Mobile Cell Phone GPS Navigation

Bachelor’s Thesis in
Computer Systems Engineering

Sha Mao Xuan, Xie Jun, Xu Xiao Lin
Mobile Cell Phone GPS Navigation

School of Information Science, Computer and Electrical Engineering
Halmstad University
Box 823, S-301 18 Halmstad, Sweden

May 2009
Acknowledgement

This GPS mobile project has taken four months to finish. There are three people who gave us a lot of help. First, I want to thanks to Lypson company supervisor Michal Lysek and Tobias Persson. They also encouraged us when sometimes we had a hard time. They really made us feel free to ask. We really appreciate their help. Secondly, we must give thanks to our thesis supervisor Ph. Niculina Mânsson. We would like to thank her for giving us the book and guidance on our thesis. Without her assistance, we could not have finished it. At last we want to give thanks to our family, to our parents. Thanks to our parents for supporting us in studying here. Our parents gave us the most important help.
Details

Surname, First Name: Sha Mao Xuan
Xie Jun
Xu Xiao Lin

University: Halmstad University, Sweden

Degree Program: Computer Systems Engineering

Title of Thesis: Mobile Cell Phone GPS Navigation

Academic Supervisor: Ph. Niculina Månsson

Cooperating company: Lypson Intelligent Systems

Company Supervisor: Michal Lysek
Tobias Persson
Abstract

Nowadays, GPS Mobile has already become an important part of our life. GPS is widely used for citizen and military. For military, they could use GPS to navigate the missile or space flight. For citizen, people use GPS, especially GPS Mobile, to navigate a car. GPS also helps people to find their way to home. GPS are widely used on police, company manage system, agriculture and so on. GPS Mobile is the most familiar device for all of us to use. To combine use GPS Mobile and Google Map, people can get their currently location and shortcut to their destination. Also, we can get information from the Google map, like shopping place, bus station and so on. The main purpose of our project is to help people reach this aim. By using our software, you can find your location on the Google map. When you travel abroad, you can have your own language map on the GPS mobile, easily and quickly. That’s really convenience. The most wonderful part, you can play game with your friend by using our software. What you need to do, it is just send a message to anyone you care about and then you can get their location and show it on the map. So, even it is a far distance between you and your friend. You can have fun together. That is our software Findyourfriendlocation.
Contents

Acknowledgement ................................................................................................. iii
Details ................................................................................................................... iv
Abstract ............................................................................................................... v
Contents ............................................................................................................... vi

1 Introduction ...................................................................................................... 1
  1.1 Background ................................................................................................. 2
  1.2 The goal of the Project ............................................................................. 2

2 Developing Tools ............................................................................................. 3
  2.1 Java ............................................................................................................. 3
  2.2 J2ME .......................................................................................................... 3
    2.2.1 Package .............................................................................................. 3
  2.3 Related Work ............................................................................................. 4
    2.3.1 Navigation software demo from Lypson ............................................. 4
    2.3.2 GuideBee ............................................................................................ 4
    2.3.3 PStreets ............................................................................................... 5
  2.4 The research of navigation software & map servers ............................... 5

3 Methods ............................................................................................................ 7
  3.1 Google static maps API ............................................................................. 7
    3.1.1 Certain parameters ........................................................................... 8
    3.1.2 Optional parameters ......................................................................... 8
  3.2 Geocoding ................................................................................................ 11
  3.3 GPS and cell phone positioning theory .................................................... 12
  3.4 The research of different function sample .............................................. 13
    3.4.1 Connect to the internet ................................................................. 13
3.4.2 Sending message ......................................................... 14
3.5 The research about route function ................................. 14

4 The Programming of FindYourFriendLocation .................. 15
4.1 The need analyzes ........................................................... 15
4.2 Program process .............................................................. 15
4.3 Design elucidation in detail .............................................. 16

4.3.1 Main interface ............................................................. 16
4.3.2 Show map ................................................................. 17
4.3.3 Current location ......................................................... 17
4.3.4 Special Place Map ...................................................... 17
4.3.5 Make marker .............................................................. 18
4.3.6 Send message .............................................................. 19
4.3.7 Text split ................................................................. 20
4.3.8 Map type ................................................................. 20
4.3.9 Tracking ................................................................. 20

5 Results .............................................................................. 21
5.1 Main interface ............................................................. 21
5.2 Menu ........................................................................... 21
5.3 Map menu ................................................................. 22

5.3.1 Zoom in ................................................................. 22
5.3.2 Zoom out ................................................................. 22
5.3.3 Map up ................................................................. 22
5.3.4 Map down .............................................................. 22
5.3.5 Map left ................................................................. 22
5.3.6 Map right ............................................................. 23

5.4 Current location ....................................................... 23
5.5 Special place map ..................................................... 23
5.6 Make marker .......................................................... 24
5.7 Send message ........................................................ 24
5.8 Map type ................................................................. 25
5.9 Tracking ................................................................. 26
5.10 About ................................................................. 26

6 Conclusion and Future Work ........................................... 27
  6.1 Conclusion ........................................................... 27
  6.2 Future Work ........................................................ 27

Bibliography ................................................................. 28

Appendix ................................................................. 30
1. Introduction

With the development of science, technology and life quality improving, mobile telephone had more and more functionality than before; one of the most popular functionality is GPS navigation.

GPS (GLOBAL POSITION SYSTEM), established by the United States satellite navigation and positioning system. GPS use 12 satellites to navigate GPS device. First, GPS device send a request to satellites. Second, satellites calculate the arrival time of the request. Third, All the satellites send the calculate result back to the mobile phone terminal. At last, the cellular telephone terminal sends evaluation result to the cell phone. Now, the consumers of GPS navigation cell phone increase day by day. The main reason is that the navigation cell phone gives us huge convenience of the daily life. For example, when you want to find a shortcut or need first aid, you all can use it.

This project is focus on programming GPS navigation software for cell phone, and this software’s name is Findyourfriendlocation. Findyourfriendlocation uses Google Maps API to download the limited area’s map’s GIF file. This image will be display on the cell phone screen. Google Maps API’s function includes searching the road name, special places within the limited area. These special places could be hotel, supermarket, restaurant and shop. If you use our software, you will find that you can share the location with your friends by sending a message which contains the location information. Since this is the most interesting function in this software, we decided to name the software Findyourfriendlocation.

The best part for this software is we can know the location of our close friend in anytime, anywhere. If you want, you can mark friend’s location on the cell phone, and then you can send message to you friend again to keep tracking his or her location. Also, there is a
function in this software called tracking; you also can keep tracking your own position every per five seconds.

1.1. Background

This interesting project’s idea comes from the Lypson Company. Lypson Company is a Swedish consultant company. This company works with software applications and game engine solution. The Lypson Company had a lot of experience about embedded systems. Their product also has many mobile phone games. If you want to know more about this company you can go to this website http://www.lypson.se/.

1.2. The Goal of the Project

The main purpose in our project is to develop a mobile cell phone application to identify the location and use the GPS data to navigate. The main data for this software come from GPS device and Google Map Server.

This software is based on following functionalities:

- Download the map to cell phone and add zoom and move map function.
- Locate the position and mark the current position on the map.
- Ask another user for his coordinate by sending a message.
- Locate other people’s position and mark his position in different color sign.
- Show the shortcut between you and your friend.
2. Developing Tools

2.1. JAVA

Java is a modern programming language. Programmers can use Java to develop applications for servers, desk computers and small handsets (e.g., mobile phone, pager) and so forth.

2.2. J2ME

Java 2 platform, Micro Edition (J2ME) is dedicated to developing mobile application. It can also develop some other small devices (e.g., notebook).

2.2.1. Package

MIDP packages Javax.Microedition have four important components, including:

<table>
<thead>
<tr>
<th>Package Name</th>
<th>The Application of Package</th>
</tr>
</thead>
<tbody>
<tr>
<td>Javax.Microedition.Rms</td>
<td>Rms is Record Management System</td>
</tr>
<tr>
<td>Javax.Microedition.Midlet</td>
<td>A framework for the definition of the Midlet and Midlet interaction with the environment</td>
</tr>
<tr>
<td>Javax.Microedition.Io</td>
<td>Provide for the network support</td>
</tr>
<tr>
<td>Javax.Microedition.Lcdui</td>
<td>Provide for the user interface</td>
</tr>
</tbody>
</table>

Table 2.1: These four packages are come from Javax.Microedition [1] package. They are the basic packages which we used in our software.
2.3. Related Work

This is our first time to develop mobile phone software and therefore we try to find mobile phone software sample from the internet. As a first step, we got three existing solutions for us to understand how navigation software works.

2.3.1. Navigation Software Demo from Lypson

We got this midlet from Lypson Company. The best part of this midlet is downloading the map from Google server. The part of downloading the map used Google static map API. It is a very good idea.

2.3.2. GuideBee

Figure 2.1: GuideBee show the example of GuideBee Cell Phone software interface.

Figure 2.1 is provided by a Chinese company (GuideBee), but the original idea is come from the Microsoft. The keypads in this software are easy to use, because this interface uses canvas to display the map. In J2ME, when u needs to do some custom drawing, we use Canvas. By using canvas, we can use cell phone’s joystick. Joystick is really convenient to enter a selection, to scroll left, right, up, down, and to move around the map on the screen. By the way, zoom in and zoom out of the map also will be easy. In the Figure 2.1, the map is also download from the Google server, but we find it is possible to download map from your own sever. However, you need to create your own map first. That is not easy.
2.3.3. PStreets

![Figure 2.2: PStreets show the example of PStreets Cell phone software interface.](image)

Microsoft develops PStreets. Actually, there much navigation software simulates this software. Our software’s function gets idea from this software (e.g., choose map type, tracing and so on).

2.4. The Research of Navigation Software & Map Servers

As you have seen above, these software are existing solutions provide by different companies. By studying Lypson Company’s example, we understand how to use J2ME’s visual design to develop Cell phone software, and we get an idea as to how to start our project from the company. By researching Guidebee and PStreets, we find that there are three kinds of the navigation software.

The first type of navigation software, map file was downloaded from the map server. They are easy to use and easy to develop. However, they need to keep charging an Internet fee from customers.

The second type of software, the map was not downloaded by the Internet. They use their own map which stored at the cell phone memory. For example, they build one map in limited area (e.g., the map for Halmstad) and collect the location information (longitude and latitude) to connect with the map which they made. The advantage of this kind of software is that they use their own map stored at the cell phone, thus they do not need to
Developing Tools

charge any Internet fee from the users. Usually, they only pay for the map to the company who created the map only once.

Finally, another type of navigation software both downloads the map from the map server, and also use the map which created by the software company. We find that there are not only Google and Microsoft has map servers, there are also many other companies that provide map servers. We made a list below for the reader to get further information about different map server types.

<table>
<thead>
<tr>
<th></th>
<th>Map</th>
<th>Satellite</th>
<th>Hybrid</th>
<th>China Map</th>
<th>India Map</th>
</tr>
</thead>
<tbody>
<tr>
<td>Google</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Microsoft</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Yahoo</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>Askdotcom</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>Openstreet</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 2.2: Map Server Research. X means which company have this type of map (e.g., Google has map, satellite, hybrid and China map type, but Google doesn’t have India Map type)

In the Table 2.2, map means road map. A road map’s information focuses on road names etc. You cannot see the real city map (e.g., planform) by a road type of map, and therefore a satellite map fixes this problem. A hybrid map is like an up-market version for the satellite. It not only supports the real city image but also includes road name information. Most of the map server they all have these three types of map. However, some of the map servers support special area maps (e.g., Google China map and Yahoo India Map). The reason is the normal road map is not supported in these areas and that is why there is a special map type used in these areas.
3. Methods

3.1. Google Static Maps API

Our project uses Google Static Maps API to download the special map image and display the map image on the cell phone screen.

Google Static Maps API is used to embed a Google Map image on any device on which you can display an image. It could be a webpage or a cell phone screen. The basic idea of how to use Google Static Maps API is based on organized URL parameters. By sending different URL parameters through a normal HTTP request, the Google server send back different image to the cell phone.

As an example, Google Static Maps API URL must have the following format: http://maps.google.com/staticmap?parameters. There are twelve different parameters which we can choose. Some of them are certain parameters and the others are optional parameters. Each of the parameters is separated by using the ampersand (&) character. The example of different parameters is shown below. In addition, the example location will be based on the Halmstad University Map.

![Google Static Maps API](image)

Figure 3.1: Figure 3.1’s URL request address is defined by three certain parameters, center, size and key (e.g., center = 56.663, 12.879 & size = 240 x 270 & key = MapAPIKey). By using the ampersand (&) character, we can combine these parameters. All in all, the parameters center, size and key, are basic parameters. You cannot download an image without them. For the reason of
3. Methods

protecting personal information, we write \("& key = MapAPIKey\) instead of the real keys. We will talk about how to use these parameters in the subchapters 3.1.1 and 3.1.2.

3.1.1. Certain Parameters

- **Key**
  
  If you want to use Google Static Maps API, you have to use your own Google API Key [2].

  \(\text{The example of form: } & key = MapAPIKey\).

- **Center**
  
  Center parameter is one of the certain parameters. We can use it to define the center of the map. This parameter is comprised of two parts: latitude and longitude. These two parts are separated by a comma.

  \(\text{The example of Form: } & center = 56.663, 12.879\)

  \(\text{Values range:}\)

  \(\text{Latitudes value between -90 and 90}\)

  \(\text{Longitude value between -180 and 180}\)

  \(\text{If the request value is over this range, the Http request will fail.}\)

- **Size**
  
  Size parameters set the rectangular area of the map image.

  \(\text{The example of form: } & size = 240 x 270\)

  \(\text{This means that the map size is 240 pixels wide by 270 pixels high.}\)

  \(\text{Values range of size: } (0 x 0) \sim (512 x 512) / \text{pixels}\)

3.1.2. Optional Parameters

- **Zoom (required if markers not present)**
  
  Using Google static maps API, each map has a zoom level, as exemplified in the following picture.

  \(\text{The example of form: } & key = MapAPIKey.\)
Values range of zoom: 0 ~ 19

Figure 3.2: These two image requests are from almost the same URL address. The difference between them is (a) set the zoom parameter 12 and (b) set zoom parameter 13, thus (a)’s map area size is smaller than (b).

- **Markers**

By using marks parameter, we can put as many markers as we want on the map.

*Form of markers:* &key=MapAPIKey.:latitude, longitude, (size) (color)

(alphanumeric-character)

*The example of form:* & markers = 56.663, 12.879, midblues

*Values range for latitudes and longitude:*

*Latitudes value between -90 and 90*

*Longitude value between -180 and 180*

*Size (the marker size):* tiny, mid and small.

*Color (the marker color):* black, brown, green, purple, yellow, blue, gray, orange, red and white

*Alphanumeric-character (the marker character):* a-z, 0-9.

If you have more than one marker, each mark could use the pipe character (|) to separate. Furthermore, we find we can also use the ampersand (&) character to separate. So, if the markers are more than one, the form could be like this:

Markers = markerform1| markerform2| markerform3|... etc.
Markers = markerform1\&markers = markerform2...etc.

In this project, we use markers parameters to show a friend’s location, or display a special place on the map. An illustration of markers is show in below.

![Marker Illustration](image)

Figure 3.3: As you can see, we use markers to indicate special points on the map. By changing marker size, marker color, marker character, we get different three points. Thus, by using this function, we can mark a friend’s location on the map. You just need make sure you get your friend’s latitude and longitude.

- Map type

There are seven map types you can choose on the map type’s option. By using different map types, we can choose different map information on the cell phone screen. As an example, you can see below.

![Map Types](image)

(a) Satellite. (b) Hybrid.

Figure 3.4: These two pictures send almost the same URL Request. However, (a) set the map type’s parameter that is satellite. (b) set the map type’s parameter that is hybrid. In (a), we only can see the real city, but in (b) we can see the road name. Similarly, there are still another 5 types
of map type. There are roadmap, mobile, terrain, mapmaker-roadmap and mapmaker-hybrid. For further discussion about these map types, you can find them on reference Google Static Map Guide [3].

- Format, Frame, Hl, Span, Path
  There are still five kinds of parameters which we have not used in this project. However they are still useful. For further discussion about these map types, you can find them on reference Google Static Map Guide [3].

### 3.2. Geocoding

In the first time, when we tried to download the special position map, we were using latitudes and longitudes, yet it is almost impossible for people to remember latitudes and longitudes. Alternatively, we could have used addresses to download the special map by using Geocoding service.

The purpose of Geocoding is convert addresses (e.g., Nitaregatan, 302 50 Halmstad, Sweden) into geographic coordinates (e.g., latitude = 56.663671, longitude = 12.875058), thus we do not need to remember the latitudes and longitudes any more. Similarly, using Geocoding is just like using Google Static Maps API. We send different URL parameters directly to the Google maps server by an http request. In contrast with Google Static Maps, API URL parameters are much simpler. The form of Geocoding URL Request address: http://maps.google.com/maps/geo?{address}{output}{key}.

- Address
  You can write the road name, place name, country name into the address. Each of the names is separated by a comma character.

    *The example of form:* & q = Nitaregatan, 302 50 Halmstad, Sweden

- Output
  Actually, there are four types of output which we can use. There are xml, kml, csv and json. However, in this project, we can only choose csv as output.
3. Methods

The reason is csv output sends back a clean simple text file as a result (e.g., 200, 6, 56.6636713, 12.8750588). The other output sends back date type and are not suitable for us. For example, xml send back webpage file which contains redundant information that we do not need. You can find more information on Map Overlays [4] (Geocoding).

The example of form: & output = csv.

- Key

You need a personal Google Maps Keys, and then you can use this web service. You can find more information on Reference Google Maps Key [2].

The example of form: & key = MapAPIKey

3.3. GPS and Cell Phone Positioning Theory

Our navigation software uses a GPS device to get latitudes and longitudes. These GPS devices should already be embedded in the mobile phone. To let you understand how a GPS device works, I will introduce GPS (Global Positioning System [5]) and Cell Phone Positioning Theory to you.

Talking about positioning technology; the most famous one is GPS. Another is TOA (Uplink Time Difference of Arrival) technology. The GPS is a global navigation satellite system (GNSS) developed by the United States Department of Defense and managed by the United States Air Force 50th Space Wing. It is the only fully functional GNSS in the world, can be used freely by anyone, anywhere, and is often used by civilians for navigation purposes. It uses a constellation of between 24 and 32 medium Earth orbit satellites that transmit precise radiowave signals, which allow GPS receivers to determine their current location, the time, and their velocity. Its official name is NAVSTAR GPS. How to use positioning technology depends on the cell phone network. In Europe, the standard cell phone network is GSM (Global System or Standard) for mobile. In fact, the GSM is using TOA technology. TOA technology is depending on a triangle form; when
it works, it needs at least three cell phone bases to receive the cell phone signal. After the signal arrives at the bases, each of the bases calculates the arrival time and then we can get the cell phone location. This technology is much more suitable in the city because there are more bases which mean more accuracy.

There is another cell phone network called CDMA (Code Division Multiple Access), and this network uses A-GPS (Assisted Global Positioning System [6]) as its positioning technology. As its name suggests, it works with GPS. Usually, the cell phone company, or an independent GPS data server, sends location information advice to cell phones which they receive from the GPS device. After the mobile receives the information, it adjusts its location. The GPS device could install in cell phone bases and the GPS data server, or it could be directly embedded in the cell phone. It also works well out of the city because it gets signal from satellite. Sprint PCS, Verizon and Nextel are all using this technology.

In conclusion, how the cell phone gets locations not only depends on the cell phone itself. It depends both on the type of cell phone network and whether the cell phone is capable of GPS functionality.

3.4. The Research of Different Function Sample

At the beginning of this project, we studied different J2ME function samples to get an idea how to start our project. There are two important functions that I must introduce to you. The first of them is the function to connect to the Internet. The second is the function of sending messages between two phones. These two programs are all studied from the book (Beginning J2ME Platform from Novice to Professional). I will only introduce you in brief.

3.4.1. Connect to the internet

This MIDlet teaches us how to download images from websites. You can find the detail of programming code at the Reference Beginning J2ME Platform Chapter 10 Connecting
3. Methods

to the World [7].

3.4.2. Sending Message

In this function model, you can send message “red”, “blue” to another cell phone. After you send the message to another phone, the image will have different response by the message content “red” or “blue”. In this model, we get an idea of how to send message between two phones. You can find the detail of programming code at the Reference Beginning J2ME Platform Chapter 11 Wireless Messaging API [7].

3.5. The Research about Route Function

In the last part of this project, we search the information for how to develop the route function, since we found the Google server only supports JavaScript API for the route function. In the beginning, we just tried to let J2ME receive the data from the Route API URL (e.g., http://ditu.google.com/maps?f=d&source=s_d&saddr=shanghai&daddr=beijing&hl=en&ie=UTF8&output=html), however, we found the response does not contain any useful information. Then we use the debug function to analyze the data obtained from the URL. We find the truth is the data we get back is a JavaScript. However, J2ME does not support JavaScript API, thus we consider using another program which can support JavaScript to handle the JavaScript API, and then send the information to the J2ME program. By research from internet, we found that there is one kind of data analysis server which could help us. It is programming by J2SE and J2SE that is supported JavaScript API. This kind of software needs using HTTPCLIENT and JERICHO HTML PARSER [8] to get the data from Google JavaScript. It will take more time than we expected, thus we change this function to the other function.
4. The Programming of FindYourFriendLocation

4.1. The Need Analyzes

A mobile cell phone GPS navigation application must be capable of downloading the map, and marking current coordinates on the map. The user can move, zoom in, and zoom out of the map and put the marker in the place where he needs on the map. The user can look up the map that he needs (for example, some cities and roads). Inputting the name of city or road, the user gets the map which he wants. The most important function is mark of the friend’s location. It is a very practical function. The users can know where their friends are now.

4.2. Program Process

Figure 4.1: After you start the application, you need to confirm that you use some services. Then
the map and your location will be displayed in the main interface. You can choose between show
the map, current location, special place map, make marker, send message, map type, tracking,
stop tracking, about and exit functions. Every function has a back button and you can return to the
main interface by pressing the back button.

4.3. Design Elucidation in Detail

There are two main parts for the software design. There are download the map from
Google Server and cell phone positioning. The rest of design we will introduce in
subchapters.

- Download the map from Google Server
  Before the beginning of all the functions, we need to display the map on the
  mobile cell phone. We use Google Static Map and get an image by sending the
  specific URL, and then we use connector’s open() method to send URL and
  return a HttpConncetion. We transform it into InputConnection and use
  openInputStream() method to read data. Next, we save the data to a byte array.
  We create an image and put image data in it, finally displaying the image on the
  cell phone form. You can find them on reference Beginning J2ME Platform
  Chapter 10 Connecting to the World [7].

- Cell Phone Positioning
  The mobile cell phone positioning needs to get the coordinate of current location.
  Using package javax.microedition.location, we use the getLocation() method to
  get the current location which the mobile cell phone GPS device measures. Then
  we use the getQualifiedCorrdinates() method to put the coordinates of the
  current location into Coordinates. Finally, we get the latitude and longitude of
  the coordinates by getLatitude() and getLongitude() method. You can find them
  on reference Package Javax.Microedition.Location [9].

4.3.1. Main Interface

We can get the map by sending the URL. Moreover, we can move and zoom in or zoom
out the map to get a new picture. We could change the value of map center coordinates (e.g., mapCenterLatitude, mapCenterLongitude) and the size of map (e.g., mapzoom) to get a new map. However, we had a problem during testing. The value of mapzoom is between 1 and 19. The value 1 is the smallest value and the most area is displayed. The value 19 is the biggest value and the area is the smallest area is displayed. The effect of moving the same distance is different in the different size of the map. If the mapzoom is much smaller, the effect is more inconspicuous. For example, when the moving distance is set 0.005 and the value of mapzoom is 19, the effect is conformable. However, when the value of mapzoom is 1, you need to move several thousand times from Europe to Asia. We set different moving distances in different sizes of map by the best effect. When the mapzoom is smaller, the moving distance is bigger. Conversely, when the mapzoom is bigger, the moving distance is smaller.

4.3.2. Show Map
You move the map, or find another place according to the place name, during your using the mobile cell phone GPS navigation application.

4.3.3. Current Location
Using this function, you can return the current location map at once. When the application is running, we put the current coordinates which we get from GPS device into locationLatitude[0] parameter and locationLongitude[0] parameter. These two parameters always show the coordinates of the mobile’s current location. We only send the URL with the value of locationLatitude[0] and locationLongitude[0], getting current location map again.

4.3.4. Special Place Map
You can find the map which you want.

- Case A
Modifying coordinates in the URL, you can get a different map. If you want to get a map of a particular place, you can input the coordinates of that area and send the URL, but most people generally cannot remember such complicated coordinates. People only remember the name of place or street. Therefore, case A is not applied.

- **Case B**
  
  We use Geo API to solve the problem in Case A. We send the URL that includes the place name and street name, getting the latitude and longitude of that place. Then we put them into `locationLatitude` array and `locationLongitude` array respectively. We send the URL again. The URL includes coordinates this time and not the name of place or street. Finally, we get the map.

### 4.3.5. Make Marker

You can make a marker in the place where you want on the map.

- **Case A**

  There is a mark location parameter besides the map center location. Sending the URL with the marker coordinate, you get the marker map. You can set marker’s color, size and name.

  We create three marker coordinates:

  - `locationLongitude1, locationLatitude1`
  - `locationLongitude2, locationLatitude2`
  - `locationLongitude3, locationLatitude3`

  When we want to make a marker, we check whether the value of `locationLongitude1` and `locationLatitude1` are null. If not, we will check `locationLongitude2` and `locationLatitude2`. If they are null, we put the value of marker coordinate into `locationLongitude2` and `locationLatitude2`. However, this method has a disadvantage that only three markers can be made. We can create
more locationLongitudeX and locationLatitudeX. That is also the limit and the judgment program will be very complicated.

- **Case B**
  To solve the problem in Case A, we want that locationLongitudeX and locationLatitudeX can plus 1 automatically. Using an array can solve this problem. We define double type locationLongitude array and locationLatitude array, using for sentence to judge which variable is null from array 0 (locationLongitude[0] and locationLatitude[0]). We put marker coordinates into the array follow the order and the value of this array should be null first.

### 4.3.6. Send Message

This is an interesting function. You can share your location with your friends and know where your friends are. Firstly, we send an invitation and send our own coordinates of current location at the same time. How do we send a message? We need two things that address and content. The content is the value of the coordinates and the address includes phone number and port number. We use the setAddress(address) method and setPayloadText(content) method to set the phone number and message content. You can find them on reference Beginning J2ME Platform Chapter 11 Wireless Messaging API [7]. We send the content to the appointed phone. There is a mechanism for monitoring in the program. It always monitors the message. After you send the invitation, another phone can choose yes or no. The other party sends a feedback and his coordinates after he chooses to accept the invitation. How can we identify receiving invitation and feedback? We set a condition variable and send a message with this variable. When we send the invitation, the value of the condition is 1. When we send feedback, the value of the condition is 2. So we identify the value of the condition when receiving the message. Then, the mobile screen displays the corresponding interface.
4.3.7. Text Split

We referred to sending content before. The content includes the condition and the coordinates. How can we identify them? When sending content, we use @ symbol between parameters. The content is a string. After receiving the message, we create a string type Strbuf array and an index parameter. The index parameter expresses the start location and the value of it is 0. We check the location of the first @ symbol and put the number of the location into the startIndex parameter. The character string from the location index to the location startIndex is saved Strbuf[0]. Then the value of the index changes to startIndex+1. We check the location of the second @ symbol, and the value of the startIndex changes to the location of the next @ symbol. The character string from the location index to the location startIndex is saved Strbuf[1]. Repeat it.

4.3.8. Map Type

The user can look up different kinds of maps. For example, there are mobile maps, road maps, satellite maps, hybrid maps, terrain maps, mapmaker-road maps and mapmaker-hybrid maps. We only add the maptype parameter into the URL when we send the URL. Then we get the corresponding type map.

4.3.9. Tracking

When the user walks, the coordinates will be changed, so we need to orientation again and again. We call the package Java.Util.Timer and create a timer. We use the timer.schedule(track,1000, 5000) method and set the positioning function to execute it every 5 seconds once. We download the map over again after positioning.
5. Results

5.1. Main Interface

Figure 5.1: Within the main window, the Google static map is show on the screen. The title is your current position. On the bottom, there are two keypads on the screen. The keypad of the left side is the exit (exit program command). The keypad of the right side is Menu.

5.2. Menu

Figure 5.2: There are nine functions in our menu. We will introduce each function in the subchapter. In (a), functions are Show Map, Current Location, Special Place Map, Make Marker, Send Message and Route. In (b), functions are Map Type, Tracking and About.
5.3. Map Menu

Figure 5.3: When you press the “show map” keypad. All the keypad will be changed. The keypad will change to zoom in, zoom out, map up, map down, map left and map right.

5.3.1. Zoom In

When the users press “zoomin” on the keypad, the current position of the map will be amplified. The zoom’s level is from 0 to 19.

5.3.2. Zoom Out

When the users press the “zoomout” on the keypad, the current position of the map will be zoomed out. The zoom’s level is from 0 to 19.

5.3.3. Map Up

When the user presses “mapUp” on the keypad, the map will be move up and current position does not change. The distance of the map moves, depending on the zoom level.

5.3.4. Map Down

When the user presses “mapDown” on the keypad, the map will be moved down and the current position does not change. The distance of the map moves, depending on the zoom level.

5.3.5. Map Left

When the user presses “mapLeft” on the keypad, the map will be moved left and the current position does not change. The distance of the map moves, depending on the zoom level.
5. Results

5.3.6. Map Right

When the user presses “mapRight” on the keypad, the map will be moving right and the current position does not change. The distance of the map moves, depending on the zoom level.

5.4. Current Location

![Map Right Image]

Figure 5.4: When the user presses the “Current Location” on the keypad, the mark will display with the initial state. When we use cell phone simulator to set current location (e.g., 56.663, 12.879), the cell phone display the current location image (Halmstad University).

5.5. Special Place Map

![Special Place Map Images]

Figure 5.5: As you can see in (a), this interface is where we put the place name and road name,
and then we can make mark in these positions. Place name and Road name is the place name and road name where you want to search in the map. Choose size and Chose color is the size of the mark and the mark color which we want to mark it in the map. Input mark letter is the mark of the letter. For example, we want to search the road Nitaregatan, Halmstad, the mark color we choose is black and size is mid and the mark letter we put is 'd'. In (b), we find the Nitaregatan road in the Halmstad city map. The mark color is red and the mark letter is 'N'.

5.6. Make Marker

What we have done in this function is that we can add the position in the center of the map directly. We can choose the color and mark letter.

![Figure 5.6: (a) is the original map and we want to add one more mark in the center. The color we choose is orange and mark letter we put ‘g’. In (b), the result show in the middle of map.](image)

5.7. Send Message
5. Results

Figure 5.7: (a) is the send message interface. The phone number is your friend’s number that you want to send the message. In the 'Message Content' space you can write any text message you want to your friend. (b) is the receive message interface. The first line is the content (e.g., this is my location) that your friend sent to you. Choose size is the size of the mark. Choose Color is the color of the mark. Input mark letter (lowercase) is the letter of the mark. The letter must be lowercase.

In this function, we can display many friends’ positions in one map.

![Figure 5.7](image)

Figure 5.8: Blue sign “I” present my current location. The other two are my friends’ positions.

5.8. Map Type

![Figure 5.8](image)

Figure 5.9: This is the map type interface. We can choose the map type in this function:

- Mobile: Special type for the mobile map (high latitude and high longitude).
- Roadmap: Display the entire road name in the map.
- Satellite: You can see the planform of city in the map.
5. Results

- **Hybrid**: We can see the road name in the planform map.
- **Terrain**: This is the map for train.
- **Mapmaker - roadmap**: The type of the map is roadmap. We can mark the position in the center of the map directly.
- **Mapmaker - hybrid**: The type of the map is hybrid.

### 5.9. Tracking

![Figure 5.10](image)

Figure 5.10: We can locate the position every 5 seconds automatically. The mark will be refreshed on the screen. Comparing (a) and (b), you can find the marker's location changed.

### 5.10. About

![Figure 5.11](image)

Figure 5.11: about interface
6. Conclusion and Future Work

6.1. Conclusion

The mobile cell phone GPS navigation application is an application for mobile products. The project has achieved the function of sending the locations between two mobiles, besides the mobile GPS’s basic functions. This makes sense for the mobile GPS applications in the market at present. This project has completed all functions that we expect, except for route function. It is not yet perfected, but it could be improved in the future.

6.2. Future Work

To display the map with our software is based on the J2ME Form component. We could change the display map on the canvas, thus we can also use the direction keypad on the cell phone to move the map and zoom in or zoom out the map. Also, for map type, now we only use Google map, and by changing the URL address, the software can use the other MAP server API. With regard to the route function, as we mentioned above, if we had enough time we could develop the data analysis server and then the route function could be completed.
Bibliography


http://code.google.com/intl/en/apis/maps/documentation/overlays.html,


28
http://en.wikipedia.org/wiki/Assisted_GPS,
2008

[7] Sing Li and Jonathan Knudsen,
Beginning J2METM Platform: From Novice to Professional,
pp.145 ~ 149, pp.170~180

[8] playfish@WEB, Http Client + Jericho HTML Parser,
http://playfish.javaeye.com/blog/150184,
2007-12-22

http://www-users.cs.umn.edu/~czhou/docs/jsr179/lapi/javax/microedition/location/package-summary.html,
2008
Appendix

MapLoader.java

package Gps;

import Gps.UI.AboutUI;
import Gps.UI.MapTypeUI;
import Gps.UI.MarkCurrentPlaceUI;
import Gps.UI.SendAcknowledgementUI;
import Gps.UI.SpecialPlaceMapUI;
import Gps.UI.SendMessageUI;
import Gps.Task.TrackPositionTask;
import java.io.*;
import java.util.Timer;
import java.util.Vector;
import javax.microedition.io.*;
import javax.microedition.lcdui.*;
import javax.microedition.location.Coordinates;
import javax.microedition.location.Criteria;
import javax.microedition.location.Location;
import javax.microedition.location.LocationException;
import javax.microedition.location.LocationProvider;
import javax.microedition.midlet.*;
import javax.microedition.wireless.messaging.BinaryMessage;
import javax.microedition.wireless.messaging.Message;
import javax.microedition.wireless.messaging.MessageConnection;
import javax.microedition.wireless.messaging.MessageListener;
import javax.microedition.wireless.messaging.TextMessage;

/**
* @author Sha Mao Xuan & Xie Jun & Xu Xiao Lin
*/

public class MapLoader extends MIDlet implements CommandListener, MessageListener,
Runnable {

    private Display mDisplay;
    private Form mForm;

    /***************Main Form*************/
    private Command ShowCurrentLocation = new Command("Current Location", Command.SCREEN, 1);
    private Command ShowMap = new Command("ShowMap", Command.SCREEN, 1);
    private Command SpecialPlaceMap = new Command("Special Place Map",
        Command.SCREEN, 1);
    private Command MakeMarker = new Command("Make Marker", Command.SCREEN, 1);
    private Command SendMessage = new Command("Send Message", Command.SCREEN, 1);
    private Command Route = new Command("Route", Command.SCREEN, 1);
private Command MapType = new Command("Map Type", Command.SCREEN, 1);
private Command TrackPosition = new Command("Tracking", Command.SCREEN, 1);
private Command StopTracking = new Command("Stop Tracking", Command.SCREEN, 1);
private Command About = new Command("About", Command.SCREEN, 1);
private Command Exit = new Command("Exit", Command.BACK, 0);

/**************************ShowMap Form**************************/
private Command zoomIn = new Command("zoomin", Command.SCREEN, 1);
private Command zoomOut = new Command("zoomout", Command.SCREEN, 1);
private Command mapUp = new Command("mapUp", Command.SCREEN, 1);
private Command mapDown = new Command("mapDown", Command.SCREEN, 1);
private Command mapLeft = new Command("mapLeft", Command.SCREEN, 1);
private Command mapRight = new Command("mapRight", Command.SCREEN, 1);
private Command Back = new Command("Back", Command.BACK, 0);

/*********************AboutUI Form intial**************************/
private AboutUI aboutUI = null;
private SpecialPlaceMapUI specialMapUI = null;
private SendMessageUI sendMessageUI = null;
private SendAcknowledgementUI sendAcknowledgementUI = null;
private MapTypeUI mapTypeUI = null;
private MarkCurrentPlaceUI markCurrentPlaceUI = null;

/********************Open MessageConnection***********************/
public MessageConnection receiveConn = null;
private boolean done = false;
public MessageReader reader = null;
public Message message = null;
private ImageItem imageItem = new ImageItem(null, null, ImageItem.LAYOUT_DEFAULT, null);
public double mapCenterLongitude;
public double mapCenterLatitude;
public double[] locationLatitude = new double[50];
public double[] locationLongitude = new double[50];
public String[] markColor = new String[50];
public String[] markName = new String[50];
public String[] markSize = new String[50];
public String maptype = null;

public String phononenumber;
public String Place;
public String Road;

private String[] StrbufPlace;
int mapzoom = 10;
int mapzoomMax = 19;
int mapzoomMin = 1;
Thread t = new Thread(this);

private Timer timer;
private TrackPositionTask track;
private int TrackCondition = 1;

public MapLoader() {
    mForm = new Form("Connecting...");
    mForm.setCommandListener(this);
}
public void startApp() {
    if (mDisplay == null) {
        mDisplay = Display.getDisplay(this);
    }
    mDisplay.setCurrent(mForm);
    setCurrentLocation();
    /*************************Main Form**************************/
    mForm.addCommand(ShowMap);
    mForm.addCommand(ShowCurrentLocation);
    mForm.addCommand(SpecialPlaceMap);
    mForm.addCommand(MakeMarker);
    mForm.addCommand(SendMessage);
    mForm.addCommand(Route);
    mForm.addCommand(MapType);
    mForm.addCommand(TrackPosition);
    mForm.addCommand(About);
    mForm.addCommand(Exit);
    t.start();
    initMessageServer();
}

private void initMessageServer() {
    done = false;
    String url = "sms://:" + this.getAppProperty("port");
    System.out.println(url);
    try {
        receiveConn = (MessageConnection) Connector.open(url);
        receiveConn.setMessageListener(this);
        reader = new MessageReader();
        new Thread(reader).start();
    } catch (IOException ex) {
        ex.printStackTrace();
    }
}

public void pauseApp() {
}

public void destroyApp(boolean unconditional) {
    notifyDestroyed();
    try {
        receiveConn.close();
        t.interrupt();
    } catch (IOException ex) {
        ex.printStackTrace();
    }
}

public void backToMain() {
    mDisplay.setCurrent(mForm);
    run();
}

public void commandAction(Command c, Displayable s) {
    if (c == Exit) {

destroyApp(true);

} else if (c == ShowMap) {
    /****************Form Command Change***************************/
    /***************Main Form------ShowMap Form*******************/
    mForm.removeCommand(ShowMap);
    mForm.removeCommand(ShowCurrentLocation);
    mForm.removeCommand(SpecialPlaceMap);
    mForm.removeCommand(MakeMarker);
    mForm.removeCommand(SendMessage);
    mForm.removeCommand(Route);
    mForm.removeCommand(TrackPosition);
    mForm.removeCommand(StopTracking);
    mForm.removeCommand(About);
    mForm.removeCommand(MapType);
    mForm.removeCommand(Exit);
    mForm.addCommand(zoomIn);
    mForm.addCommand(zoomOut);
    mForm.addCommand(mapUp);
    mForm.addCommand(mapDown);
    mForm.addCommand(mapLeft);
    mForm.addCommand(mapRight);
    mForm.addCommand(Back);

} else if( c == TrackPosition){
    TrackCondition = 1;
    trackposition();
    mForm.removeCommand(ShowMap);
    mForm.removeCommand(ShowCurrentLocation);
    mForm.removeCommand(SpecialPlaceMap);
    mForm.removeCommand(MakeMarker);
    mForm.removeCommand(SendMessage);
    mForm.removeCommand(Route);
    mForm.removeCommand(TrackPosition);
    mForm.removeCommand(About);
    mForm.removeCommand(MapType);
    mForm.removeCommand(Exit);
    mForm.addCommand(ShowMap);
    mForm.addCommand(ShowCurrentLocation);
    mForm.addCommand(SpecialPlaceMap);
    mForm.addCommand(MakeMarker);
    mForm.addCommand(SendMessage);
    mForm.addCommand(Route);
    mForm.addCommand(TrackPosition);
    mForm.addCommand(About);
    mForm.addCommand(MapType);
    mForm.addCommand(Exit);

} else if(c == StopTracking){
    TrackCondition = 2;
    timer.cancel();
    mForm.removeCommand(ShowMap);
    mForm.removeCommand(ShowCurrentLocation);
    mForm.removeCommand(SpecialPlaceMap);
mForm.removeCommand(MakeMarker);
mForm.removeCommand(SendMessage);
mForm.removeCommand(Route);
mForm.removeCommand(StopTracking);
mForm.removeCommand(About);
mForm.removeCommand(MapType);
mForm.removeCommand(Exit);
mForm.addCommand(ShowMap);
mForm.addCommand(ShowCurrentLocation);
mForm.addCommand(SpecialPlaceMap);
mForm.addCommand(MakeMarker);
mForm.addCommand(SendMessage);
mForm.addCommand(Route);
mForm.addCommand(MapType);
mForm.addCommand(TrackPosition);
mForm.addCommand(About);
mForm.addCommand(Exit);

} else if(c == ShowCurrentLocation){
    CleanAllLocation();
    setCurrentLocation();
    run();
} else if (c == SpecialPlaceMap) {
    specialMapUI = new SpecialPlaceMapUI(MapLoader.this);
    mDisplay.setCurrent(specialMapUI);
} else if (c == MakeMarker) {
    markCurrentPlaceUI = new MarkCurrentPlaceUI(MapLoader.this);
    mDisplay.setCurrent(markCurrentPlaceUI);
} else if (c == SendMessage) {
    sendMessageUI = new SendMessageUI(this);
    mDisplay.setCurrent(sendMessageUI);
} else if (c == Route) {

} else if (c == MapType) {
    mapTypeUI = new MapTypeUI(MapLoader.this);
    mDisplay.setCurrent(mapTypeUI);
} else if (c == About) {
    aboutUI = new AboutUI(MapLoader.this);
    mDisplay.setCurrent(aboutUI);
} else if (c == Back) {
    /*******************Form Command Change*************************/
    /***************ShowMap Form ------Main Form*********************/
    mForm.removeCommand(zoomIn);
    mForm.removeCommand(zoomOut);
    mForm.removeCommand(mapUp);
    mForm.removeCommand(mapDown);
    mForm.removeCommand(mapLeft);
    mForm.removeCommand(mapRight);
    mForm.removeCommand(Back);
    mForm.addCommand(ShowMap);
    mForm.addCommand(ShowCurrentLocation);
    mForm.addCommand(SpecialPlaceMap);
    mForm.addCommand(MakeMarker);
    mForm.addCommand(SendMessage);
    mForm.addCommand(Route);
    mForm.addCommand(MapType);
switch track();
mForm.addCommand(About);
mForm.addCommand(Exit);
} else if (c == zoomIn) {
    if (mapzoom < mapzoomMax) {
        mapzoom = mapzoom + 1;
        run();
    }
} else if (c == zoomOut) {
    if (mapzoom > mapzoomMin) {
        mapzoom = mapzoom - 1;
        run();
    }
} else if (c == mapRight) {
    switch (mapzoom) {
    case 1:
        mapCenterLongitude = mapCenterLongitude + 12;
        run();
        break;
    case 2:
        mapCenterLongitude = mapCenterLongitude + 4;
        run();
        break;
    case 3:
        mapCenterLongitude = mapCenterLongitude + 3;
        run();
        break;
    case 4:
        mapCenterLongitude = mapCenterLongitude + 2;
        run();
        break;
    case 5:
        mapCenterLongitude = mapCenterLongitude + 1.024;
        run();
        break;
    case 6:
        mapCenterLongitude = mapCenterLongitude + 0.768;
        run();
        break;
    case 7:
        mapCenterLongitude = mapCenterLongitude + 0.512;
        run();
        break;
    case 8:
        mapCenterLongitude = mapCenterLongitude + 0.256;
        run();
        break;
    case 9:
        mapCenterLongitude = mapCenterLongitude + 0.128;
        run();
        break;
    case 10:
        mapCenterLongitude = mapCenterLongitude + 0.064;
        run();
        break;
    case 11:
mapCenterLongitude = mapCenterLongitude + 0.032;
run();
break;
case 12:
    mapCenterLongitude = mapCenterLongitude + 0.016;
    run();
    break;
case 13:
    mapCenterLongitude = mapCenterLongitude + 0.008;
    run();
    break;
case 14:
    mapCenterLongitude = mapCenterLongitude + 0.004;
    run();
    break;
case 15:
    mapCenterLongitude = mapCenterLongitude + 0.002;
    run();
    break;
case 16:
    mapCenterLongitude = mapCenterLongitude + 0.001;
    run();
    break;
case 17:
    mapCenterLongitude = mapCenterLongitude + 0.0005;
    run();
    break;
case 18:
    mapCenterLongitude = mapCenterLongitude + 0.00025;
    run();
    break;
case 19:
    mapCenterLongitude = mapCenterLongitude + 0.000125;
    run();
    break;
default:
    run();
}
} else if (c == mapLeft) {
    switch (mapzoom) {
    case 1:
        mapCenterLongitude = mapCenterLongitude - 12;
        run();
        break;
    case 2:
        mapCenterLongitude = mapCenterLongitude - 4;
        run();
        break;
    case 3:
        mapCenterLongitude = mapCenterLongitude - 3;
        run();
        break;
    case 4:
        mapCenterLongitude = mapCenterLongitude - 2;
        run();
        break;
case 5:
    mapCenterLongitude = mapCenterLongitude - 1.024;
    run();
    break;

case 6:
    mapCenterLongitude = mapCenterLongitude - 0.768;
    run();
    break;

case 7:
    mapCenterLongitude = mapCenterLongitude - 0.512;
    run();
    break;

case 8:
    mapCenterLongitude = mapCenterLongitude - 0.256;
    run();
    break;

case 9:
    mapCenterLongitude = mapCenterLongitude - 0.128;
    run();
    break;

case 10:
    mapCenterLongitude = mapCenterLongitude - 0.064;
    run();
    break;

case 11:
    mapCenterLongitude = mapCenterLongitude - 0.032;
    run();
    break;

case 12:
    mapCenterLongitude = mapCenterLongitude - 0.016;
    run();
    break;

case 13:
    mapCenterLongitude = mapCenterLongitude - 0.008;
    run();
    break;

case 14:
    mapCenterLongitude = mapCenterLongitude - 0.004;
    run();
    break;

case 15:
    mapCenterLongitude = mapCenterLongitude - 0.002;
    run();
    break;

case 16:
    mapCenterLongitude = mapCenterLongitude - 0.001;
    run();
    break;

case 17:
    mapCenterLongitude = mapCenterLongitude - 0.0005;
    run();
    break;

case 18:
    mapCenterLongitude = mapCenterLongitude - 0.00025;
    run();
    break;
case 19:
    mapCenterLongitude = mapCenterLongitude - 0.000125;
    run();
    break;
default:
    run();
}
} else if (c == mapDown) {
    switch (mapzoom) {
    case 1:
        mapCenterLatitude = mapCenterLatitude - 12;
        run();
        break;
    case 2:
        mapCenterLatitude = mapCenterLatitude - 4;
        run();
        break;
    case 3:
        mapCenterLatitude = mapCenterLatitude - 3;
        run();
        break;
    case 4:
        mapCenterLatitude = mapCenterLatitude - 2;
        run();
        break;
    case 5:
        mapCenterLatitude = mapCenterLatitude - 1.024;
        run();
        break;
    case 6:
        mapCenterLatitude = mapCenterLatitude - 0.768;
        run();
        break;
    case 7:
        mapCenterLatitude = mapCenterLatitude - 0.512;
        run();
        break;
    case 8:
        mapCenterLatitude = mapCenterLatitude - 0.256;
        run();
        break;
    case 9:
        mapCenterLatitude = mapCenterLatitude - 0.128;
        run();
        break;
    case 10:
        mapCenterLatitude = mapCenterLatitude - 0.064;
        run();
        break;
    case 11:
        mapCenterLatitude = mapCenterLatitude - 0.032;
        run();
        break;
    case 12:
        mapCenterLatitude = mapCenterLatitude - 0.016;
        run();
break;
case 13:
    mapCenterLatitude = mapCenterLatitude - 0.008;
    run();
    break;
case 14:
    mapCenterLatitude = mapCenterLatitude - 0.004;
    run();
    break;
case 15:
    mapCenterLatitude = mapCenterLatitude - 0.002;
    run();
    break;
case 16:
    mapCenterLatitude = mapCenterLatitude - 0.001;
    run();
    break;
case 17:
    mapCenterLatitude = mapCenterLatitude - 0.0005;
    run();
    break;
case 18:
    mapCenterLatitude = mapCenterLatitude - 0.00025;
    run();
    break;
case 19:
    mapCenterLatitude = mapCenterLatitude - 0.000125;
    run();
    break;
default:
    run();
} 
} else if (c == mapUp) { 
    switch (mapzoom) {
    case 1:
        mapCenterLatitude = mapCenterLatitude + 12;
        run();
        break;
    case 2:
        mapCenterLatitude = mapCenterLatitude + 4;
        run();
        break;
    case 3:
        mapCenterLatitude = mapCenterLatitude + 3;
        run();
        break;
    case 4:
        mapCenterLatitude = mapCenterLatitude + 2;
        run();
        break;
    case 5:
        mapCenterLatitude = mapCenterLatitude + 1.024;
        run();
        break;
    case 6:
        mapCenterLatitude = mapCenterLatitude + 0.768;
run();
break;
case 7:
    mapCenterLatitude = mapCenterLatitude + 0.512;
run();
break;
case 8:
    mapCenterLatitude = mapCenterLatitude + 0.256;
run();
break;
case 9:
    mapCenterLatitude = mapCenterLatitude + 0.128;
run();
break;
case 10:
    mapCenterLatitude = mapCenterLatitude + 0.064;
run();
break;
case 11:
    mapCenterLatitude = mapCenterLatitude + 0.032;
run();
break;
case 12:
    mapCenterLatitude = mapCenterLatitude + 0.016;
run();
break;
case 13:
    mapCenterLatitude = mapCenterLatitude + 0.008;
run();
break;
case 14:
    mapCenterLatitude = mapCenterLatitude + 0.004;
run();
break;
case 15:
    mapCenterLatitude = mapCenterLatitude + 0.002;
run();
break;
case 16:
    mapCenterLatitude = mapCenterLatitude + 0.001;
run();
break;
case 17:
    mapCenterLatitude = mapCenterLatitude + 0.0005;
run();
break;
case 18:
    mapCenterLatitude = mapCenterLatitude + 0.00025;
run();
break;
case 19:
    mapCenterLatitude = mapCenterLatitude + 0.000125;
run();
break;
default:
    run();
run();
public void setCurrentLocation() {
    try {
        Criteria cr = new Criteria();
        cr.setHorizontalAccuracy(500);
        LocationProvider lp;
        lp = LocationProvider.getInstance(cr);
        Location l = lp.getLocation(60);
        Coordinates c = l.getQualifiedCoordinates();
        if (c != null) {
            mapCenterLatitude = c.getLatitude();
            mapCenterLongitude = c.getLongitude();
            locationLatitude[0] = c.getLatitude();
            locationLongitude[0] = c.getLongitude();
            markSize[0] = "mid";
            markColor[0] = "blue";
            markName[0] = "i";
        } else {
            throw new RuntimeException("Failed to retrieve current location.");
        }
    } catch (LocationException ex) {
        ex.printStackTrace();
    } catch (InterruptedException ex) {
        ex.printStackTrace();
    }
}

public void run() {
    ShowMap();
}

public void Geo() {
    HttpConnection hc = null;
    DataInputStream in = null;
    try {
        String url4 = this.getAppProperty("Google_Geo_URL")
            + Place
            + ","
            + Road
            + ";output=csv"
            + ";key="+this.getAppProperty("Google_Maps_Key");
        hc = (HttpConnection) Connector.open(url4);
        int length = (int) hc.getLength();
        byte[] data = null;
        if (length != -1) {
            data = new byte[length];
            in = new DataInputStream(hc.openDataInputStream());
            in.readFully(data);
        } else {
            int chunkSize = 512;
            int index = 0;
            int readLength = 0;
            in = new DataInputStream(hc.openInputStream());
        }
    } catch (IOException ex) {
        ex.printStackTrace();
    } catch (InterruptedException ex) {
        ex.printStackTrace();
    }
}
Appendix – MapLoader.java

data = new byte[chunkSize];
do {
    if (data.length < index + chunkSize) {
        byte[] newData = new byte[index + chunkSize];
        System.arraycopy(data, 0, newData, 0, data.length);
        data = newData;
    }
    readLength = in.read(data, index, chunkSize);
    index += readLength;
} while (readLength == chunkSize);
length = index;

String string = new String(data, "UTF-8");
StrbufPlace = split(string, ",");
int i;
for(i=0;locationLatitude[i]!=0||locationLongitude[i]!=0;i++){
    locationLatitude[i] = Double.parseDouble(StrbufPlace[2]);
    locationLongitude[i] = Double.parseDouble(StrbufPlace[3]);
    mapCenterLatitude = Double.parseDouble(StrbufPlace[2]);
    mapCenterLongitude = Double.parseDouble(StrbufPlace[3]);
    markSize[i] = specialMapUI.markSSPM;
    markColor[i] = specialMapUI.markCSPM;
    markName[i] = specialMapUI.markNSPM;
    run();
} catch (IOException ioe) {
    StringItem stringItem = new StringItem(null, ioe.toString());
} catch (IllegalStateException e) {
    e.printStackTrace();
} finally {
    try {
        if (in != null) {
            in.close();
        }
        if (hc != null) {
            hc.close();
        }
    } catch (IOException ioe) {
    }
}

public void ShowMap() {
    HttpConnection hc = null;
    DataInputStream dis = null;
    try {
        String s = "";
        for(int i=0;locationLatitude[i]!=0||locationLongitude[i]!=0;i++){
            s = s + "&markers=" + locationLatitude[i] + "," + locationLongitude[i] + "," + markSize[i] + markColor[i] + markName[i];
        }
        String u = this.getAppProperty("Google_Static_Maps_URL")
            + "center=" + mapCenterLatitude + "," + mapCenterLongitude + s
            + "&zoom=" + mapzoom
            + "&size=240x270"
Appendix – MapLoader.java

```java
+ "&maptype=" + maptype
+ "&key=" + this.getAppProperty("Google_Maps_Key");

hc = (HttpConnection) Connector.open(u);
int length = (int) hc.getLength();
byte[] data = null;
if (length != -1) {
    data = new byte[length];
    dis = new DataInputStream(hc.openDataInputStream());
    dis.readFully(data);
} else {
    int chunkSize = 512;
    int index = 0;
    int readLength = 0;
    dis = new DataInputStream(hc.openInputStream());
    data = new byte[chunkSize];
    do {
        if (data.length < index + chunkSize) {
            byte[] newData = new byte[index + chunkSize];
            System.arraycopy(data, 0, newData, 0, data.length);
            data = newData;
        }
        readLength = dis.read(data, index, chunkSize);
        index += readLength;
    } while (readLength == chunkSize);
    length = index;
}
Image image = Image.createImage(data, 0, length);
imageItem.setImage(image);
mForm.deleteAll();
mForm.append(imageItem);
mForm.setTitle("You current Location" + 
    "(float)mapCenterLatitude","(float)mapCenterLongitude");
//mForm.setTitle("Welcome to use FindYourFriend Software");
}
```

```java
} catch (IOException ioe) {
    StringItem stringItem = new StringItem(null, ioe.toString());
    mForm.append(stringItem);
    mForm.setTitle("Your current Location" + 
        "(float)mapCenterLatitude", "(float)mapCenterLongitude");
    //mForm.setTitle("Welcome to use FindYourFriend Software");
} catch (IllegalStateException e) {
    e.printStackTrace();
} finally {
    try {
        if (dis != null) {
            dis.close();
        }
        if (hc != null) {
            hc.close();
        }
    } catch (IOException ioe) {
    }
}

public void notifyIncomingMessage(MessageConnection messconn) {
    
}
if (messconn == receiveConn) {
    reader.handleMessage();
}

public class MessageReader implements Runnable, CommandListener {
    private int i;
    private int pendingMessage = 0;
    private String senderAddress;
    private String messagecontent;
    private Form content;
    private int condition;
    private Command comfirmCommand1 = new Command("Comfirm", Command.OK, 1);
    private Command comfirmCommand2 = new Command("Comfirm", Command.OK, 1);
    private Command cancelCommand = new Command("Cancel", Command.BACK, 1);
    private String[] Strbuf;
    private ChoiceGroup SizeChoice = new ChoiceGroup("Choose Size", Choice.POPUP);
    private ChoiceGroup ColorChoiceL = new ChoiceGroup("Choose Color", Choice.POPUP);
    private String Info1 = "Your friend want to share location with you. Do you agree?";
    private String Info2 = "Your friend agree to share location with you.");
    private TextField markN = null;

    public MessageReader() {
        content = new Form("SMS Receive");
        markN = new TextField("Input mark letter(lowercase)(a~z,0~9)", "", 1,
                               TextField.ANY);
        content.setCommandListener(this);
    }

    public void run() {
        while (!done) {
            synchronized (this) {
                if (pendingMessage == 0) {
                    try {
                        wait();
                    } catch (InterruptedException ex) {
                        ex.printStackTrace();
                    }
                }
                pendingMessage--;
            }
            try {
                message = receiveConn.receive();
                if (message != null) {
                    senderAddress = message.getAddress();
                    phonenumber = senderAddress;
                    content.setTitle("From: " + senderAddress);
                    if (message instanceof TextMessage) {
                        Strbuf = split(((TextMessage) message).getPayloadText(), "@");
                        for(i=0;locationLatitude[i]!=0||locationLongitude[i]!=0;i++){
                        }
                        locationLongitude[i] = Double.parseDouble(Strbuf[0]);
                        locationLatitude[i] = Double.parseDouble(Strbuf[1]);
                        mapCenterLongitude = Double.parseDouble(Strbuf[0]);
                        mapCenterLatitude = Double.parseDouble(Strbuf[1]);
                    }
                }
            }
        }
    }
}
messagecontent = Strbuf[2];
condition = Integer.parseInt(Strbuf[3]);
switch (condition) {
    case 1:
        content.append(Info1);
        content.append(messagecontent);
        content.append(SizeChoice);
        SizeChoice.append("mid", null);
        SizeChoice.append("tiny", null);
        SizeChoice.append("small", null);
        content.append(ColorChoiceL);
        ColorChoiceL.append("black", null);
        ColorChoiceL.append("brown", null);
        ColorChoiceL.append("green", null);
        ColorChoiceL.append("purple", null);
        ColorChoiceL.append("yellow", null);
        ColorChoiceL.append("blue", null);
        ColorChoiceL.append("gray", null);
        ColorChoiceL.append("orange", null);
        ColorChoiceL.append("red", null);
        ColorChoiceL.append("white", null);
        content.append(markN);
        content.removeCommand(comfirmCommand2);
        content.addCommand(comfirmCommand1);
        content.addCommand(cancelCommand);
        break;
    case 2:
        content.append(Info2);
        content.append(messagecontent);
        content.append(SizeChoice);
        SizeChoice.append("mid", null);
        SizeChoice.append("tiny", null);
        SizeChoice.append("small", null);
        content.append(ColorChoiceL);
        ColorChoiceL.append("black", null);
        ColorChoiceL.append("brown", null);
        ColorChoiceL.append("green", null);
        ColorChoiceL.append("purple", null);
        ColorChoiceL.append("yellow", null);
        ColorChoiceL.append("blue", null);
        ColorChoiceL.append("gray", null);
        ColorChoiceL.append("orange", null);
        ColorChoiceL.append("red", null);
        ColorChoiceL.append("white", null);
        content.append(markN);
        content.removeCommand(comfirmCommand1);
        content.addCommand(comfirmCommand2);
        break;
} else {
    StringBuffer buf = new StringBuffer();
    byte[] data = ((BinaryMessage) message).getPayloadData();
    for (int i = 0; i < data.length; i++) {
        int intData = (int) data[i] & 0xFF;
        if (intData < 0x10) {
buf.append("0");
}
buf.append(Integer.toHexString(intData));
buf.append(" ");
}
content.append(buf.toString());
}
mDisplay.setCurrent(content);
}
} catch (IOException e) {
}
}

public synchronized void handleMessage() {
pendingMessage++;
notify();
}

public void commandAction(Command c, Displayable d) {
if (c == confirmCommand1) {
    markSize[i] = SizeChoice.getString(SizeChoice.getSelectedIndex());
    markColor[i] = ColorChoiceL.getString(ColorChoiceL.getSelectedIndex());
    markName[i] = markN.getString();
    markN.setString("*");
    SizeChoice.deleteAll();
    ColorChoiceL.deleteAll();
    content.deleteAll();
    sendAcknowledgementUI = new SendAcknowledgementUI(MapLoader.this);
    mDisplay.setCurrent(sendAcknowledgementUI);  
}
else if(c == confirmCommand2){
    markSize[i] = SizeChoice.getString(SizeChoice.getSelectedIndex());
    markColor[i] = ColorChoiceL.getString(ColorChoiceL.getSelectedIndex());
    markName[i] = markN.getString();
    markN.setString("*");
    SizeChoice.deleteAll();
    ColorChoiceL.deleteAll();
    content.deleteAll();
    backToMain();
}
else if (c == cancelCommand) {
    markN.setString("*");
    SizeChoice.deleteAll();
    ColorChoiceL.deleteAll();
    content.deleteAll();
    backToMain();
}
}

public void switchtrack() {
    switch (TrackCondition){
    case 1:
        mForm.addCommand(TrackPosition);
        break;
    }
case 2:
    mForm.addCommand(StopTracking);
    break;
}

public double getLocationLongitude() {
    return locationLongitude[0];
}

public double getLocationLatitude() {
    return locationLatitude[0];
}

public void trackPosition() {
    try{
        track = new TrackPositionTask(MapLoader.this);
        timer = new Timer();
        timer.schedule(track, 1000, 5000);
    }catch(Exception e){
    }
}

public void cleanAllLocation(){
    mapCenterLongitude = 0;
    mapCenterLatitude = 0;
}

private static String[] split(String original, String regex) {
    int startIndex = 0;
    Vector v = new Vector();
    String[] str = null;
    int index = 0;
    startIndex = original.indexOf(regex);
    while (startIndex < original.length() && startIndex != -1) {
        String temp = original.substring(index, startIndex);
        System.out.println(" "+startIndex);
        v.addElement(temp);
        index = startIndex + regex.length();
        startIndex = original.indexOf(regex, startIndex + regex.length());
    }
    v.addElement(original.substring(index + 1 - regex.length()));
    str = new String[v.size()];
    for (int i = 0; i < v.size(); i++) {
        str[i] = (String) v.elementAt(i);
    }
    return str;
}
TrackPositionTask.java

```java
package Gps.Task;

import Gps.MapLoader;
import java.util.TimerTask;

/**
 * @author Sha Mao Xuan
 */
public class TrackPositionTask extends TimerTask {
    private MapLoader midlet;

    public TrackPositionTask(MapLoader midlet) {
        this.midlet = midlet;
    }

    public void run() {
        midlet.setCurrentLocation();
        midlet.run();
    }
}
```

SpecialPlaceMapUI.java

```java
package Gps.UI;

import Gps.MapLoader;
import javax.microedition.lcdui.Choice;
import javax.microedition.lcdui.ChoiceGroup;
import javax.microedition.lcdui.Command;
import javax.microedition.lcdui.CommandListener;
import javax.microedition.lcdui.Displayable;
import javax.microedition.lcdui.Form;
import javax.microedition.lcdui.TextField;

/**
 * @author Sha Mao Xuan & Xie Jun & Xu Xiao Lin
 */
public class SpecialPlaceMapUI extends Form implements CommandListener{
    private MapLoader midlet;
    private Command backCommand = new Command("Back",Command.BACK,0);
    private Command SpecialDownloadMap = new Command("DownloadMap",Command.SCREEN,1);
    private String SpecialMapInfo = new String("Please Enter Location");
    public TextField SpecialPlaceName = new TextField("Place Name", null , 255 ,
    TextField.ANY);
    public TextField SpecialRoadName = new TextField("Road Name", null , 255 ,
```
Appendix – SpecialPlaceMapUI.java

TextField.ANY);
private ChoiceGroup ColorChoice = new ChoiceGroup("Choose Color",Choice.POPUP);
private ChoiceGroup SizeChoice = new ChoiceGroup("Choose Size",Choice.POPUP);
private TextField markN = null;
public String markNSPM = "";
public String markCSPM = "";
public String markSSPM = "";

public SpecialPlaceMapUI(MapLoader midlet) {
    super("Special Location Map");
    this.midlet = midlet;
    markN = new TextField("Input mark letter(lowercase)(a~z,0~9)*", ",", 1, TextField.ANY);
    this.append(SpecialMapInfo);
    this.append(SpecialPlaceName);
    this.append(SpecialRoadName);
    this.append(SizeChoice);
    SizeChoice.append("mid", null);
    SizeChoice.append("tiny", null);
    SizeChoice.append("small", null);
    this.append(ColorChoice);
    ColorChoice.append("black", null);
    ColorChoice.append("brown", null);
    ColorChoice.append("green", null);
    ColorChoice.append("purple", null);
    ColorChoice.append("yellow", null);
    ColorChoice.append("blue", null);
    ColorChoice.append("gray", null);
    ColorChoice.append("orange", null);
    ColorChoice.append("red", null);
    ColorChoice.append("white", null);
    this.append(markN);
    this.addCommand(backCommand);
    this.addCommand(SpecialDownloadMap);
    this.setCommandListener(this);
}

public void commandAction(Command cmd,Displayable displayable){
    if(cmd == backCommand){
        midlet.backToMain();
    }
    else if(cmd == SpecialDownloadMap){
        midlet.Place = SpecialPlaceName.getString();
        midlet.Road = SpecialRoadName.getString();
        markSSPM = SizeChoice.getString(SizeChoice.getSelectedIndex());
        markCSPM = ColorChoice.getString(ColorChoice.getSelectedIndex());
        markNSPM = markN.getString();
        midlet.backToMain();
        midlet Geo();
    }
}
package Gps.UI;

import Gps.MapLoader;
import javax.microedition.io.Connector;
import javax.microedition.lcdui.*;
import javax.wireless.messaging.*;

/**
 * @author Sha Mao Xuan & Xie Jun & Xu Xiao Lin
 */

public class SendMessageUI extends Form implements CommandListener,Runnable {

    private MapLoader midlet = null;
    private TextField address = null;
    private TextField message = null;
    private String Info = "Do you want to share your location?";
    private String number = "";
    private String content1 = "";
    private String content2 = "";
    private String content3 = "";
    private int content4;
    public static final Command sendCommand = new Command("Send",Command.OK,1);
    public static final Command backCommand = new Command("Back",Command.BACK,2);

    public SendMessageUI(MapLoader midlet) {
        super("Send Message");
        this.midlet = midlet;
        address = new TextField("Phone Number", "", 20, TextField.PHONENUMBER);
        this.append(address);
        this.append(Info);
        message = new TextField("Message Content","",255,TextField.ANY);
        this.append(message);
        this.addCommand(sendCommand);
        this.addCommand(backCommand);
        this.setCommandListener(this);
    }

    public void commandAction(Command cmd,Displayable displayable){
        if(cmd == backCommand){
            midlet.backToMain();
        }else if(cmd == sendCommand){
            number = address.getString();
            content1 = midlet.getlocationLongitude()+"";
            content2 = midlet.getlocationLatitude()+"";
            content3 = message.getString();
            content4 = 1;
            new Thread(this).start();
        }
    }

    public void run(){
        String address = "sms://"+number+":"+midlet.getAppProperty("port");
        MessageConnection mconn = null;
        }
try{
    mconn = (MessageConnection)Connector.open(address);
    TextMessage tm =
        (TextMessage)mconn.newMessage(MessageConnection.TEXT_MESSAGE);
    tm.setAddress(address);  
    tm.setPayloadText(content1+"@"+content2 +"@"+content3+"@"+content4);  
    mconn.send(tm);  
    midlet.backToMain();
}catch(Exception ex){
    ex.printStackTrace();
}
}

SendAcknowledgementUI.java

package Gps.UI;

import Gps.MapLoader;
import javax.microedition.io.Connector;
import javax.microedition.lcdui.*;
import javax.wireless.messaging.*;

/**
 * @author Sha Mao Xuan & Xie Jun & Xu Xiao Lin
 */

public class SendAcknowledgementUI extends Form implements CommandListener,Runnable {

    private MapLoader midlet = null;
    private String number = "";
    private String Info = "";
    private TextField message = null;
    private String content1 = "";
    private String content2 = "";
    private String content3 = "";
    private int content4;
    public static final Command sendCommand = new Command("Send",Command.OK,1);
    public static final Command backCommand = new Command("Back",Command.BACK,2);

    public SendAcknowledgementUI(MapLoader midlet) {
        super("Send Message");
        this.midlet = midlet;
        number = midlet.phonenumber;
        Info = "Send the your position to " + number + ".
Are you sure?";
        this.append(Info);
        message = new TextField("Message Content","",255,TextField.ANY);
        this.append(message);
        this.addCommand(sendCommand);
        this.addCommand(backCommand);
        this.setCommandListener(this);
    }

    public void commandAction(Command cmd,Displayable displayable){
        if(cmd == backCommand){
            // Code here
        }
    }
}
midlet.backToMain();
} else if(cmd == sendCommand){
    content1 = midlet.getlocationLongitude()+"";
    content2 = midlet.getlocationLatitude()+"";
    content3 = message.getString();
    content4 = 2;
    new Thread(this).start();
}
}

public void run(){
    String address = number+:"="+midlet.getAppProperty("port");
    MessageConnection mconn = null;
    try{
        mconn = (MessageConnection)Connector.open(address);
        TextMessage tm =
            (TextMessage)mconn.newMessage(MessageConnection.TEXT_MESSAGE);
        tm.setAddress(address);
        tm.setPayloadText(content1+"@"+content2 +"@"+content3+"@"+content4);
        mconn.send(tm);
        midlet.backToMain();
    }catch(Exception ex){
        ex.printStackTrace();
    }
}

MarkCurrentPlaceUI.java

package Gps.UI;

import Gps.MapLoader;
import java.lang.String;
import javax.microedition.lcdui.Choice;
import javax.microedition.lcdui.ChoiceGroup;
import javax.microedition.lcdui.Command;
import javax.microedition.lcdui.CommandListener;
import javax.microedition.lcdui.Displayable;
import javax.microedition.lcdui.Form;
import javax.microedition.lcdui.Item;
import javax.microedition.lcdui.TextField;

/**
 * @author Sha Mao Xuan
 */

public class MarkCurrentPlaceUI extends Form implements CommandListener,ItemStateListener{
    private MapLoader midlet;
    private Command backCommand = new Command("Back",Command.BACK,0);
private Command MarkPlace = new Command("MarkPlace", Command.SCREEN, 1);
private String PlaceLocationInfo = new String("Please Enter the Place Location");
private ChoiceGroup SizeChoice = new ChoiceGroup("Choose Size", Choice.POPUP);
private ChoiceGroup ColorChoice = new ChoiceGroup("Choose Color", Choice.POPUP);
private TextField markN = null;

public MarkCurrentPlaceUI(MapLoader midlet) {
    super("Mark Current Place Form");
    this.midlet = midlet;
    markN = new TextField("Input mark letter(lowercase)(a~z,0~9)", ",", 1, TextField.ANY);
    this.append(PlaceLocationInfo);
    this.append(SizeChoice);
    SizeChoice.append("mid", null);
    SizeChoice.append("tiny", null);
    SizeChoice.append("small", null);
    this.append(ColorChoice);
    ColorChoice.append("black", null);
    ColorChoice.append("brown", null);
    ColorChoice.append("green", null);
    ColorChoice.append("purple", null);
    ColorChoice.append("yellow", null);
    ColorChoice.append("blue", null);
    ColorChoice.append("gray", null);
    ColorChoice.append("orange", null);
    ColorChoice.append("red", null);
    ColorChoice.append("white", null);
    this.append(markN);
    this.addCommand(backCommand);
    this.addCommand(MarkPlace);
    this.setCommandListener(this);
}

public void commandAction(Command cmd, Displayable displayable) {
    if(cmd == backCommand) {
        midlet.backToMain();
    } else if(cmd == MarkPlace) {
        int i;
        for(i=0;midlet.locationLatitude[i]!=0||midlet.locationLongitude[i]!=0;i++){
            midlet.locationLongitude[i] = midlet.mapCenterLongitude;
            midlet.locationLatitude[i] = midlet.mapCenterLatitude;
            midlet.markSize[i] = SizeChoice.getString(SizeChoice.getSelectedIndex());
            midlet.markColor[i] = ColorChoice.getString(ColorChoice.getSelectedIndex());
            midlet.markName[i] = markN.getString();
            midlet.backToMain();
            midlet.run();
        }
    }
}

public void itemStateChanged(Item item) {
}
import Gps.MapLoader;
import javax.microedition.lcdui.Command;
import javax.microedition.lcdui.Item;
import javax.microedition.lcdui.ItemStateListener;
public class MapTypeUI extends Form implements CommandListener,ItemStateListener{

    private MapLoader midlet;
    private Command backCommand = new Command("Back",Command.BACK,0);
    private Command ChangeMapType = new Command("Choose",Command.SCREEN,1);
    private String MapTypeInfo = new String("Please choose the maptype");
    private ChoiceGroup MapTypeChoice = new ChoiceGroup("Choose maptype",Choice.EXCLUSIVE);

    public MapTypeUI(MapLoader midlet) {
        super("Map Type Form");
        this.midlet = midlet;
        this.append(MapTypeInfo);
        this.append(MapTypeChoice);
        MapTypeChoice.append("mobile", null);
        MapTypeChoice.append("roadmap", null);
        MapTypeChoice.append("satellite", null);
        MapTypeChoice.append("hybrid", null);
        MapTypeChoice.append("terrain", null);
        MapTypeChoice.append("mapmaker-roadmap", null);
        MapTypeChoice.append("mapmaker-hybrid", null);
        this.addCommand(backCommand);
        this.addCommand(ChangeMapType);
        this.setCommandListener(this);
    }

    public void commandAction(Command cmd,Displayable displayable){
        if(cmd == backCommand){
            midlet.backToMain();
        }
        if(cmd == ChangeMapType){
            midlet.maptype = MapTypeChoice.getString(MapTypeChoice.getSelectedIndex());
            midlet.backToMain();
            midlet.run();
        }
    }
}
Appendix – AboutUI.java

```java
public void itemStateChanged(Item item) {
}
}

AboutUI.java

package Gps.UI;

import Gps.MapLoader;
import javax.microedition.lcdui.Command;
import javax.microedition.lcdui.CommandListener;
import javax.microedition.lcdui.Displayable;
import javax.microedition.lcdui.Form;

/**
 * @author Sha Mao Xuan & Xie Jun & Xu Xiao Lin
 */
public class AboutUI extends Form implements CommandListener{
    private MapLoader midlet;
    private Command backCommand = new Command("Back", Command.BACK, 0);

    public AboutUI(MapLoader midlet) {
        super("About");
        this.midlet = midlet;
        String content = "This Software is programming by
     Sha Maoxuan
     Xie Jun
     Xu XiaoLin

Superviser: 
    Michal
    Tobias
    Nicholna"

        this.append(content + "\n");
        this.addCommand(backCommand);
        this.setCommandListener(this);
    }

    public void commandAction(Command cmd, Displayable displayable){
        if(cmd == backCommand){
            midlet.backToMain();
        }
    }
}
```
Appendix – Attributes and Source Package Structure

Attributes

<table>
<thead>
<tr>
<th>Type</th>
<th>Name</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Required</td>
<td>MIDlet-Name</td>
<td>FindYourFriend_Final</td>
</tr>
<tr>
<td>Required</td>
<td>MIDlet-Vendor</td>
<td>Vendor</td>
</tr>
<tr>
<td>Required</td>
<td>MIDlet-Version</td>
<td>1.0</td>
</tr>
<tr>
<td>Custom</td>
<td>Google_Geo_URL</td>
<td><a href="http://maps.google.com/maps">http://maps.google.com/maps</a></td>
</tr>
<tr>
<td>Custom</td>
<td>Google_MapsKey</td>
<td>ABC4ADD7BEbcv8NJt213DkYdvc2JFTkO2_J5_6437z51R388_D2F9V/JzYmKg/202956AIbW</td>
</tr>
<tr>
<td>Custom</td>
<td>Google_Static_Maps_URL</td>
<td><a href="http://maps.google.com/stationmap">http://maps.google.com/stationmap</a></td>
</tr>
<tr>
<td>Custom</td>
<td>port</td>
<td>5000</td>
</tr>
</tbody>
</table>

Figure 1: All the attributes we need to define by ourselves in the project.

Source Package Structure

Figure 2: You can flow this package structure to create your own Find Your Friend Midlet.