Boys' & Children's
  - Apparel - Retail
  - Apparel - Women's
  - Bags & Handbags - Retail
  - Book Dealers - New

  ![Search Results](image)
What is around me?

- A list of POIs (Places of Interest) and a compass to help you to find them.
What is around me?

- Layers over imposed to the reality or to a map representation

- Reality
- List
- Map
How do I get there?

- In **car**, on a **motorbike**, **trekking**
- Provide **maps** with large spatial coverage
- Can also **play music** and **store data**
- Provide **voice-based** directions
- **Touchscreen**
- **Integrated** with mobile phone, iPod or a PC
- Can store **waypoints** and **itineraries**
- Show **points of interest** (POI)
- Can also show other information s.a. your heart frequency.
Location-Based vs. Localization

- **Location-based Services**
  - *IT services that generate, compile, select, or filter information or perform other actions by taking into consideration the locations of one or several target persons or mobile objects*

- **Localization services**
  - ... deal with the localization of target persons or objects
  - ... make location data (e.g., WGS-84 coordinates obtained by GPS) available to external actors
  - ... do not imply the processing of location data for composing information or performing high-level functions
  - ... are not end-user services
  - ... are important **sub-services** of LBSs
Location-Based Services

- LBSs as a special appearance of context-aware services
- **Context-aware services**
  - Services that automatically adapt to one or several parameters (context information) reflecting the context of a target
  - **Primary context**
    - Any kind of raw data derived from sensors
    - Sensors: light sensors, bio sensors, microphones, accelerometers, location sensors
  - **Secondary context**
    - High-level context derived from raw data by combination, selection, filtering....
    - Example: state of a person (sleeping, working, eating,...)
Contextual data inference and usage

- Contextual information is normally used to infer other (contextual) information that is not directly observables and the to support services.

**Acquisition of Contextual Data**

- **Explicit**
  - Contextual data entered by users

- **Implicit**
  - Contextual data automatically captured by sensors or other devices

**Inference**

- Context is inferred or estimated from other available contextual data.

**Application of Contextual Data**

- **Presentation** of information and services
- **Automatic execution** of a service
- **Tagging** of context to information for later retrieval
Example

- Hospital scenario

**ACQUISITION OF CONTEXTUAL DATA**

- **Explicit**
  Personal status as specified by the doctor on his mobile phone.

- **Implicit**
  Location data captured using RFID.

**APPLICATION OF CONTEXTUAL DATA**

- **Presentation**
  Doctor’s colleagues are informed about his activity, location and personal status.

- **Execution**
  Doctor’s mobile phone’s profile is changed automatically depending on his activity.

- **Tagging**
  Doctor’s colleagues can tag text messages so that the doctor receives them only when he engages in a particular activity.
Fundamental positioning data

- **Latitude** (angular distance from Equator)
- **Longitude** (angular distance from Greenwich meridian)
- **Altitude** (above the sea level)
- **Orientation** (angular distance from the north pole)
Google Maps Labs

Google Maps Labs is a testing ground for experimental features that aren't quite ready for prime time. They may change, break or disappear at any time.

If such a feature breaks, and you're having trouble loading Maps, use this escape hatch: maps?tfr=0.

**Distance Measurement Tool**
Adam S, Andrey S, James M, Seth L

Measure the distance of a path on the Earth

**Short URL**
Ben A, Holly D

Shorten the Maps "Link" to be a more compact URL.

**Show Me Here!**
Cornelius Q

Add an option to the context menu that lets you zoom directly to the maximum zoom level at the point under the cursor.

**Drag 'n' Zoom**
Dave D

Zooming in on a specific part of the map is now fast and easy. Simply click the Drag 'n' Zoom button, draw a box on the map.

bolzano
WGS84 (World Geodetic System 1984)

- The latitude range is between +90.0 and -90.0
- The longitude range is between +180.0 and -180.0
- The altitude is expressed in meters and represents the altitude of the point from the ellipsoid defined in the WGS84 standard

http://en.wikipedia.org/wiki/Geoid
Positioning Infrastructure

- Distributed infrastructure for implementing positioning
  - Terminal attached to the target person
  - Reference stations at well-known coordinates
  - Control units for coordinating positioning
Position from the Mobile Network

- **Cell Broadcast Service**, or CBS is a carrier version of SMS - it enables a cell phone operator to broadcast messages to a group of cell phone users – e.g. cell info on channel 50 in Italy

- **The current cell ID** - can be used to identify the Base Transceiver Station (BTS) that the device is communicating with and the location of that BTS

- The accuracy of this method depends on the size of the cell – inaccurate as a GSM cell may be anywhere from 2 to 20 kilometers in diameter

- Other techniques are based on **triangularization** used along with cell ID can achieve accuracy within 150 meters
Network Based Solutions

The MS emits a signal and the antennas get it at different times.

Current Accuracy = 200m - 2km
Future Accuracy = 100m - 500m
Location Technologies Network-based

- UL-TOA (Uplink Time of Arrival)
- \( c = 299 \, 792 \, 458 \, \text{m/s} \)
Location Technologies Network-based

- AOA (Angle of Arrival)
Using Satellites

- The **Global Positioning System** (GPS), controlled by the US Department of Defense, uses a constellation of 24 satellites orbiting the earth.

- GPS determines the device's position by calculating **differences in the times signals from different satellites** take to reach the receiver.

- GPS signals are **encoded**, so the mobile device must be equipped with a **GPS receiver**.

- GPS is potentially the **most accurate method** (between 4 and 40 meters if the GPS receiver has a clear view of the sky).

- It has some **drawbacks**:  
  - The extra hardware can be costly.  
  - Consumes battery while in use.  
  - Requires some warm-up after a cold start to get an initial fix on visible satellites.  
  - It also suffers from "canyon effects" in cities, where satellite visibility is intermittent.
GPS Satellite Orbits

- The satellites orbit the earth with a speed of 3.9 km/sec.
- Have a circulation time of 11 h 58 min earth time.
- The height of the orbits is then about 20200 km.
- The satellites are arranged on 6 planes, each of them containing 4.
- The inclination angle of the planes towards the equator is 55°.
- The planes are rotated in the equatorial plane by 60° against each other.

Orbits

- Two round trip in a day – yellow arrow is the position at 9:30
- The correlating “zone of sight”, within which the satellite signal can be received, is mark in light blue.
Measuring the time delay

- Satellite $i$ sends its $x_i, y_i, z_i$ coordinates and $tr_i$ (time when message was generated).
- Receiver and satellite use the same code generation procedure and they are synchronized.
- Compare incoming code with receiver generated code.

**Diagram:**
- Generated by the receiver: Series of ones and zeroes repeating every 1023 bits. So complicated alternation of bits that pattern looks random thus called "pseudorandom code".
- Received from the satellite:

Measure time difference between the same part of code.
Accurate Timing is the Key

- Satellites have highly accurate atomic clocks
- Receivers have less accurate clocks
- Measurements made using “nanoseconds”
  - 1 nanosecond = 1 billionth of a second
  - Speed of light $c = 299\,792\,458\,\text{m/s}$
  - In 1 nanosecond the light travels for 0.299 m
- 1/1000th of a second error could introduce error of 299 Km
- Discrepancy between satellite and receiver clocks introduces errors in the time delay computation.
How GPS works?

- The **time delay** for a signal to arrive to the GPS receiver is obtained from the message.
- **Range** from each satellite calculated:
  - range = “time delay” * “speed of light”
- Technique called trilateration is used to determine your position or “fix”:
  - Intersection of 3 spheres are (in general) giving two points – the closest to the earth is selected.
- However, 4 satellites should always be used:
  - The 4th satellite used to compensate for inaccurate clock in GPS receivers.
  - Yields much better accuracy and provides 3D fix.
Sources of Errors

- Largest source is due to the atmosphere
  - Atmospheric refraction
    - Charged particles
    - Water vapor
Refraction

- $n_i$ is called refraction index
- $n_2 > n_1$; $v_2 < v_1$
- Light “search” the shortest path
Other Sources of Errors

- Geometry of satellite positions
- Satellite clock errors
- Satellite position or “ephemeris” errors
- Quality of GPS receiver
- Multi-path errors
Short-range positioning beacons

- In relatively small areas, such as a single building, a **local area network** can provide locations along with other services
  - For example, appropriately equipped devices can use **Bluetooth** or **wifi** signals for short-range positioning
- Another possibility is using **RFID (Radio Frequency Identification)**
  - They can encode the position
  - Signal the position to the reader when it is close to the tag.
Active RFID and Reader

**CCC Sputnik OpenBeacon Tag**
OpenBeacon is a free design for an active RFID device which operates in the 2.4GHz ISM band. The device contains a unique serial number, but may have other information. OpenBeacon is designed as a transceiver device and therefore both transmits and receives radio waves.

**OpenBeacon USB node**
The OpenBeacon USB-node can be used as a sniffer to display tag sightings. You simply have to plug it into your computer via USB and follow the instructions on: [http://wiki.openbeacon.org/wiki/OpenBeacon_USB](http://wiki.openbeacon.org/wiki/OpenBeacon_USB)
The OpenBeacon USB-node is a very small (50.5 mm x 24 mm) variation of the 2.4GHz OpenBeacon reader which contains a freely programmable 32 bit ARM controller. It can be reprogrammed without special tools over USB and the firmware is available under the GNU General Public License.
Positioning with Active RFID (I)

- the **landmarks** are equipped with **active RFID**
- the **mobile user** has a mobile device with a **RFID reader**
- the **RFID transponders** (landmarks) **send their ID** (at different signal strengths) **to the reader** of the mobile user
- the **reader** use an **algorithm** for deciding the **closest transponder** (the transponder with the largest number of ID received at the highest signal strengths)
- the position of the transponder is know and therefore also the position of the user.
Positioning with Active RFID (II)

- The **landmarks** are equipped with **active RFID**
- The **mobile user** is also equipped with an **active RFID**
- **Readers** are installed in various locations to **cover the required area**
- The **RFID transponders** (landmarks and user) send their ID (at different signal strength, as before) to the readers

  - **Option 1:** **landmarks and user RFID** detect their **mutual proximity** using an algorithm for deciding the closest transponder (as in solution 1) and send this info to the reader
  - **Option 2:** **the reader compares** the signal received from the user RFID to the signals received from the landmarks and determines (similarity) the closed landmark.
Location API – JSR 179

- The JSR 179 Location API are the J2ME tools for building LBS services
- Are available for
  - CDC
  - CLDC 1.1 (Floating Point required)
- The main classes are:
  - LocationProvider
  - Location and LocationListener
  - Coordinates and PromixityListener
  - LandMark and LandMarkStore
General Location API MIDlet Model

- **Landmark Store**
  - setLocationListener()
  - locationUpdated()

- **LocationProvider**
  - setLocationListener()
  - getInstance()

- **LocationListener**
  - getInstance()

- **ProximityListener**
  - addProximityListener()
  - proximityEvent()

- **GPS module**

- **Custom MIDlet**
  - getInstance()
Fundamental Classes
**LocationProvider Class**

- Abstract Factory Class that build concrete LocationProvider objects satisfying some Criteria passed in the construction.
- Criteria are encapsulated in a Criteria object.
- `getLocation(int timeout)` provides information about user Location within a timeout (otherwise returns an exception).
- It can be in following states:
  - AVAILABLE
  - OUT_OF_SERVICE
  - TEMPORARILY_UNAVAILABLE
- `LocationListener` and `PromixityListener` can listen to events send by a `LocationProvider`.

```
<<abstract>> LocationProvider
    static getInstance(Criteria criteria)

<<abstract>> Location
    getLocation(int timeout)

LocationImpl
    getLocation(int timeout)
```

```
LocationProviderImpl
    static getInstance(Criteria criteria)
    getLocation(int timeout)
```

```
LocationImpl
    getLocation(int timeout)
```
Criteria Class

- Information that describes the requirements that a particular instance of LocationProvider **must satisfy**
- The requirement regards aspect like:
  - Horizontal accuracy
  - Vertical accuracy
  - Response time
  - Max consumption of service
  - Eventual cost
  - Info on speed or direction needed
  - Info on altitude needed
  - Info related to addresses needed
- It is not mandatory for a LocationProvider to satisfy all the requirements.
Creating Criteria

Criteria crit1 = new Criteria();
crit1.setHorizontalAccuracy(25); // 25m
crit1.setVerticalAccuracy(25); // 25m
crit1.setPreferredResponseTime
    (Criteria.NO_REQUIREMENT);
crit1.setPreferredPowerConsumption
    (Criteria.NO_REQUIREMENT);
crit1.setCostAllowed(false);
crit1.setSpeedAndCourseRequired(true);
crit1.setAltitudeRequired(true);
crit1.setAddressInfoRequired(true);
Provider = LocationProvider.getInstance(crit1);
Location

- Can be obtained using the `getLocation(int timeout)` method of `LocationProvider`

- It provides info about
  - Associated address
  - Coordinates
  - Speed
  - Direction (Course)
  - Timestamp of acquired position

- It can be valid or invalid: `isValid()`
  - If valid we can get `QualifiedCoordinates` by using `getQualifiedCoordinates()`
  - An invalid `Location` object doesn't have valid coordinates, but the extra info that is obtained from the `getExtraInfo()` method can provide information about the reason why it was not possible to provide a valid `Location`.

GPS or Network or Cell ID or ...
AddressInfo

- Is obtained by using the method `getAddressInfo()` of `Location`.
- By using the `getField()` method is possible to have various information some of them provided by the following constants:
  - `CITY`
  - `COUNTRY`
  - `POSTAL_CODE`
  - `STATE`
  - `STREET`
  - `PHONE_NUMBER`
  - `BUILDING_NAME`

<table>
<thead>
<tr>
<th>AddressInfo</th>
</tr>
</thead>
<tbody>
<tr>
<td>String <code>getField(int field)</code></td>
</tr>
<tr>
<td><code>setField(int field, String value)</code></td>
</tr>
</tbody>
</table>
QualifiedCoordinates Class

- Is obtained from Location using the getQualifiedCoordinates() method if the Location object is valid (isValid() = true)
- Extend the Coordinates class, adding accuracy information
- Coordinates are expressed in the WGS84 (World Geodetic System 1984) standard.
- static String convert(double coords, outType) converts a double representation of a coordinate with decimal degrees into a string representation. For example, for the double value of the coordinate 61.51d, the corresponding syntax 1 string is "61:30:36" and the corresponding syntax 2 string is "61:30.6"
- static double convert(String coords) converts a String representation of a coordinate into the double representation as used in this API.

Coordinates

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>float getAltitude()</td>
<td>void setAltitude(float a)</td>
</tr>
<tr>
<td>double geLatitude()</td>
<td>void setLatitude(float l)</td>
</tr>
<tr>
<td>double getLongitude</td>
<td>void setLongitude(float l)</td>
</tr>
<tr>
<td>float azimuthTo(Coordinates to)</td>
<td>static String convert(double coords, outType)</td>
</tr>
<tr>
<td>static double convert(String coords)</td>
<td>float distance(Coordinates to)</td>
</tr>
</tbody>
</table>

QualifiedCoordinates

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>float horizontalAccuracy</td>
<td>float verticalAccuracy</td>
</tr>
</tbody>
</table>

outputType - identifier of the type of the string representation wanted for output The constant DD_MM_SS identifies the syntax 1 and the constant DD_MM identifies the syntax 2.
Example: Obtain the Location of the Device

...
This program pops an alert message showing your current location.
The essence of the location extraction is in the following lines (inside the `givePositionAlert()` function):

```java
Criteria cr = new Criteria();
cr.setHorizontalAccuracy(500);
LocationProvider lp = LocationProvider.getInstance(cr);
Location l = lp.getLocation(60);
Coordinates c = l.getQualifiedCoordinates();
```

To get the co-ordinates of current location we basically follow three logical steps:
1. Set the criteria of the location provider
2. Get the location provider instance according to set criteria
3. Get the location object from the location provider
4. Get the co-ordinates from the returned location object
Simulating the position

- You can simulate a location provider and your current position using the external event generator.
import javax.microedition.lcdui.*;
import javax.microedition.midlet.*;
import javax.microedition.location.*;
public class CoordinateAlert extends MIDlet implements CommandListener {
    private Display display;
    private Form mainForm;
    private Alert positionAlert;
    public CoordinateAlert() {
        mainForm = new Form("HelloMIDlet");
        mainForm.append(new StringItem(null, "You will be alerted of your current position."));
        mainForm.addCommand(new Command("Exit", Command.EXIT, 0));
        mainForm.addCommand(new Command("OK", Command.OK, 1));
        mainForm.setCommandListener(this);
    }
    public void startApp() {
        display = Display.getDisplay(this);
        display.setCurrent(mainForm);
    }
    public void pauseApp() {}
    public void destroyApp(boolean unconditional) {}  
    public void commandAction(Command c, Displayable s) {
        String label = c.getLabel();
        if (label.equals("Exit")) {
            notifyDestroyed();
        } else if (label.equals("OK")) {
            mainForm.set(0, new StringItem(null, "Tracking location ...");
            givePositionAlert ();
            mainForm.set(0, new StringItem(null, "Again get alerted of your current position?"));
        }}
private void givePositionAlert() {
    try {
        Criteria cr = new Criteria();
        cr.setHorizontalAccuracy(500);
        LocationProvider lp = LocationProvider.getInstance(cr);
        Location l = lp.getLocation(60);
        Coordinates c = l.getQualifiedCoordinates();
        if (c != null) {
            String lat = c.convert(c.getLatitude(), c.DD_MM);
            if (c.getLatitude() > 0) {
                lat = "E " + lat;
            } else {
                lat = "W " + lat;
            }
            String lon = c.convert(c.getLongitude(), c.DD_MM);
            if (c.getLongitude() > 0) {
                lon = "N " + lon;
            } else {
                lon = "S " + lon;
            }
            positionAlert = new Alert("Location Alert");
            positionAlert.setString("Your present location is 
" + lat + "\n" + lon);
            positionAlert.setTimeout(Alert.FOREVER);
        } else {
            positionAlert = new Alert("Location Error");
            positionAlert.setString("Null Coordinate Received");
            positionAlert.setTimeout(Alert.FOREVER);
        }
    } catch (Exception e) { //LocationException OR InterruptedException
        positionAlert = new Alert("Location Error");
        positionAlert.setString("Exception Encountered " + e);
        positionAlert.setTimeout(Alert.FOREVER);
    }
    display.setCurrent(positionAlert); }
}
Coordinate Alert Using Thread

- The CoordinateAlert can work only if the user interface does not require any input.
- The MIDlet is only working for managing the coordinates.
- Set the security domain to “maximum” if you want to see something.
- Run a new project using CoordinateAlertThread.java.
- It uses a separate thread to access the location info - the UI remains active with the main thread taking care of it.

[Code]
Multi Threading

- Multi threading is important:
  - actions like getting information from a location provider, loading an image, fetching data from a server often require a significant amount of time
  - Such actions should be taken care of in a separate thread so that the user can still interact with the UI
- While doing time-consuming jobs always give a message to the user denoting what the program is doing.
- E.g.: in this program “Tracking location …” is printed to notify the user of the current action.
It is possible to associate a `LocationListener` to a `LocationProvider` specifying:
- `interval`
- `timeout`
- `maxAge`

The `LocationListener` receive notification about:
- `Location` change
- `LocationProvider` status change
Location Listener

- A location listener, e.g., `LocationListenerTest` must implement the interface `LocationListener`.
  - It must implement the methods:
    - `locationUpdated()`
    - `providerStateChanged()`
- The location provider instance **notifies** changes (location, state of the location provider) to the registered location listener.
  - The location provider calls the `locationUpdated()` method in regular intervals as specified while setting the listener.
  - If the state of the location provider changes, `providerStateChanged()` is invoked.
About Location Listener

- Check the values passed as `interval`, `timeout` and `maxAge` arguments to:
  ```java
  setLocationListener(LocationListener listener, int interval, int timeout, int maxAge)
  ```

- `-1` is passed to set default values specific to the provider

- Setting default values is a safe option but should match your application requirement

- The implementation tries to provide location info at the specified `interval` but the interval might not be exact
  - E.g.: If the provider becomes temporarily unavailable or out of service or if the interval specified is too short for the provider, the implementation might update with an invalid location

- `LocationUpdated()` method should immediately pass the control to a different function – ready to take care of the next location update.
ProximityListener Class

- It is possible to associate a group of ProximityListener to a LocationProvider specifying:
  - The coordinates to be registered
  - Proximity radius – in meters to be used as threshold for being in the proximity
- The ProximityListener receives notification about
  - Proximity monitoring status changes (true=active, false=is down)
  - The current location of the terminal is within the defined proximity radius of the registered coordinates.

```
LocationProvider

1

LocationListener
locationUpdated
providerStateChanged

0..1

ProximityListener
monitoringStateChanged(boolean isMonitoringActive)
proximityEvent(Coordinates coords, Location location)

0..*
```

```java
abstract LocationProvider

static addPromixityListener(ProximityListener listener, Coordinates coords, float radius)
static removePromixityListener(ProximityListener listener)

interface ProximityListener

monitoringStateChanged(boolean isMonitoringActive)
proximityEvent(Coordinates coords, Location location)
```
Handling Landmarks

- **Landmarks** are of prime importance for developing LBS application.
- **Any point** with a specific latitude and longitude can act as a landmark.
- The location API comes with a built-in Landmark class to hold `name, description, position` and any other relevant info.
- Applications such as **Point of Interest locators** (finding hotels, restaurants, petrol pumps etc.), or routing (route from one place to another) use the concept of Landmark.
Landmark Class

- A Landmark is a known location with a name.
- It contains information related to:
  - Address
  - Description
  - Name
  - Coordinates with accuracy information
- It can be made persistent using a LandmarkStore.

<table>
<thead>
<tr>
<th>Landmark</th>
</tr>
</thead>
<tbody>
<tr>
<td>AddressInfo getAddressInfo()</td>
</tr>
<tr>
<td>String getDescription()</td>
</tr>
<tr>
<td>String getName()</td>
</tr>
<tr>
<td>QualifiedCoordinates getQualifiedCoordinates()</td>
</tr>
<tr>
<td>… and corresponding set methods</td>
</tr>
</tbody>
</table>
Landmark Example

- Assume that a form contains some text fields, e.g., stateField, nameField, etc.
- Info related to the landmark are collected, and then the Landmark is created:

```java
AddressInfo info = new AddressInfo();
info.setField(AddressInfo.COUNTRY, countryField.getString());
info.setField(AddressInfo.STATE, stateField.getString());
info.setField(AddressInfo.CITY, cityField.getString());
info.setField(AddressInfo.STREET, streetField.getString());
info.setField(AddressInfo.BUILDING_NAME, buildingNameField.getString());

Landmark lm = new Landmark(nameField.getString(), //name
descField.getString(), //descr.
coord, //QualifiedCoordinates
info); //AddressInfo
```
**LandmarkStore Class**

- It’s a class for storing, deleting and retrieving Landmark from a persistent store.
- It allows to organize Landmark in categories.
- It has some methods for access and search the Landmark using category and location information.
- It is shared between all the J2ME device applications.

<table>
<thead>
<tr>
<th>LandmarkStore Method</th>
</tr>
</thead>
<tbody>
<tr>
<td>static createLandmarkStore(String storeName)</td>
</tr>
<tr>
<td>static deleteLandmarkStore(String storeName)</td>
</tr>
<tr>
<td>static String[] listLandmarkStore()</td>
</tr>
<tr>
<td>static LandmarkStore getInstance(String storeName)</td>
</tr>
<tr>
<td>addCategory(String categoryName)</td>
</tr>
<tr>
<td>addLandMark(Landmark landmark, String category)</td>
</tr>
<tr>
<td>deleteCategory(String categoryName)</td>
</tr>
<tr>
<td>deleteLandmark(Landmark landmark)</td>
</tr>
<tr>
<td>Enumeration getCategories()</td>
</tr>
<tr>
<td>Enumeration getLandmarks()</td>
</tr>
<tr>
<td>Enumeration getLandMarks(String category, String name)</td>
</tr>
<tr>
<td>Enumeration getLandmarks(String category, double minLat, double maxLat, double minLong, double maxLong)</td>
</tr>
<tr>
<td>removeLandmarkFromCategory(Landmark lm, String cat)</td>
</tr>
<tr>
<td>updateLandmark(Landmark landmark)</td>
</tr>
</tbody>
</table>
Getting or Creating a LandmarkStore

```java
try {
    store = LandmarkStore.getInstance(STORENAME);
} catch(NullPointerException npe) {...}
if (store == null)  // if there is no store with that name
    // code ...

---

try {
    LandmarkStore.createLandmarkStore(STORENAME);
} catch(IllegalArgumentException iae) {}
    // Name is too long or landmark store with the specified name
    // already exists.
    catch (IOException e) {}
        // Landmark store couldn't be created due to an I/O error
        catch (LandmarkException e) {}
            // Implementation does not support creating new landmark stores.
```
Getting Landmarks

- The `LandmarkStore` class contains a set of landmark management methods.
- Enumeration of `Landmark` objects can be obtained with one of the three `getLandmarks()` methods:
  - `getLandmarks()`: all landmarks
  - `getLandmarks(String category, String name)`: all landmarks with given category and name
  - `getLandmarks(String category, double minLat, double maxLat, double minLong, double maxLong)`: all landmarks in a category a given area.
Orientation

- It can be available in devices with a compass (call Orientation.getOrientation())
- It represents the physical orientation of the device and provides information about:

  - [Azimuth](#)
  - [Pitch](#)
  - [Roll](#)

**Orientation**

- float getCompassAzimuth()
- float getPitch()
- float getRoll()
- static Orientation getOrientation()
- boolean isOrientationMagnetic()
## MIDP 2.0 Security Framework

<table>
<thead>
<tr>
<th>Permission Name</th>
<th>Methods protected by this permission</th>
</tr>
</thead>
</table>
| javax.microedition.location.Location     | LocationProvider.getLocation()  
|                                         | LocationProvider.setLocationListener()                                    |
| javax.microedition.location.Orientation  | Orientation.getOrientation()                                               |
| javax.microedition.location.PromixityListener | LocationProvider.addPromixityListener()                                        |
| javax.microedition.location.LandmarkStore.read | LandmarkStore.getInstance()  
|                                         | LandmarkStore.listLandmarkStores()                                        |
| javax.microedition.location.LandmarkStore.write | LandmarkStore.addLandmark()  
|                                         | LandmarkStore.deleteLandmark()                                             |
|                                         | LandmarkStore.removeLandmarkFromCategory()                                 |
|                                         | LandmarkStore.updateLandMark()                                             |
| javax.microedition.location.LandmarkStore.category | LandmarkStore.addCategory()  
|                                         | LandmarkStore.deleteCategory()                                             |
| javax.microedition.location.LandmarkStore.management | LandmarkStore.createLandmarkStore()  
|                                         | LandmarkStore.deleteLandmarkStore()                                        |
Many users consider location information to be highly sensitive, and are concerned about a number of privacy issues, including:

- **Target marketing**: Mobile users' locations can be used to classify customers for focused marketing efforts
- **Embarrassment**: One customer's knowledge of another's location may lead to embarrassing situations
- **Harassment**: Location information can be used to harass or attack a user
- **Service denial**: A health insurance firm might deny a claim if it learned that a user visited a high-risk area
- **Legal restrictions**: Some countries regulate the use of personal data.
Guidelines

- Handle **unavailability** of services gracefully - the user's location may not always be available:
  - The device cannot access any location methods (e.g., in a tunnel)
  - The user refrains to give the information
- Determining the location may **take a long time** that the end result isn't useful anymore - keep the user informed
- Be sensitive to **privacy** concerns:
  - Tell customers **about the information** being collected on them and **how it will be used**
  - Offer customers the **choice** of what location information to disclose, and when appropriate an **option** not to participate
  - Allow customers to **review their permission profiles** so that they know what they are permitting
  - **Protect** location information so that it cannot be accessed by unauthorized persons.
Getting the coordinates

- The coordinates reference system is the WGS84 standard
- The following web site let you retrieve the appropriate lat and long: http://boulter.com/gps/
Showing the lat long of a place

Google Maps Latitude, Longitude Popup

Simply click on the map on a location and it will provide you with the latitude and longitude in the callout window.

Map data ©2009 Tele Atlas - Terms of Use

* Coordinates:
  * Lat: 45.748360
  * Lon: 11.623535

* Flickr tags:
  geotagged geo:lat=45.748360 geo:lon=11.623535

* RoboGEO tags:
  45.748360;11.623535

* Show location on Multimap
* Permanent Link

MapInfo OnDemand / SAAS
Your workspace (.war) on the web
Sign Up It's Free for MapInfo users

http://www.gorissens.info/Pierre/maps/googleMapLocation.php
For advanced users:

- The Michelin Geocoding **Web Service** allows you to enter an address and obtain an ordered list of locations with an address description and the associated WGS84 encoded geographic coordinates.

Location scripts (WTK simulator)

- For more elaborate testing, you can set up a location script that describes motion over time.
- Location scripts are XML files consisting of a list of locations, called waypoints, and associated times.
- The Sun JavaTM Wireless Toolkit for CLDC determines the current location of the emulator by interpolating between the points in the location script.
- Example - a simple location script that specifies a starting point (time="0") and moves to a new point in ten seconds:

```xml
<waypoints>
  <waypoint time="0"
    latitude="14" longitude="50" altitude="310" />
  <waypoint time="10000"
    latitude="14.005" longitude="50.001"
    altitude="315" />
</waypoints>
```
- The altitude measurement is in meters, and the time values are in milliseconds.
Related material

- MIDP Location API Developer’s Guide


- The Java Location API (ddj article)
  - http://www.ddj.com/dept/java/184406388

- JSR 179 - Location API JavaDoc

- Lots of information about Location API, GPS
  - http://www.trekbuddy.net/