KEMI-TORNIO UNIVERSITY OF APPLIED SCIENCES

Online Map Application

Tracking Vehicles and Personnel

Shrestha Padam

Bachelor’s Thesis of Degree Programme in Information Technology

KEMI 2011

PREFACE

This Bachelor’s Thesis Project, Online Map Application for Tracking Vehicles and Personnel was done in IMSS Ltd. Oy Ab.

I would like to thank Antti-Pekka Aaltonen, CTO (Chief Technical Officer) of IMSS Ltd. Oy Ab for providing me the thesis topic with interesting challenges and giving excellent opportunity to develop my skills and abilities in the information technology industry which offered professional growth.

My special thanks go to Sami Kleemola, Lead Programmer of IMSS Ltd. Oy Ab for assisting me throughout the project and helping to develop my skills professionally as a programmer.

I am also thankful to the staff of Kemi-Tornio University of Applied Sciences for providing me with supportive learning environment and especially Thai Bui for being my thesis instructor.

Kemi, June 2011

ABSTRACT

Kemi-Tornio University of Applied Sciences, Technology

Degree Programme	Information Technology
Name	Shrestha Padam
Title	Online Map Application for Tracking Vehicles
Type of Study	Bachelor's Thesis
Date	15 May 2011
Pages	49 + 2 appendices
Instructor	Thai Bui
Company	IMSS Ltd. Oy Ab
Contact Person/Supervisor from Company	Antti-Pekka Aaltonen, CTO, IMSS Ltd. Oy Ab

The objective of this Bachelor's thesis was to analyze, design and implement an online map application so that the customer can track vehicles and personnel in working field via web browsers, which is the product of IMSS Ltd. Oy Ab.

The research methods were mainly programming and Java programming language was used for developing the application. A research was conducted on Java Applet looking for its requirements, advantages and disadvantages.

The whole project was categorized in three different parts: mobile application, server application and map application (for desktop). The mobile application and server application were developed by other team members in the company. The purpose of this Bachelor's thesis was to develop map application for desktop computers though it was originally planned to be developed for smart phones and desktop computers.

Keywords: Java Applet, Map application, software development, tracking vehicles.

CONTENTS

EXPLANATION OF CHARACTERS AND ABBREVIATIONS

API	Application Programming Interface
AWT	Abstract Window Toolkit
CTO	Chief Technical Officer
DBMS	Database Management System
DOM	Document Object Model
GPS	Global Positioning System
GSM	Global System for Mobile Communications
GUI	Graphical User Interface
HTML	Hypertext Mark-up Language
IDE	Integrated Development Environment
I/O	Input/output
JAXP	Java API for XML Processing
JDK	Java Development Kit
JFC	Java Foundation Classes
JVM	Java Virtual Machine
J2SE	Java 2 Standard Edition
J2EE	Java 2 Enterprise Edition
J2ME	Java 2 Micro Edition
KTUAS	Kemi - Tornio University of Applied Sciences
OS	Operating System
PHP	Hypertext Preprocessor
SAX	Simple API for XML
SWT	Standard Widget Toolkit
UTF-8	UCS Transformation Format - 8 bit
UCS	Universal Character Set
URI	Uniform Resource Identifier
URL	Uniform Resource Locator
URN	Uniform Resource Name
W3C	World Wide Web Consortium

XML	eXtensible Markup Language
2D	Two Dimensional
3D	Three Dimensional

1. INTRODUCTION

Getting a topic for Bachelor’s thesis from an established company in Finland is not an easy job, especially for the international student who does not have good Finnish language skills. After contacting several local companies in Finland, finally IMSS Ltd. Oy Ab agreed to provide thesis work for me.

Today everyone wants to use automated technologies in their businesses for various purposes. A taxi company in Oulu, Finland needed an automated web-based map application to track its vehicles around the Oulu region and IMSS Ltd. Oy Ab got the contract to work on that project. This thesis is a result of part of that project.

1.1. IMSS Ltd. Oy Ab

IMSS Ltd. Oy Ab was established in 2008. It had its origin already from 2004 but from 2008 it was called IMSS. IMSS started to form in 2006, when Antti-Pekka Aaltonen and Panu Koivurova decided to work together to improve industrial maintenance by making innovative software. In 2007, a project was started, where Outokumpu, Kemi-Tornio University of Applied Sciences (Unit of Technical Education) and the founders of IMSS worked together to make a product to meet Outokumpu’s need. The project ended in 2008, and after that IMSS was founded to make business use of the project’s results. In the beginning there were three people working in the company. All of them were owners of the company. The main customer was Outokumpu Stainless Tornio.

IMSS provides software for various business uses and the main focus is on work management system and mobile software. Their main customers are industry, taxi, security and maintenance companies. Currently, IMSS has nine people working for the company in the spring of 2011.

IMSS has several software and hardware partners and resellers. In the near future, IMSS will concentrate on constantly improving its current services and software.

1.2. Project Introduction

This project is about building a web based map application for tracking vehicles and personnel on the working field. For this particular project, application is done to track vehicles of the taxi company in Oulu region.

The application is based on Java Applet technology, which makes the application more flexible and portable. Although the application is presently intended for desktop computers, these flexibility and portability will help further to develop the application and run on smart phones and other mobile devices. Due to Java Technology it has been possible to have high controls over graphics too.

The first release of this application will be able to track about fifty vehicles of the taxi company in Oulu region in real-time. Each vehicle is installed with GPS tracking mobile devices which sends its GPS data to server and can be tracked easily with map application via Java enabled internet browsers. Java Applet is embedded in web pages, much in the same way a flash video is embedded in a page. This thesis’s project is based on the development of map application. Therefore, other parts of the project such as mobile and server applications are not discussed here. Map application is developed independently. There is no direct communication between mobile application and map application. Map application uses data available in database which is retrieved using PHP and creates HTML code for embedding applet is created.

Fig. 1 describes general communication between vehicle, server and desktop applications. Each GPS receiver installed in vehicles receives its GPS location from GPS satellite and transmits it to server using GSM network. Further, server sends the received GPS data to the desktop application. The desktop application processes the GPS data received and locates it into the map.

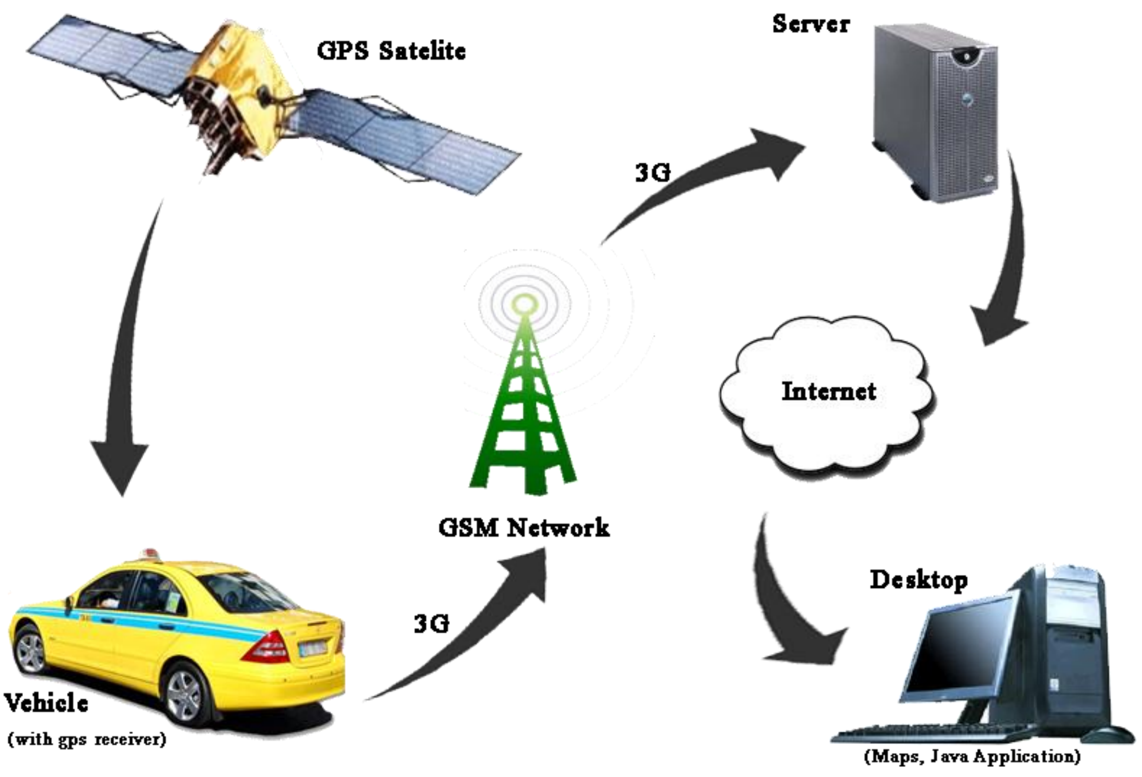


Fig. 1. Overview Description of the Project

1.3. Project Scope and Objectives

1.3.1. Project Objectives

The main objective of this project is to develop a cost effective web based application to track vehicles of the taxi company through receiving GPS data from the vehicles. Despite the fact that the application could be developed by using Google Map API/Bing Map API/ Open Street Map API (open source), the application is developed without using any third party APIs. The paid Map APIs such as Google and Bing are quite expensive and hence not appropriate for this project. Since the source code of the application contains confidentiality, it cannot be made available for public in the future. The use of Open Source Map API such as Open Street also does not seem to be a right choice.

The application is made as light weight as possible with particular focus on the customer needs. The main objective of the project is to show the real-time location of vehicles into the map.

1.3.2. Project Scope

The vehicle can be tracked by the customer using this application. Users can see the real-time location of single or multiple vehicles at the same time. The GPS locations from the vehicles are updated in the certain time interval. Since the application is done by Java programming language, it is portable, light weight, can be further developed easily for smart phones using the same source code. Since the application is not using any third party APIs, it is more controlled by hard code. It is also cheap in cost in long run business. In future, the application can be developed to cover even larger territory apart from the Oulu region.

Though the project seems quite large with multiple applications such as server side application, mobile application and map application for web browsers, this project is more concerned only to map application. Map application is done by Java Applet Technology so that it can be embedded into web pages and can easily access via Java enabled web-browsers. Though Java has already built in class library for Graphical Interface, most of the graphical interface in this application is done by using custom images, so that it can be controlled and source code can be further modified easily to develop same application for smart phones or other mobile devices. All the map images in the application are provided by other company partners of IMSS Ltd. Oy Ab. A single static map image is split into several pieces and drawn into applet on demand. GPS location of vehicles in the map are drawn either by querying XML file from the server or by using values of parameter from HTML document where applet is embedded. Since the application is targeted for only to desktop computers initially, mobile handsets properties are not inspected. Some basic navigation controls from mouse and keyboard are also implemented into the application.

2. BACKGROUND

2.1. Java Platforms

Java is becoming popular among programmers who need to develop application to run on a wide variety of different computer systems. The most important feature of Java is that it is object oriented. Java does not execute directly on the computer. It runs on a standardized hypothetical computer that is called the Java Virtual Machine (JVM), which is emulated inside your computer by a program. Due to this, Java has become possible to run into different computer system which has JVM. To develop a Java application, Java Development Kit is necessary which contains Java compiler. /6, page 8-10/

Basically Java is found into three different platforms adapting different circumstances. Java is formally known as Java 2 platform.

- **J2SE** (Java 2 Standard Edition): This edition is used for developing Desktop-based applications, which provides necessary user interface classes.
- **J2EE** (Java 2 Enterprise Edition): J2EE is generally used for developing Server-based applications, which emphasizes component-based programming and deployment.
- **J2ME** (Java 2 Micro Edition): This edition is targeted for handheld and embedded devices. For example, mobile phones and setup boxes.

Each edition of the platform provides a complete environment for running Java-based applications, including the JVM and runtime classes./17/

2.1.1. Java Applet

Java enables to write small programs called applets. These are the programs that can be embedded in Internet Web Pages to provide some intelligence. Java Applets run within Internet Web Browser using JVM. Since Java is cross-platform, it can be executed by many platforms, including Microsoft Windows, UNIX, Mac OS and Linux. Java Applet can execute on either within Java enabled web browsers or by using the applet viewer. Java Applets can capture mouse and keyboard inputs. /18/

Embedding an Applet to an HTML Document

The applet can be embedded into web pages by using `<applet>` tag in HTML document. Tags like `<object>` and `<embed>` can also be used to embed applet into web pages. Source Code List 1 shows an example of how to include an applet into HTML document using `<applet>` tag.

```
<html>
<head>
<title>
Hello World Java Applet
</title>
</head>
<body>
<applet code=helloWorld.class  width=300  height=200>
</applet>
</body>
</html>
```

Source Code List 1. Simple way of embeding applet into html document /6, page 15/

The name of the file containing the byte codes for the applet is specified as the value of code attribute in the `<applet>` tag. The other two attributes width and height specifies the width and height of the region on the screen for applet to display. The values of height and width should be large enough to display all the applet contents. /6, page 15/

Cross browser applet

The `<applet>` tag was initially introduced in HTML 3.2 which got deprecated in HTML 4.0. Further, the `<applet>` tag was criticized and was suggested to use the tag `<object>` instead. Oracle now provides a maintained JavaScript code to launch applets with cross platforms workarounds. However `<object>` tag is also introduced to embed applet into

HTML document. Few examples on embedding applet into HTML document in different ways are given below, equivalent to previous HTML code.

```
<object width=300 height=200>
<param name=code value=helloWorld.class>
<param name=type value=”application/x-java-applet;version=1.3”>
</object>
```

Source Code List 2. Different way of embeding applet into HTML document equivalent to Source Code List 1. /6/

The code above will not work on Internet Explorer unless it has a virtual machine that can find a JApplet class. /6, page 17/

The HTML code given below shows how Java Applet can be embedded using `<embed>` tag rather than using the tag `<applet>`. The code is equivalent to the previous HTML code despite the fact that `<embed>` tag has been used instead of `<applet>`.

```
<embed type=”application/x-java-applet;version=1.3”
code=helloWorld.class
width=300
height=200
pluginspace=http://java.sun.com/products/plugin/1.3/plugin-
install.html>
</embed>
```

Source Code List 3. Another way of embedding applet into HTML document

2.2. Java Development Tools

2.2.1. Java Development Kit (JDK)

The JDK is a development environment for developing Java Applications. It is necessary for developing any kind of application with Java. There are numbers of excellent and friendly Java Program Development Environment developed by Sun, Borland and

Symantec. JDK can be downloaded for variety of hardware platforms and operating system for free of charge. /6, page 10/. The JDK includes tools useful for developing and testing Java programs and running them on the Java platform. The JDK contains development tools such compliers and debuggers. This JDK is a must for developing a program, software using Java programming language. Programs or software developed using other programming languages do not require this JDK. /16/

2.2.2. NetBeans IDE

Source code of any programming language can be written in plain text editor or in IDE (Integrated Development Environment). The IDE makes programmer to write code easily and faster. IDE is not just an editor for writing code and debugging, it has many features such as: error checking, code navigation, code completion, code generation, code coloring, refactoring, version control integration, smart typing, test runner, debugging, etc.

The NetBeans is free, open source Integrated Development Environment for creating applications with Java platform as well as with C/C++, PHP, JavaScript and Groovy. This IDE has been developed with Java and can run in different operating systems such as Windows, Linux and Macintosh as long as Java Virtual Machine (JVM) is installed in the system. /14/

Features of NetBeans IDE

– **Multi-Language Editor**: The NetBeans IDE supports several languages such as: Java, C/C++, XML, HTML, PHP, Groovy, Javadoc, JavaScript, and JSP.
– **Live Parsing**: The IDE parses the source code live while typing in the editor and immediately marks errors and highlights occurrences.
– **Refactoring:** This feature allows restructuring code without break.
– **Code Completion:** IDE lists all possible completions while typing and most obvious and common are listed at the top.
– **Insert Code:** For Java and some other programming languages, IDE can generate common code snippets such as getters and setters, constructors, try-catch-blocks, loops etc.

- **Version Control Integration:** NetBeans uses mercurial for controlling different versions of projects. Additional application might be needed to install to use this feature. /15/

2.3. Global Positioning System (GPS)

Global Positioning System (GPS) is a satellite-based navigation system made up of a network of 24 satellites placed into orbit by the U.S. Department of Defense. The GPS satellites revolve around the earth twice a day in a very precise orbit and transmit signal information to earth. It is freely accessible for anyone with a GPS receiver. GPS works in any weather conditions, anywhere in the world, 24 hours a day. A GPS receiver on the earth or nearby earth receives the signals from the GPS satellites and uses the transmitted information to calculate the user’s three-dimensional position (i.e. longitude, latitude and altitude) and time. The three-dimensional position is used to navigate in map. These days we can find GPS enabled mobile phones and some with navigation support too. /4/

GPS satellites do not have comprehensive map databases such as Google Earth, Bing. GPS satellites do not know the location of GPS receiver. Navigation devices (GPS receiver) use the satellites as reference points to determine their coordinates on the earth in terms of latitude, longitude, and altitude. Navigation devices have built-in digital map which navigate users using the coordinates. /5/

Fig. 2. GPS Satellites revolving around the earth. /4/

2.4. Extensible Markup Language (XML)

XML is a textual data format containing structured information and was designed to carry and store data rather than displaying data. It is a markup language much like HTML but HTML was designed to display data. /25/ XML is playing an increasingly important role in the exchange of a wide variety of data on the Web. XML became a W3C recommendation from February 10, 1998. XML is not a replacement of HTML but a complement to HTML. In most of web application XML is used to either transport or store data. Some applications use XML for storing data rather than using Database Management System (DBMS). The XML specification defines a standard way to add markup to documents. Unlike HTML, XML tag semantics and the tag set are not fixed. The tag `<books>` is meaningless in HTML but can have meaning in XML, user can define their own tags and use in XML. XML specifies neither semantics nor a tag set. In fact XML is really a meta-language for describing markup languages. Since there is no predefined tag set, there cannot be any preconceived semantics. All of the semantics of a XML document will either be defined by the application that processes them or style sheets. XML document should be human-legible and reasonably clear and designed quickly. /26/

In this particular project, XML document is used to store records of GPS locations received from mobile and map application uses its information to display markers in map. It contains information such as vehicle's name, latitude, longitude, marker color, etc. Below is an example of XML document used in this project.

```
<?xml version="1.0" encoding="UTF-8"?>
<locations>
        <location>
                <latitude>65.025475</latitude>
                <longitude>25.480206</longitude>
                <color>ff0000</color>
                <name>Test one</name>
        </location>
        <location>
                <latitude>65.02631</latitude>
                <longitude>25.473938</longitude>
```

```
                        <color>ff0000</color>
                        <name>Oulu two</name>
            </location>
            <location>
                        <latitude>65.0588</latitude>
                        <longitude>25.4835</longitude>
                        <color>000000</color>
                        <name>Bin23</name>
            </location>
</locations>
```

Source Code List 4. XML document used in the map application

The above XML document has the records of three locations. The first line is the XML declaration. It defines the XML version (1.0) and the encoding used (UTF-8). With the tree structure of XML, it is easier to describe the structure of data. In the example above `<locations>` is the root element of the XML document which holds the `<location>` as its child element. The child element `<location>` further has four sub elements: `<latitude>`, `<longitude>`, `<color>` and `<name>`. The element `<latitude>` and `<longitude>` contain GPS coordinate values, latitude and longitude in decimal format respectively. The `<color>` element contains the hexadecimal color value for the marker and the `<name>` element contains the name for the marker, this name can be the name of vehicle’s driver or any unique name to identify the vehicle. /26/

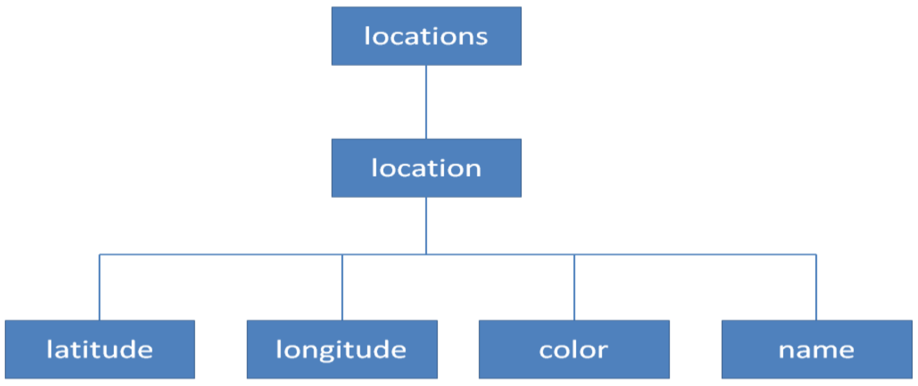


Fig. 3. Tree structure of XML document used for storing vehicles’ information.

The above Fig. 3 explains the tree structure of XML document used in this project to store GPS coordinates and other information of the vehicles. From the above figure we see `locations` element on the top of the tree and all other elements are descendent of `locations` element.

3. IMPLEMENTATION

3.1. System Requirement

System requirements for developing a Java Application and running a Java Application vary in both hardware and software. A Java Application can run on any computer having 128 mb memory as mentioned in Java Official website (http://www.java.com/). JVM (Java Virtual Machine) should be installed and Java enabled web-browser should exist in the system to run Java Applet. For developing a Java Application, Java complier and JDK (Java Development Kit) should be installed in the system. IDE like NetBeans is also necessary to develop a Java Application more efficiently and conveniently. Here is a system requirement for installing NetBeans IDE provided in NetBeans website. /6, page 12/

Table 1. System requirement for NetBeans IDE /13/

Minimum hardware configurations	**Recommended hardware configurations**
Microsoft Windows XP Professional SP3/Vista/Windows 7 Professional: o Processor : 800MHz Intel Pentium III or equivalent o Memory: 512 MB o Disk space: 750 MB of free space	Microsoft Windows XP Professional SP3/Vista/Windows 7 Professional: o Processor : 2.7GHz Intel Pentium VI or equivalent o Memory: 2 GB o Disk space: 1 GB of free space
Ubuntu 9.10 o Processor : 800MHz Intel Pentium III or equivalent o Memory: 512 MB o Disk space: 650 MB of free space	Ubuntu 9.10 o Processor : 2.6 GHz Intel Pentium IV or equivalent o Memory: 2 GB o Disk space: 850 MB of free space
Solaris OS version 10 (SPARC)	Solaris OS version 10 (SPARC)

○ Processor : UltraSPARC II 450 MHz ○ Memory: 512 MB ○ Disk space: 650 MB of free space	○ Processor : UltraSPARC IIIi 1 GHz ○ Memory: 2 GB ○ Disk space: 850 MB of free space
Solaris OS version 10 (x86/x64 platform edition) ○ Processor : AMD Opteron 1200 Series 1.8 GHz ○ Memory: 512 MB ○ Disk space: 650 MB of free space	Solaris OS version 10 (x86/x64 platform edition) ○ Processor : AMD Opteron 1200 Series 2.8 GHz ○ Memory: 2 GB ○ Disk space: 650 MB of free space
Macintosh OS x 10.5 Intel: ○ Processor : Dual-Core Intel (32 or 64-bit) ○ Memory: 512 MB ○ Disk space: 650 MB of free space	Macintosh OS x 10.5 Intel: ○ Processor : Dual-Core Intel (32 or 64-bit) ○ Memory: 2 GB ○ Disk space: 850 MB of free space
Required Software: JDK 6 Update 13 or later is required. The 6.9.1 version of the IDE cannot be installed or run using JDK 5.0.	

The above table shows the system requirements for installing NetBeans IDE in different platforms: Microsoft Windows, Ubuntu, Solaris, and Macintosh.

3.2. System/Application Architecture

3.2.1. Use Case Diagram

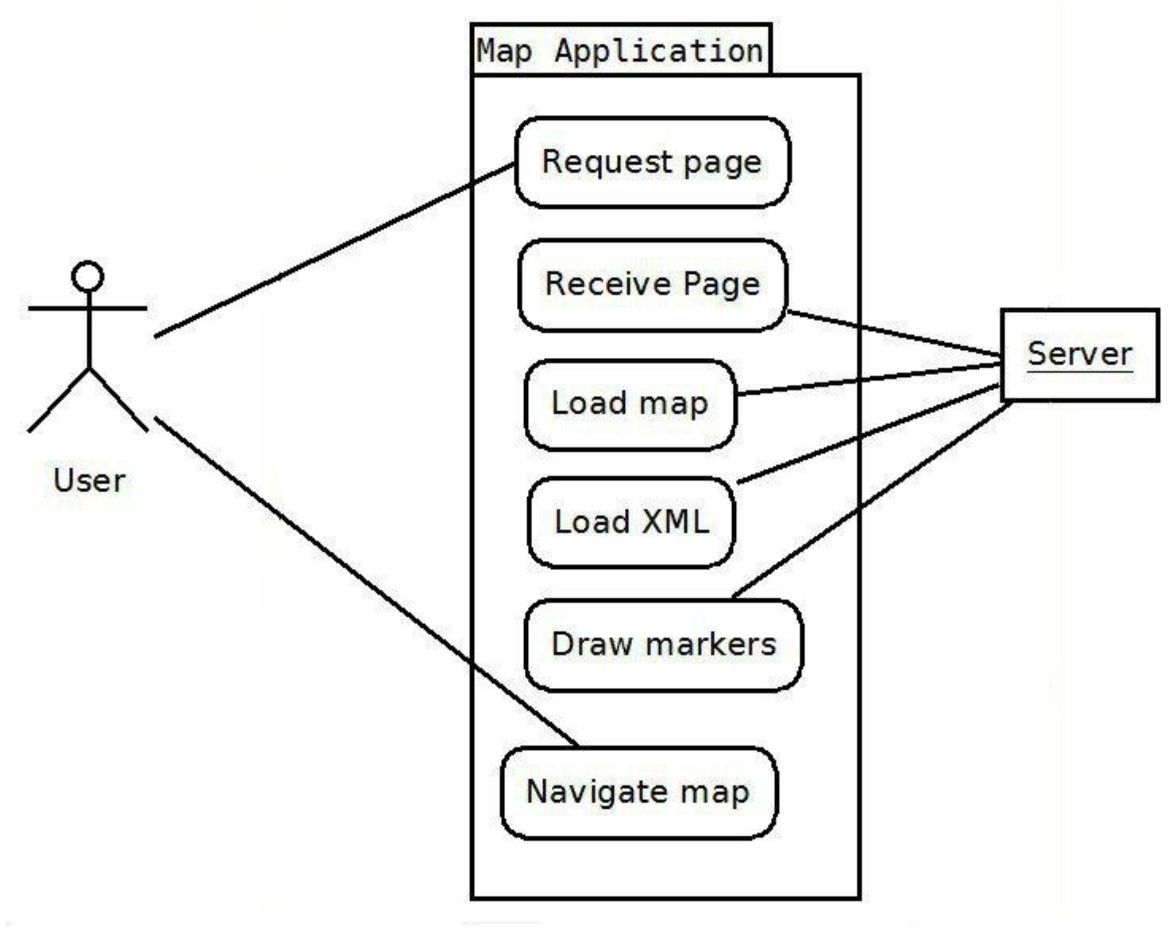


Fig. 4. Use Case Diagram of the application

Fig. 4 is a use case diagram of the map application. There are two actors, one is the user of the application and the other is server. The user requests the applet via browser and the browser loads the map application from the server. XML file is loaded from the server and markers are drawn using the data from the XML document.

3.2.2. Class Diagram

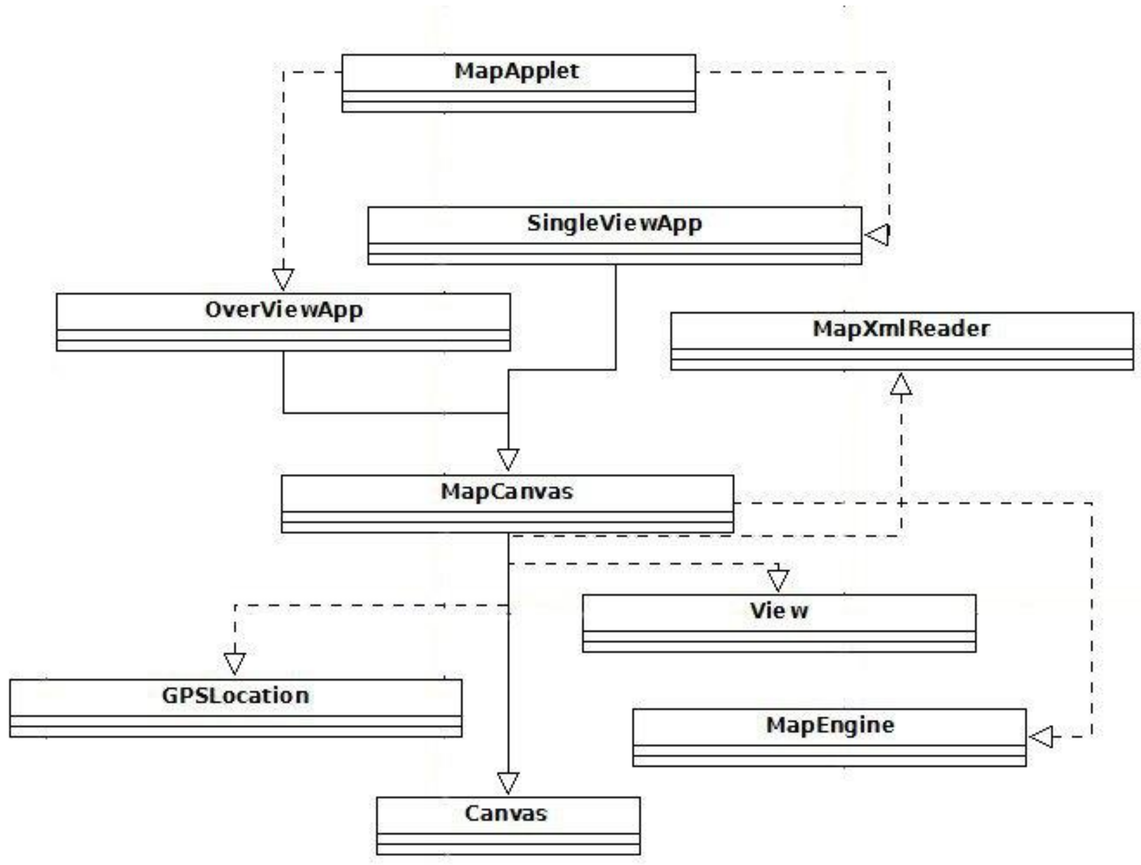


Fig. 5. Class Diagram of the application

Fig. 5 describes the class diagram of the application. `MapAplet` class uses either `SingleViewApp` or `OverViewApp` to load applet into the browser according the application type requested. `SingleViewApp` and `OverViewApp` are the extended classes of `MapCanvas`, `MapCanvas` class is the extended class of `Canvas`. `SingleViewApp` class is responsible drawing marker for single location and `OverViewApp` class is responsible for drawing markers for all the available locations. In the `SingleViewApp` class the view is centered to the marker and in the `OverViewApp` view is centered to the map. `MapCanvas` class is the main class in the map application. GPS markers, map images and GUI buttons are drawn in this `MapCanvas` class. All the mouse and keyboard events are also implemented in the `MapCanvas` class. `Canvas` class is an extended class of panel. Java has its own `Canvas` class but a different `Canvas` class was created in this map application for particular requirements. The `Canvas` class in this map application is responsible for creating multiple graphical views if needed. `MapXmlReader` class is responsible for parsing the XML document and `View` class has methods to manage the view of the applet. `MapEngine` class consists of all the methods of necessary calculations for applet such as: calculating GPS coordinates, calculating markers' locations into the applet, etc. `GPSLocation` class contains the information about the markers such as: longitude, latitude, name and color. The detailed about attributes and methods of classes are show in APPENDIX 1.

3.3. Graphical User Interface (GUI) Design

3.3.1. GUI design in Java

Java comes with different GUI toolkits for different purposes. Swing toolkit is a part of Java Foundation Classes (JFC) which provides GUI programming in Java. GUI programming with Swing toolkit in NetBeans IDE offers many advantages over coding with a text editor. Swing toolkit provides a rich set of GUI components and offers much more functionalities than a collection of standard widgets. Swing toolkit provides visual guide to Swing Components. Most of the GUI components are designed by drag and drop method. The Swing toolkit includes a rich array of components: from basic components, such as buttons and check boxes, to rich and complex components such as tables and text. Even deceptively simple components, such as text fields, offer sophisticated functionality, such as formatted text input or password field behavior. There are file browsers and dialogs to suit most needs, and if not, customization is possible. It allows deciding how to configure the particular look and feel the application. /19/

AWT is another GUI toolkit in Java for creating GUI. It is very simple toolkit with limited GUI components, layout managers, and even events. Only GUI components defined for all Java host environments would be used in AWT. /3/

Java also has another GUI toolkit: SWT toolkit. SWT is a low-level GUI toolkit comparable in concept of AWT. Like AWT and Swing Layout managers, SWT provides a comprehensive set of layouts. Unlike AWT and Swing, SWT and JFace do not come with Java technology. They must be installed separately as part of an Eclipse installation or as separate libraries. /3/

3.3.2. GUI in Map Application

Map Application of this project needs some basic controls for navigating map using mouse and keyboard. Though Java comes with built in packages for Graphical User Interface programming, custom images are used in this project to create graphical controls. Firstly, map images are drawn into the applet, secondly, GPS markers of vehicles are drawn and finally, graphical controls are drawn so that these graphical controls always remain on the top level. There are six static images for drawing controls for the map: step up, step down, step right, step left, zoom in and zoom out.

Firstly, images are added to `Media-tracker` and drawn into the applet. The `MediaTracker` class is a utility class to track the status of a number of media objects /9/. The `MediaTracker` will help to keep a track on loading lots of images. The image positions are defined by calculating the ratio of width and height of the applet. These image positions (x, y) retain always the same ratio according to the width and height of the applet.

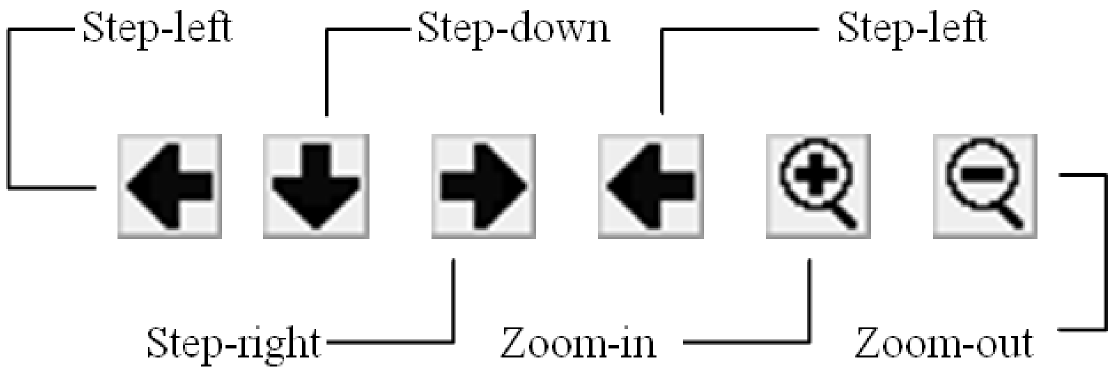


Fig. 6. Images used for graphical controls in the map application

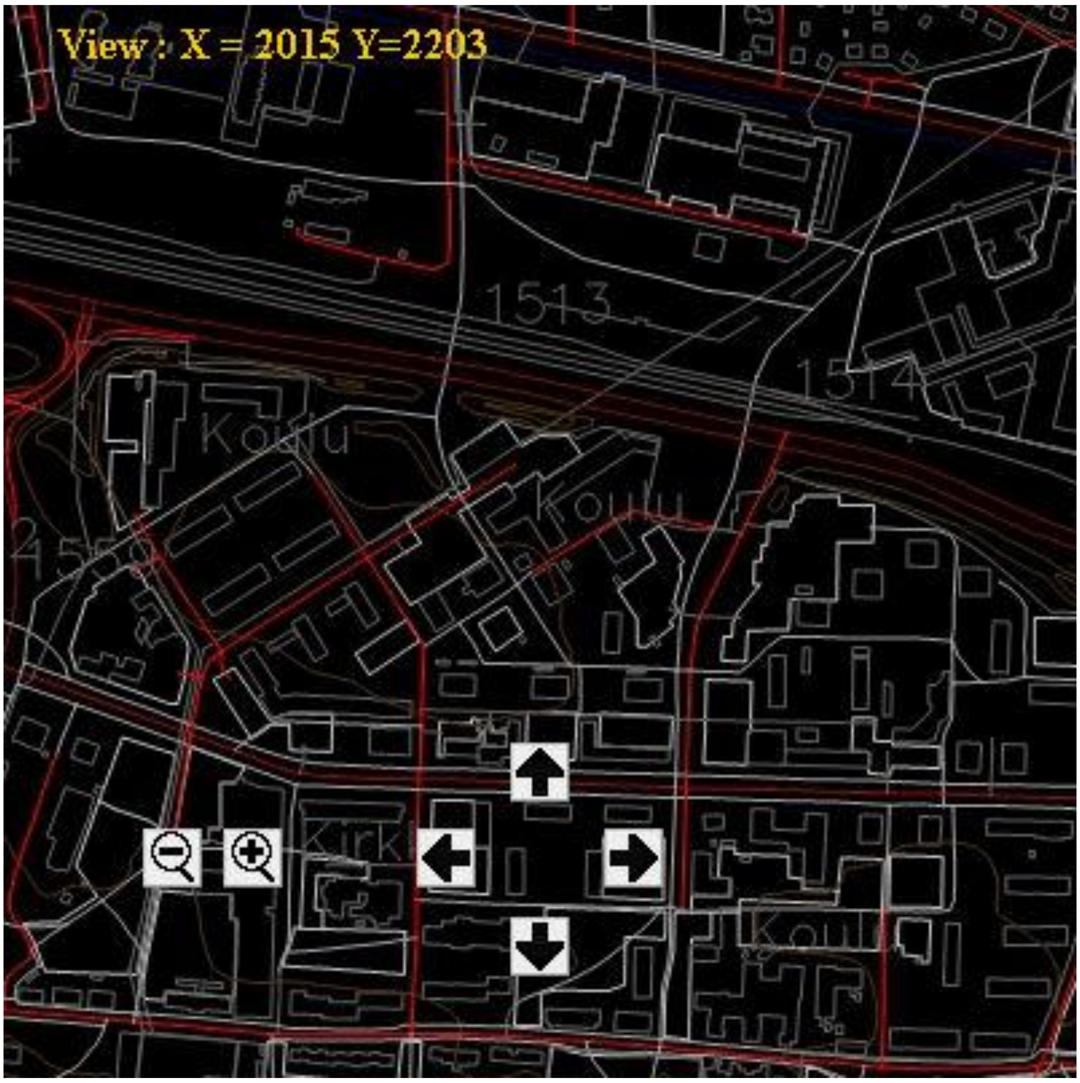


Fig. 7. Graphical controls in the map application

3.4. Mapping Mechanism

The map used in this project is a large static image in JPEG format. Images in an applet are drawn from server, not from local computer. Hence, images should be light enough to load in the applet so that the program would respond quicker. Loading large images in the applet would not please the customer and it is also not recommended due to memory loss. Therefore, it was required to split the single large image into several small images. Small images would help to load image only on demand. The initial map image provided was of 20MB with dimension of 4431 x 4806 pixels which had to split into several pieces with dimension of 400 x 400 pixels. According to this split dimension, there are about 12 horizontal pieces and 13 vertical pieces which make 156 pieces altogether.

Splitting a single large image into several small pieces would require a high attention. For this purpose, different desktop applications were used but they were not very effective. Naming the split images has to be done carefully to distinguish images position. For example, if split dimension would be lower, there would be more small images and locating images into the map application would get complex. Therefore, naming small images such as 0_0.jpg would make easier to locate images in right place. The first digit "0" is X position and second digit "0" is the Y position. Images are loaded into the applet on demand; only required images for particular view are loaded. Therefore, images should be easier to distinguish.

Different desktop applications such as Photoshop, Gimp were used initially to split images but there were problems in naming the images. These applications also required huge amount of time, for example splitting image into 1000 pieces would require significant amount of time. Using scripting application such as "imagick" would help to split images but using this application was complex. Finally, own PHP script was used to split image. Few lines of code made it possible to split the large map image into several small pieces. The small pieces of image created by PHP script were between 13KB to 150 KB and naming those pieces was easier. The PHP source code is given below.

```
<?php
$width = 400; //width of split image
$height = 400; //height of split image
$source = @imagecreatefromjpeg("map_raw_image.jpg"); //Source of image
$source_width = imagesx($source);
$source_height = imagesy($source);
$save_dir = "mapImages/"; //directory to save splited images
for($col=0; $col<($source_width/$width); $col++){
           for( $row=0; $row<($source_height/$height); $row++){
                      $fn = sprintf("%d_%d.jpg", $col, $row);
                      $im = @imagecreatetruecolor($width, $height);
                      imagecopyresized($im, $source, 0, 0, $col*$width,
                      $row*$height, $width, $height,   $width, $height);
```

```
                    imagejpeg($im, $save_dir.$fn, 100);
                    imagedestroy($im);
          }
}
?>
```

Source Code List 5. Source code for splitting single image into multiple pieces.

In the above PHP code `$source` is the variable of image source which needs to be split into several pieces. First, the source image is split into row and then into column. Variable `$fn` is the name of the split image.

After splitting the large image into several pieces of image, those pieces were plotted into applet according to their names which represent the location of the image in the map. The picture below shows the plotting principle of small images which finally forms the original and full image.

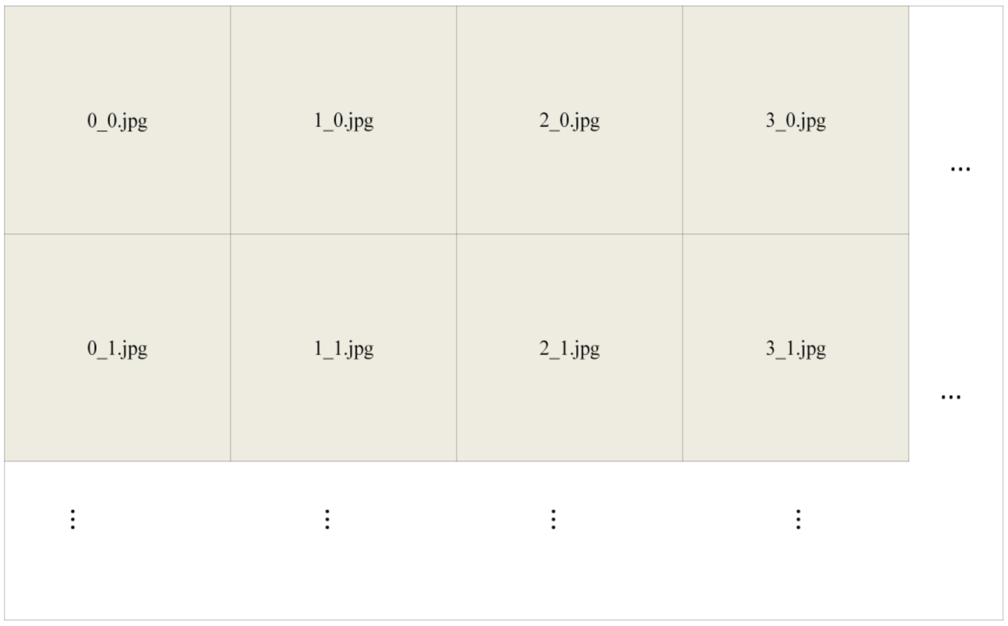


Fig. 8. Small pieces of image plotted to form original image

Fig. 8 shows the plotting principle of images into the applet for creating the original like image.

3.5. Navigation Controls

3.5.1. Mouse Controls

As given in the project requirements, the map application needs some basic navigation controls to drag, move up, move down, move left, move right, zoom in and zoom out the map. Six buttons are created and when those graphical buttons are pressed, actions such as: move up, move down, move left, move right, zoom in, and zoom out are performed. Graphical buttons for these actions are placed as shown in Fig. 7.

First the locations of those graphical buttons are determined and cursor style is changed if the mouse reaches to those controls. For determining the mouse location while moving the mouse: `MouseMotionListener` event is implemented. The following source code example is a method to determine the location of mouse on left control.

```
if (mouseOnApplet) {
/*-----*/
//leftcontrol
if (_mousePosX1 > (_engine.getARROW_LEFT_X() - 1)
   && _mousePosX1 < (_engine.getARROW_LEFT_X()+ MapEngine.ARROW_WIDTH+1)
   && _mousePosY1 > (_engine.getARROW_LEFT_Y() - 1)
   && _mousePosY1 < (_engine.getARROW_LEFT_Y()+MapEngine.ARROW_HEIGHT+1))
{
    mouseOnMap = false;
    mouseOnLeftControl = true;
    setCursor(Cursor.getPredefinedCursor(Cursor.HAND_CURSOR));
}
//upcontrol
//downcontrol
//rightcontrol
//zoom in control
//Zoom out control
}
```

Source Code List 6. Source code to determine the location if mouse in the map application.

Firstly, the above source code checks whether the mouse is inside the applet or not. If the mouse cursor is inside the applet, the source code above determines the location of button and sets the clickable area by calculating button's height and width. Further the source code sets the `Boolean` variable "`mouseOnLeftControl`" to true. Now whenever the mouse is on this control, an action for left control can be performed by checking the `Boolean` variable "`mouseOnLeftControl`". Also for other controls the locations can be determined and action can be performed in the similar way.

3.5.2. Keyboard Controls

In Java, keyboard controls are handled by implementing `KeyListener`. There are four keys which are used to move map up, down, left and right. The keys are: up, down, left and right arrow key of keyboard. To perform action for these keys "`keyPressed`" event is implemented. The "`keyPressed`" event is lower-level and is dependent on the platform and keyboard layout. The "`keyPressed`" event is generated whenever a key is pressed. The key being pressed is indicated by the `getKeyCode` method, which returns a virtual key code. Virtual key codes are used to report which keyboard has been pressed, rather than character generated by the combination of one or more keystrokes (such as "A", which comes from "`shift + a`"). /7/ The list of `KeyEvent` is provided in Java doc (http://download.oracle.com/javase/6/docs/api/java/awt/event/KeyEvent.html). The following source code is an example of `keyPressed` event for up key in the keyboard.

```
public void keyPressed(KeyEvent e){
   if(mouseOnApplet){
        //up key
        if(e.getKeyCode() = e.VK_UP){
            setY(getY() - 20); // step up by 20 pixels
        }
   }
}
```

Source Code List 7. Source code example for keyPressed event in the map application.

In the above source code example, "`VK_UP`" is the constant for the non-numpad up arrow key, "`setY(), getY()`" are the methods to set and get "y" axis of the applet. The map is moved upward by 20 pixels when the up arrow key is pressed from keyboard.

3.6. Load Images on Demand

The original map image is split into several pieces, but only necessary images are loaded into applet to reduce memory consumption and to load the image fast. All the images are in the server. Therefore, loading large or many images would make application run slowly for even crash due to over memory consumption. Images are loaded according to the height and width of the applet, for example if the applet's height and width is 400 x 400 pixels, the required images for that height and width is loaded while the other images remain in the queue. The image below describes the overview of loading images into the applet only on demand.

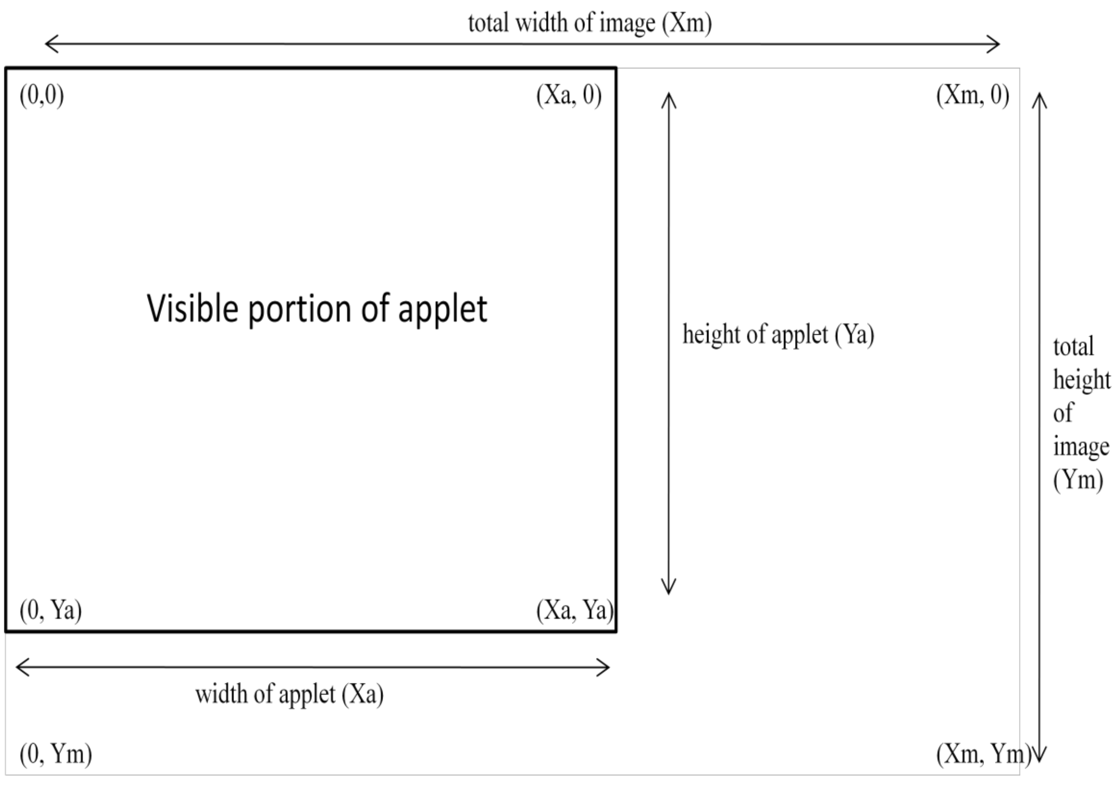


Fig. 9. Overview of loading images on demand

The big rectangle in the Fig. 9 is the total dimension of the images. Small pieces of map are plotted on this rectangle so that it can form an original map. The bold small rectangle is the visible portion of map which is also the dimension of applet. Maps are loaded and seen only on this visible portion of applet. First the area covered by applet is calculated by its dimension and the start position of the applet. After the total area covered by the applet is calculated, images on that area are counted and loaded into the applet. Images are plotted into large rectangle as shown in Fig. 8.

If the height and width of split image is 200px and 200px respectively, and height and width of the applet is 400px and 400px respectively, the number of images to be visible in the applet is 4 (400/200 = 2 in horizontal and 400/200 = 2 in vertical). If the applet start position is (0, 0) then the images to be loaded are: 0_0.jpg, 1_0.jpg horizontally and 1_0.jpg, 1_1.jpg vertically. The following source code shows the example of loading images into the applet on demand.

```
private void drawMap(Graphics g) {
// calculate scaled width and height from the view scale
double split_width = MapEngine.SPLIT_WIDTH *
                      _engine.getView().getScale();
double split_height = MapEngine.SPLIT_HEIGHT *
                      _engine.getView().getScale();
int cur_x = (int) (_engine.getView().getX() / split_width);
int max_x = (int) ((_engine.getView().getX() +
           _engine.getView().getWidth()) / split_height);
int cur_y = (int) (_engine.getView().getY() / split_height);
int max_y = (int) ((_engine.getView().getY() +
           _engine.getView().getHeight()) / split_height);

for(int y = cur_y; y<max_y + 1; y++){
for(int x = cur_x; x<max_x + 1; x++){
 if(_images[x][y] == null) {
    if(_imageLoading[x][y] == false) {
      _images[x][y] =
                      _parent.loadImage(MapApplet.IMAGE_SERVER
           _LOCATION+x+"_"+y+".jpg");
     //Set the images as loading
```

```
      _imageLoading[x][y] = true;
            }
} else {
Image scaledImage = scaleImage(_images[x][y],
                      _engine.getView().getScale());
g.drawImage(scaledImage, x*(int)split_width -
_engine.getView().getX(), y*(int)split_height-
_engine.getView().getY(), this);
         }
       }
    }
 }
```

Source Code List 8. Source code for loading images on demand

In above source code images are determined after looping and only valid images are loaded and drawn. The (x, y) coordinates of the applet is get through `getX()` and `getY()` methods respectively. Firstly, the locations for images are calculated and loaded only the valid images with the references of the applet’s and split image’s dimension. Then those images are drawn into applet according to their positions. `getScale()` method in above source code returns the zoom level of the map. The method `scaleImage()` scales given image with scale provided in its parameters.

3.7. Image Scaling

Images on the applet have to be scaled in different levels. The application has the feature to zoom in and zoom out the map for clear view of the locations of markers. Only the visible images are scaled and drawn into the applet. The `AffineTransform` class is used to scale images into different scale levels. The `AffineTransform` is a 2D affine transform that performs a linear mapping from 2D coordinates to other 2D coordinates. Affine transformations are the concept based in Euclidean algebra, and help to define ways to modify coordinates spaces so that parallelism and perpendicularity are preserved. /8/

The scale level is determined while pressing "zoom in" and "zoom out" button in the applet. The following source code shows the example of scaling image using the `Affine Transform` class in Java.

```
private Image scaleImage(Image image, double scale) {
     BufferedImage buffer;
     Graphics2D g;
     AffineTransform transformer;
     AffineTransformOp operation;
     buffer = new BufferedImage(image.getWidth(this),
     image.getHeight(this), BufferedImage.TYPE_INT_ARGB);
     g = buffer.createGraphics();
     g.drawImage(image, 0, 0, null);
     transformer = new AffineTransform();
     transformer.scale(scale, scale);
     operation = new AffineTransformOp(transformer,
                 AffineTransformOp.TYPE_BILINEAR);
     buffer = operation.filter(buffer, null);
     return Toolkit.getDefaultToolkit().createImage(buffer.getSource());
}
```

Source Code List 9. Source code for scalling image using `AffineTransformer` class.

In the above `Java Image` function, there are two parameters: image and scale. An `image buffer` object is created with the image provided in parameter and drawn in `Graphics2D`. A `transformer` scale is created with the provided scale value in second parameter of the function. The minimum scaling level is 0.1 and the maximum scaling level is 2.0. This scaling range helps to scale up image up to double of its original size and scale down up to one tenth of its original size. The scale level 1.0 means the original size of the image. The image buffer is converted and filtered using operation object of `TransformOp`. Finally an image is created using newly formed buffer image and returned.

3.8. Calculation of GPS Coordinates

Calculating GPS coordinates in the map is one of the most challenging features in this project. Since map resources are static JPEG images, determining GPS coordinates into the map with more accuracy is difficult. GPS coordinates in the map is calculated as accurate as possible.

3.8.1. Using Google Map as Reference

Though the map resources were expected to have some reference points for GPS coordinates, the initially provided map image did not have any reference points for GPS coordinates. Hence, the given map image was compared with Google Map before splitting into several pieces and two key points were marked and further calculations were done. Firstly, the easy points in map are determined so that it will be easier to find in Google Map.

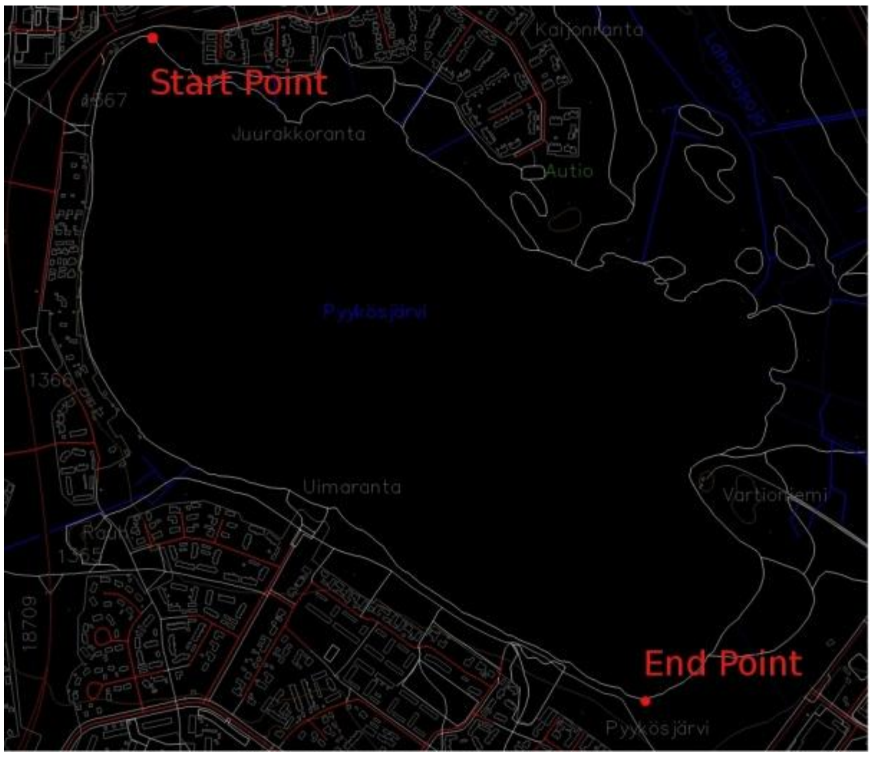


Fig. 10. Key points in map image

In the above Fig. 10, the red dots are the key points decided to take them as reference for determining GPS coordinates. The top most red dot is named as start point and the lower red dot is named as end point. Now Google Map is navigated to same view where those two points are marked. Google Map and map image are zoomed to approximately the same level as shown in Fig. 11.

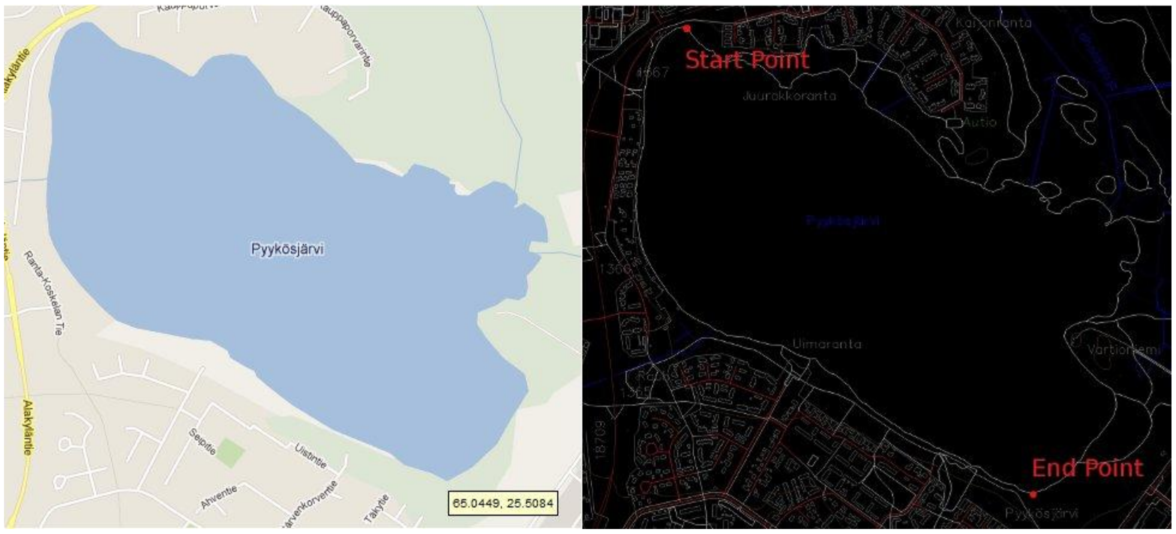


Fig. 11. Making same view in Google Map and provided map image.

In the Google map, when the mouse cursor is moved to one of the same point as in map image and SHIT button in keyboard is pressed, the GPS coordinates of that point (should have logged in with Google Account to use this feature in Google Map) is displayed. The first number of the displayed GPS coordinates is latitude and the second number is longitude. GPS coordinate is found out for second point with similar method. GPS coordinates found for the points during the development process of the application were:
The start point GPS data: (65.0588, 25.4835) => (latitude, longitude)
The end point GPS data: (65.0449, 25.5088) => (latitude, longitude)

Now let's assume the original map image is opened with GIMP 2 image editor. When the mouse cursor is moved on the image, the corresponding pixel value in the image is displayed in the bottom-left corner of the status bar. In the image below, the red arrow points at the pixels values of a point in GIMP 2. The first number is X value and the second is Y value.

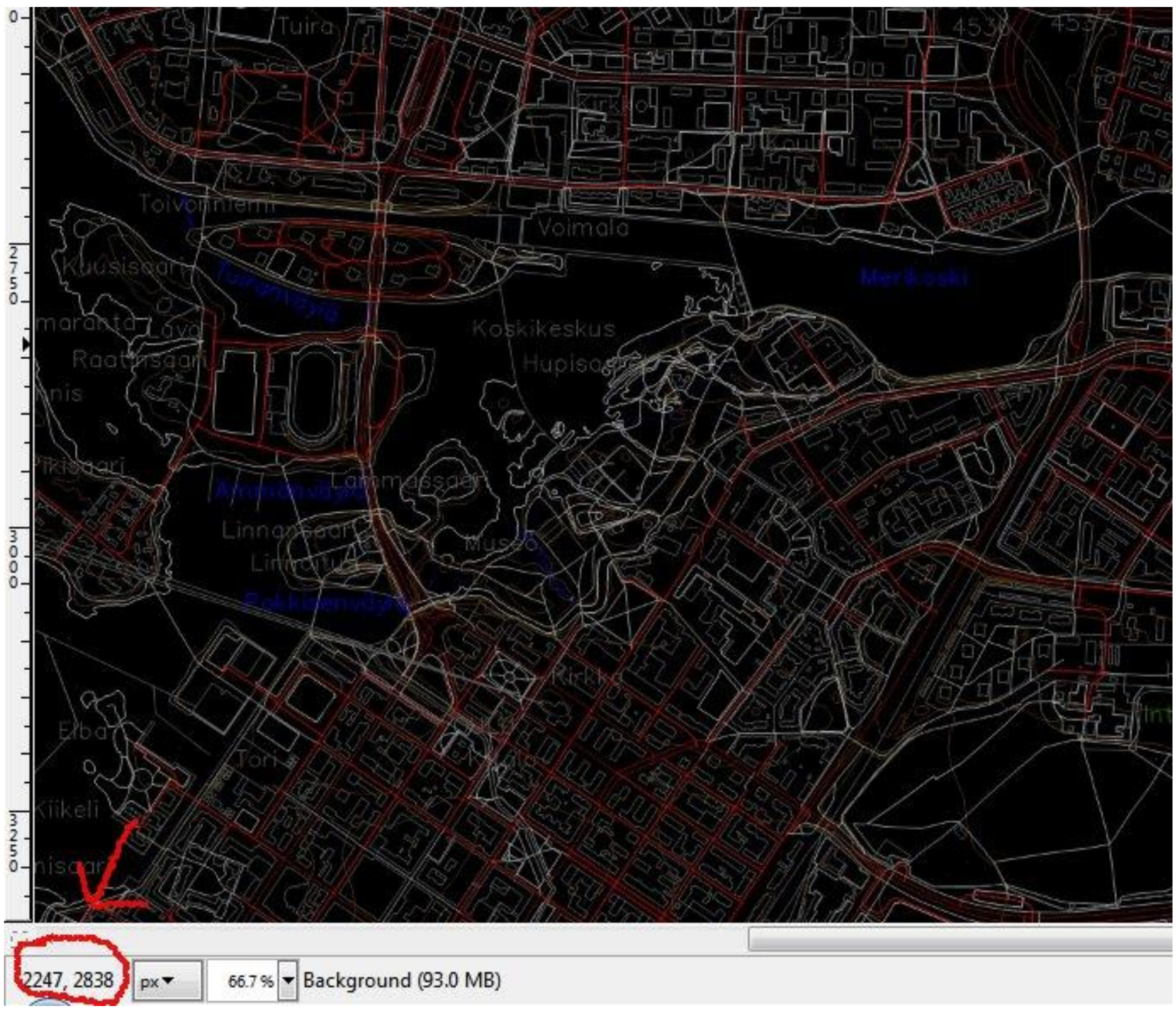


Fig. 12. Finding pixel value in the image using GIMP 2.

During the development process of the map application the pixel values of the two key points are determined by method mentioned above. The values are:

The start point pixel value: (2330, 728) => (x, y)

The end point pixel value: (2883, 1473) => (x, y)

After the pixel values and GPS coordinates are found for two key points, calculation is done to find GPS coordinates of (x, y) pixel in the map image. The mathematical calculation for finding GPS coordinates of (x, y) pixel in the map image with reference to data found above is mentioned below.

For finding longitude of given x:

Let us assume,

x be the x-pixel of given point,

X value of start point = Xs,

$\text{X value of end point} = \text{Xe},$

$\text{Logitude of start point} = \text{Ls},$

$\text{Longitude of end point} = \text{Le},$

$\text{Longitude difference per pixel} = \text{Ldiff},$

$\text{X difference} = \text{Xdiff},$

$\text{Longitude of x position} = \text{Lx}$

$$\text{Ldiff} = \frac{\text{Le} - \text{Ls}}{\text{Xe} - \text{Xs}}$$

$$\text{Xdiff} = \text{x} - \text{Xs}$$

$$\text{Lx} = (\text{Xdiff} * \text{Ldif}) + \text{Ls}$$

The image scale level is not included in this calculation. The x value of given point is a scaled x value.

In the similar way latitude of y is found.

For finding x of given longitude.

$\text{x be the x-pixel of the point to be found},$

$\text{Longitude of x position given} = \text{Lx},$

$\text{Scale of map} = \text{sc}, \text{(zoom level)}$

$\text{X value of start point} = \text{Xs},$

$\text{X value of end point} = \text{Xe},$

$\text{Logitude of start point} = \text{Ls},$

$\text{Longitude of end point} = \text{Le},$

$\text{Longitude difference} = \text{Ldiff},$

$\text{X difference per longiture} = \text{Xdiff},$

$$\text{Xdiff} = \frac{\text{Xe} - \text{Xs}}{\text{Le} - \text{Ls}}$$

$$\text{Ldiff} = \text{Lx} - \text{Ls}$$

$$\text{x} = \big((\text{Ldiff} * \text{Xdiff}) + \text{Xs}\big) * \text{sc}$$

In the similar way y of given latitude is found.

Note: There can be other different mathematical calculations for calculating more accurate GPS coordinates. The above given mathematical calculations were done only for thesis purpose by the student and the accuracy was about approximately 200meters compare to Google Map. The actual location of GPS coordinate is placed into the Google Map and the distance between the actual point and the point located by the map application is calculated with the help of online GPS distance calculator /33/.
If the start and end points are in long distance, the calculation gives the more accurate values.

3.8.2. GPS Value Reference from Static Map Image

There are few GPS coordinates reference provided in map image with their pixel coordinates. The mathematical calculations are done as shown above. The only difference between GPS value reference from static map image and Google Map is that the former key points’ values are more accurate than the values taken from Google Map.

3.9. Drawing Markers

There are two types of markers in this map application: single marker and multiple markers. Single marker is drawn into the map to view single vehicle and multiple markers are drawn to view the entire vehicles from the system.

3.9.1. Single Marker

Java applet is embedded into the web pages with HTML document. Single marker is drawn into map with the references of parameters’ values provided in HTML document. The HTML code given below is the structure for embedding applet into the HTML document for viewing single marker.

```
<APPLET code=map.MapApplet archive=Map.jar width=800 height=800>
   <PARAM name=appType value=singleLocation>
   <PARAM name=gpsLatitude value=25.478432>
   <PARAM name=gpsLongitude value=65.055439>
   <PARAM name=locationName value=Test Location>
   <PARAM name=color value=#FFFOOO>
</APPLET>
```

Source Code List 10. HTML code for embedding applet for single marker.

In the above HTML code there are five parameters between `<applet>` tags. The first parameter, “`appType`” gives the information about the type of applet view, “`singleLocation`” is for single marker and “overview” is for multiple markers. Second parameter, “`gpsLatitude`” contains the latitude of GPS coordinates of the vehicle. Third parameter, “`gpsLongitute`” contains the longitude of GPS coordinates of the vehicle. Fourth parameter, “`locationName`” contains the name for the marker; it can be vehicle’s number or driver’s name. The fifth parameter, “`color`” contains the hexadecimal color value for the marker in the map. All the values for parameters are retrieved from database. The application calculates the pixel coordinates for given GPS coordinates in the parameters (“`gpsLongitude`” and “`gpsLatitude`”) and draws the marker into the map. The map view is centered to the marker with defined color in parameter “color” and location name is visible when mouse cursor is moved over the marker.

3.9.2. Multiple Markers

Multiple markers are also drawn with the reference of parameters’ values provided in HTML document. The HTML code below is the structure of embedding applet into the HTML document for viewing multiple markers.

```
<APPLET code=map.MapApplet archive=Map.jar width=800 height=800>
   <PARAM name=appType value=overview>
   <PARAM name=xmlUrl value=http://localhost/projects/IMSS/test.xml>
</APPLET>
```

Source Code List 11. HTML code for embedding applet for multiple markers.

In the above HTML code there are two parameters between `<applet>` tags: `appType` and `xmlUrl`. Parameter, “`appType`” contains the applet view type. The second parameter, “`xmlUrl`” contains the location of XML file where the information about multiple markers is stored. The map application processes the XML file found in the server and draws the markers into the map. The contents of the XML file have been already discussed in section 2.4.

3.10. Java API for XML Processing

The Java API for XML processing (JAXP) is used for processing XML data using applications written in the Java programming language. JAXP enables the parser standards Simple API for XML Parsing (SAX) which also leverages Document Object Model (DOM) to parse data as a stream of events or build an object representation of it. /22/

XML-DEV group and the W3C define the SAX and DOM APIs. There are several libraries that are used to define those APIs. The libraries are as follows:

- `javax.xml.parsers`: This library provides a common interface for different vendors’ SAX and DOM parsers
- `org.w3.dom`: This library defines the Document class (a DOM) and classes for all the components of a DOM
- `org.xml.sax`: This library defines the basic SAX APIs /23/

In this project, the Simple API for XML (SAX) is used to parse XML data from server in order to draw multiple markers into the map. The SAX, being event-driven and serial-access mechanism, helps to do element-by-element processing. The **Figure 13** below presents the basic outline of the SAX parsing APIs. An instance of the `SAXParserFactory` class is used to start the process and generate an instance of the parser.

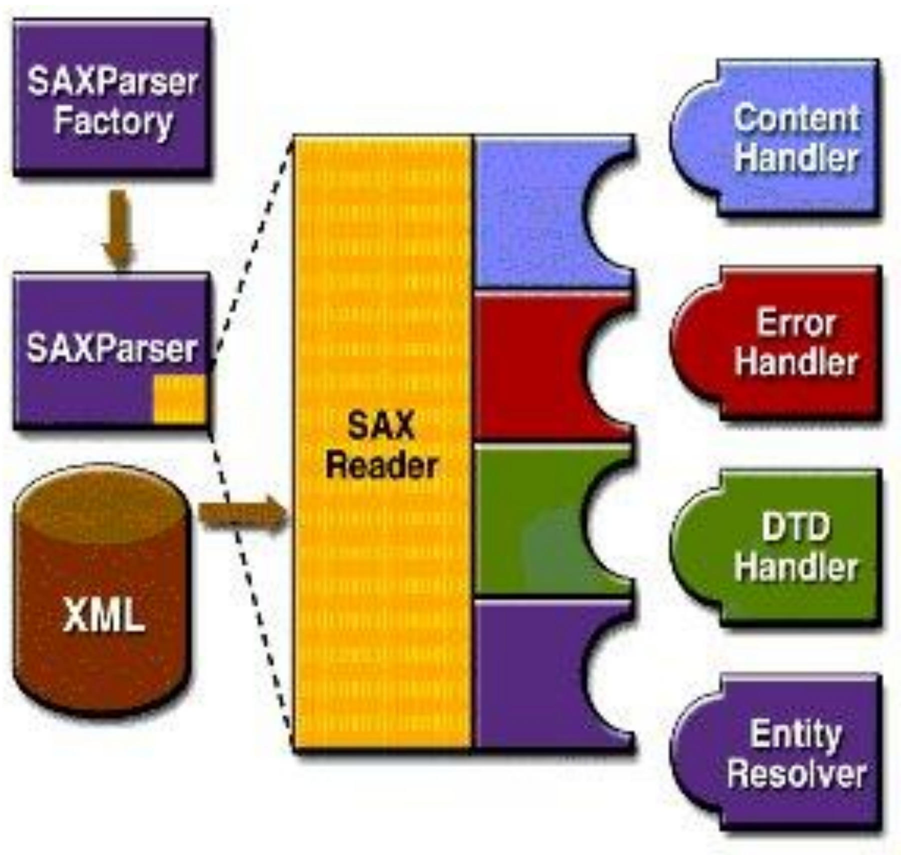


Fig. 13. Outline of SAX parsing /24/

`SAXReader` object is wrapped by the parser. When the parser's `parse()` method is called upon, the reader calls one of the several callback methods implemented in the application. The interfaces `ContentHandler`, `ErrorHandler`, `DTDHandler`, and `EntityResolver` define these methods. /24/

`SAXParserFactory`

A `SAXParserFactory` object creates an instance of the parser determined by the system property, `javax.xml.parsers.SAXParserFactory`./24/

`SAXParser`

Several kinds of `parse()` methods are defined in the `SaxParser` interface. In general, XML data source and a `DefaultHandler` object are passed to the parser, which processes the XML and calls the appropriate methods in the handler object. /24/

`SAXReader`

A `SAXReader` is wrapped by the `SAXParser`. Typically, what `SAXReader` does is not very important. `SAXParser`'s `getXMLReader()` is used to get hold of `SAXReader` so that the `SAXReader` can be configured. It is the `SAXReader` that carries on the conversation with the SAX event handlers defined by the users. /24/

`DefaultHandler`

The `ContentHandler`, `ErrorHandler`, `DTDHandler`, and `EntityResolver` interfaces (with null methods) are implemented by a `DefaultHandler` which can also be overriden. /24/

`Content Handler`

Methods such as `startDocument`, `endDocument`, `startElement`, and `endElement` are called when an XML tag is recognized. The methods `characters` and `processingInstruction` are also determined by this interface and are invoked when a parser encounters the text in an XML element or an inline processing instruction, respectively. /24/

`ErrorHandler`

In response to various parsing errors, Methods error, `fatalError`, and warning are invoked. The default error handler throws an exception for fatal errors while the other errors (including validation errors) are ignored. Therefore knowledge about SAX parser is important even if the DOM is used. Sometimes, the application may recover from the validation error. Other times, an exception may be generated. /24/

`DTDHandler`

DTD defines methods which are rarely called upon to use. The DTDHandler is used when processing a DTD to recognize and act on declarations for an unparsed entity. /24/

`EntityResolver`

When the data indentified by a URI is necessary to be identified by the parser, the resolveEntity method is invoked. In most cases, a URI is simply a URL, which is used to specify the location of a document. In other cases the document can be identified by a

URN – a public identifier, or name, that is unique in the web space. The public identifier can be specified in addition to the URL. The `EntityResolver` can then use the public identifier instead of the URL to find the document – for example, to access a local copy of the document if one exists. /24/

4. MEMORY OPTIMIZATION

Even though there has been the significant increase in the development of processor speeds but the development of memory speeds does not seem to have remarkable progress. Today's applications use more and more memory. Consuming more memory means diminishing the computer performance. Hence, to make the computer perform better, memory has to be optimized. Basically, memory can be optimized by two different approaches: hardware solution and software solution. Hardware solution approach means increasing physical memory size in the computer to increase the size of caches. Software approach provides programmer to exploit hardware sources freely according to the specific needs of the application. The application programmer might not want to worry about optimizing for a certain platform and doing the analysis necessary to make efficient optimizations. Java is used in all types of systems and memory settings and has been improved and adapted to be efficient and executable on different platforms. The Java Virtual Machine (JVM) is responsible for running all types of Java application. In JVM there are several ways to optimize memory such as: multi-threading, synchronization and memory management which need to be handled with great care. /1/

4.1. Memory Management in Java

When a new Java object is created, its data and control structures are allocated in the memory by the memory management system. After the use of the object (no pointers are pointing to it), its memory has to be reclaimed by a garbage collector. The Java language relies on a Java memory management system to allocate and de-allocate memory for the data structures used while using Java object. There are methods such as `System.gc()` and `Runtime.gc()` which are used to send request of Garbage collection to JVM but it is not guaranteed that garbage collection will happen. Garbage collection in Java is carried by a daemon thread called Garbage Collector. /1/

In the map application, few methods of software approach were done for optimizing the memory such as: only necessary images were loaded into the applet and disposed after use, java objects were disposed after using them. No hardware approaches were applied for memory optimization in this project. The map application is expected to run on latest computer with high speed processor and memory.

4.2. Image Buffering

In this project, images are buffered in order to make the application run smoothly. Buffering the large number of images means allocating large memory space which might make more memory consumptions. Therefore, only the needed images are buffered into memory. Sometimes images are drawn but they remain unused. These unused images are disposed using certain rules applied in the application. Only certain scales of drawn images are saved into the memory and the remaining images are disposed to free the system resources.

Java 2D™ supports loading external digital images into its `BufferedImage` format using its Image I/O API which is in the `javax.imageio` package. The following source code is an example of loading image from specific file. /20/

```
BufferedImage img = null;
try {
    img = ImageIO.read(new File("strawberry.jpg"));
} catch (IOException e) {
}
```

Source Code List 12. Source code example for loading image in Java.

`ImageIO.read()`is the most straightforward convenience API for most applications, but `javax.imageio.ImageIO` provides many more static methods for more advance usages of the Image I/O API. /20/

Image can be disposed by calling abstract method `dispose()` of `Graphics` class. This method disposes graphics content and releases any system resources being used. Basically when creating GUI, a large number of `Graphics` objects can be created within short time frame which need to be disposed after use. Garbage collector also disposes of the same system resources while finalizing process of creating GUI but it is better to free the associated resources manually by calling `dispose()` method. /10/

4.3. Threading

4.3.1. Process

A process is an instance of a computer program that is executed sequentially. Generally, a process has a complete, private set of basic run-time resources; in particular, each process has its own memory space. A process has several instructions which are executed simultaneously at the run time. Processes are often seen as synonymous with programs or applications. For instance, checking the spelling in a single process in the Word Processor program can also use other processes associated with this program such as printing, formatting, drawing, etc. /21/

4.3.2. Thread

A thread is a lightweight process that exists within a program and is executed to perform a special task. Threads exist within a process – every process has at least one thread. Threads share the process’s resources, including memory and open files. A process with multiple threads is called a multi-threaded process. In Java Programming language, thread is a sequential path of code execution within a program and has its own local variables, program counter, and lifetime. Threading concept is very important in Java through which the speed of any application can be increased. Every thread in Java is created and controlled by the `java.lang.Thread` class. When any standalone application is running, it firstly executes the `main()` method and runs in a thread. This thread is called the main thread. The main thread creates some other threads called child threads. The `main()` method execution can finish, but the program will keep running until all the threads have completed their execution. /11/ In Java, a thread can be created in two ways:

- By implementing the `Runnable` interface (`java.lang.Runnable`)

– By extending the `Thread` class (`java.lang.Thread`)

Implementing the `Runnable` interface

One way to create a thread in java is to implement `Runnable` interface and then instantiate an object of the class. The `run()` method is overridden into the class which is the only method that needs to be implemented. The `run()` method contains the logic of the thread. An example of creating a thread in Java with `Runnable` interface is given below.

```
public class HelloRunnable implements Runnable {
        public void run() {
           System.out.println("Hello from a thread!");
        }
        public static void main(String args[]) {
        (new Thread(new HelloRunnable())).start();
        }
}
```

Source Code List 13. Source code example for creating thread in Java with `Runnable` interface.

In the above example the `start()` method is invoked to start the tread. The thread ends when the `run()` method ends, either by normal completion or by throwing an uncaught exception. This approach of creating a thread by implementing the `Runnable` Interface must be used whenever the class being used to instantiate the thread object is required to extend some other class. /11/

Extending Thread Class

The `Thread` class itself implements `Runnable` Interface, though its `run()` method does nothing. Applications can have subclass `Thread`, providing its own implementation of `run()` method, as in the `runnable` interface method. The subclass may call a `Thread` constructor explicitly in its constructors to initialize the thread, using the `super()` call. An example of running thread by extending `Thread` class is given below.

```
public class HelloThread extends Thread {
    public void run() {
        System.out.println("Hello from a thread!");
        }
        public static void main(String args[]) {
            (new HelloThread()).start();
        }
    }
```

Source Code List 14. Source code example for creating thread in Java by extending `Thread` class.

The `start()` method can be invoked on the object to start the thread. The `Thread` class defines a number of methods useful for thread management. These include `static` methods, which provide information about, or affect the status of the thread invoking the method. The other methods are invoked from other threads involved in managing the tread and `Thread` object. /11/

In the map application, threads are not used much. Besides the java applet thread, one thread is used to keep the application running so that the correct images are drawn into the applet according the view of the applet.

5. TESTING

All kinds of applications are tested in order to check that the application is working as it is expected. According to Standish Group International, a consulting firm in Dennis, Mass: "Seven out of ten applications fail in some way upon deployment" /12/. Even though the application is developed by expert with best development tools and languages (like Java), it is impossible to produce defect-free code for the first time. Defects on application can put business at risk. Therefore, it is very important to test the application properly before using.

This project consists of three different applications: mobile application, server application and web based map application. Mobile application is used to send GPS coordinates to server and update the database. Map application uses the database on the server and do not have direct communication with mobile application. Hence, map application was tested alone in this project by retrieving several data from database.

This map application was tested on the computer running a version of Microsoft 32-bit operating system, Microsoft Windows 7 Professional Edition with the physical memory of 2GB. Various internet browsers such as Mozilla Firefox 4.0, Google Chrome and Windows Internet Explorer 9 were used to test the application loading from localhost. Each features of the map application were tested or debugged on the process of development. Errors found while debugging were fixed instantly. After developing all the features of the application and fixing the encountered debugging errors, the final testing was done. Various minor syntax errors were found in final testing. One major challenge found in final testing was memory consumption. The application was consuming more memory than expected which diminished the computer performance. Therefore, different approaches were applied for memory optimization such as: drawing only necessary images into the applet, disposing unnecessary images loaded into the memory, threading, and using garbage collection.

6. FURTHER DEVELOPMENT

This map application project is the product of IMSS Ltd. Oy Ab and they want to use this application for their business. The application can be further developed for commercial use by improving the features and adding some advanced features on it. Currently, the single marker view is not in real time update. To view the current location of single marker, the page has to be refershed. A new thread can be implemented to view the real time update of the single marker location. The multiple markers' locations are drawn by querying the XML file priodically from the server. This feature can be improved by creating communication protocols between the server and the map application. Third party map APIs such as Google and Bing Map can be implemented for better quality of map images and GPS accuracy. Some advance features such as street view, and vehicle search can be implemented to make this application more efficient and commercial.

Currently, the map application's user interface is very simple and supports basic functions. The user interface can be improved by adding more buttons for different features such as switching from multiple markers view to single marker view. In this thesis project, the GPS coordinates are calculated from the reference of two key points. The farther the key points are the more accurate the GPS coordinates are. More keypoints can be taken as reference for more accurate value.

The company is also thinking to develop the same application for smart phones and mobile devices. Same algorithms used in this project can be used while developing the map application for smart phones and mobile devices. Since mobile devices and smart phones have less resources, more optimization processes have to be done. There are different operating systems for mobile devices such as Windows Mobile, Symbian, Andriod, etc. Therefore, while developing the application for mobile devices, it is necessary to know for which operating system the application is being developed. Applications developed in Java Micro Edition Platform can run in most of the mobile operating systems. However, there could be limitations in features depending on the JVM of the mobile operating system.

CONCLUSIONS

The map application was further developed by the experts from the IMSS Ltd. Oy Ab using the source code developed during the thesis project. The application has been accepted by client company (Taxi Company) and is being used. Currently, about eighty vehicles are being tracked around Oulu region with the newly updated map.

It was a very useful experience to work as the software developer in IMSS Ltd. Oy Ab. Developing such an application was a very challenging task which required high level programming skills. Though I was a novice programmer, CTO (Chief Technical Officer) of IMSS Ltd. Oy Ab, Antti-Pekka Aaltonen, trusted me and provided motivation in doing this project, which was the great opportunity for me to develop my skills professionally in the information technology industry. During my thesis project, I got an opportunity to work deeply on Java technology and learnt to write the source code professionally.

Currently, the map application has the map of Oulu region. The map can be easily extended in this application, which can be further scope of the application. Although this project has three different applications: mobile, map (for desktop computers) and server application, I was involved in the map application only. There was no direct communication between the mobile and the map application. Therefore, it was possible to develop the map application independently. Since the map application was developed using Java applet technology, I got opportunity to work and understand the web services of Java.

In future, the company is planning to develop this application for smart phones and mobile devices, which would make this application more portable. Since the application was developed with Java programming language, it will be easier to further develop the application for mobile devices and smart phones on Java Platform using the same techniques and algorithms.

Working at IMSS, with essential experience, added a new dimension for future professional career as a software developer, which I am looking forward to embrace.

REFERENCES

/1/ Bell Doung, Make Java Fast: Optimize, [WWW-document], [http://www.javaworld.com/javaworld/jw-04-1997/jw-04-optimize.html], 4 September 1997.

/33/ Draft Logic, Google Maps Distance Calculator, [WWW-document], [http://www.daftlogic.com/projects-google-maps-distance-calculator.htm], 23rd May 2011.

/3/ Feigenbaum Barry, SWT, Swing or AWT: Which is right for you?, [WWW-document], [http://www.ibm.com/developerworks/grid/library/os-swingswt/], 21 Feb 2006.

/4/ Gramin, What is GPS?, [WWW-document], [http://www8.garmin.com/aboutGPS/], 20 June 2011.

/5/ GPS.GOV, Space Satelite Orbits, [WWW-document], [http://www.gps.gov/systems/gps/space/], 6 June 2011.

/6/ Horton, Ivor Beginning Java 2, 1.3, Wrox Press Ltd, 2001.

/7/ Java API Docs, Class KeyEvent, [WWW-document], [http://download.oracle.com/javase/6/docs/api/java/awt/event/KeyEvent.html], 20 July 2011.

/8/ Java API Docs, Class AffineTransform, [WWW-document], [http://download.oracle.com/javase/1.4.2/docs/api/java/awt/geom/AffineTransform.html], 20 June 2011.

/9/ Java API Docs, Class MediaTracker, [WWW-document], [http://download.oracle.com/javase/1.4.2/docs/api/java/awt/MediaTracker.html], 30 August 2011.

/10/ Java API Docs, Class Graphics, [WWW-document], [http://download.oracle.com/javase/1.3/docs/api/java/awt/Graphics.html], 20 June 2011.

/11/ Java beginner.com, Java Threads Tutorial, [WWW-document], [http://www.javabeginner.com/learn-java/java-threads-tutorial],5 August 2011.

/12/ Mercury Interactive Corporation, FundingUniverse, [WWW-document], [http://www.fundinguniverse.com/company-histories/Mercury-Interactive-Corporation-Company-History.html], 30 August 2011.

/13/ NetBeans, NetBeans IDE 6.9.1 Release Notes, [WWW-document], [http://netbeans.org/community/releases/69/relnotes.html#system_requirements], 23 August 2010.

/14/ NetBeans, NetBeans IDE 7.0 Features, [WWW-document], [http://netbeans.org/features/index.html], 10 June 2011.

/15/ NetBeans, NetBeans IDE 7.0 Features, [WWW-document], [http://netbeans.org/features/ide/editor.html], 10 June 2011.

/16/ Nourine Dana, Java Technologies for Web Applications, [WWW-document], [http://www.oracle.com/technetwork/articles/javase/webapps-1-138794.html], November 2006.

/17/ Nourie Dana and Monica Pawlan, Introducing the Java Platform, [WWW-document], [http://www.oracle.com/technetwork/topics/newtojava/intro-139083.html], July 2007.

/18/ Oracle Sun Developer Network (SDN), Applets, [WWW-document], [http://java.sun.com/applets/], 15 July 2011.

/19/ Oracle, Learning Swing with the Netbeans IDE, [WWW-document], [http://download.oracle.com/javase/tutorial/uiswing/learn/index.html], 16 July 2011.

/20/ Oracle, Concurrency, [WWW-document],
[http://download.oracle.com/javase/tutorial/essential/concurrency/], 6 August 2011.

/21/ Oracle, Concurrency, [WWW-document],
[http://download.oracle.com/javase/tutorial/essential/concurrency/], 6 August 2011.

/22/ The J2EE 1.1, Java API for XML Processing, [WWW-document],
[http://download.oracle.com/javaee/1.4/tutorial/doc/JAXPIntro.html], 25 July 2011.

/23/ The J2EE 1.1, An Overview of the Packages(JAXP), [WWW-document],
[http://download.oracle.com/javaee/1.4/tutorial/doc/JAXPIntro3.html], 26 July 2011.

/24/ The J2EE 1.1, An Overview of the Packages(JAXP), [WWW-document],
[http://download.oracle.com/javaee/1.4/tutorial/doc/JAXPIntro3.html], 26 July 2011.

/25/ W3C, Extensible Markup Language (XML), [WWW-document],
[http://www.w3.org/XML/], 23 April 2011.

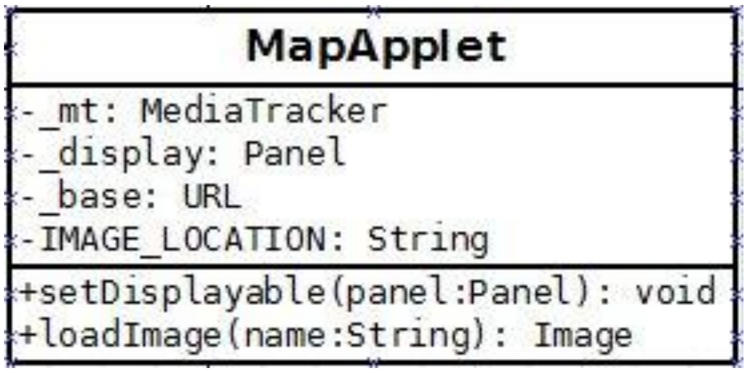
/26/ W3schools.com, XML Tutorial, [WWW-document],
[http://www.w3schools.com/xml/xml_usedfor.asp], 5 July 2011.

LIST OF APPENDICES

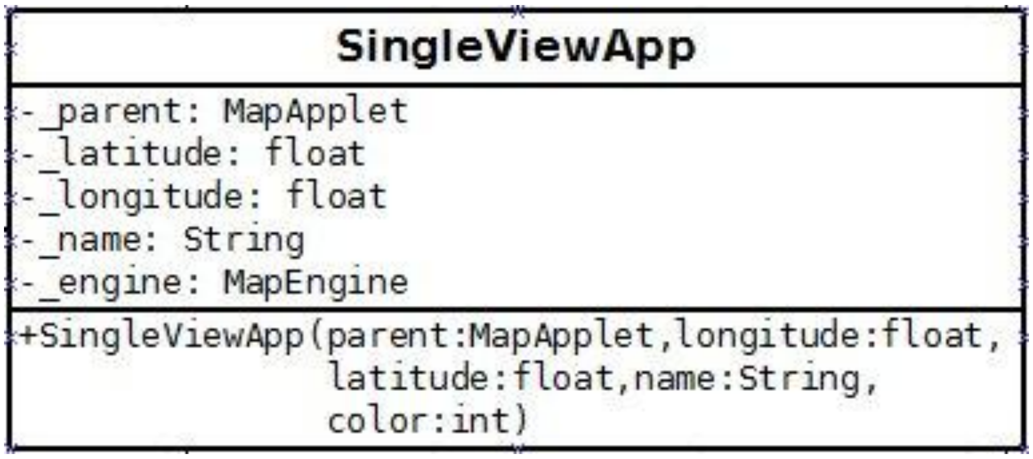
APPENDIX 1/1

Java Classes of Map Application

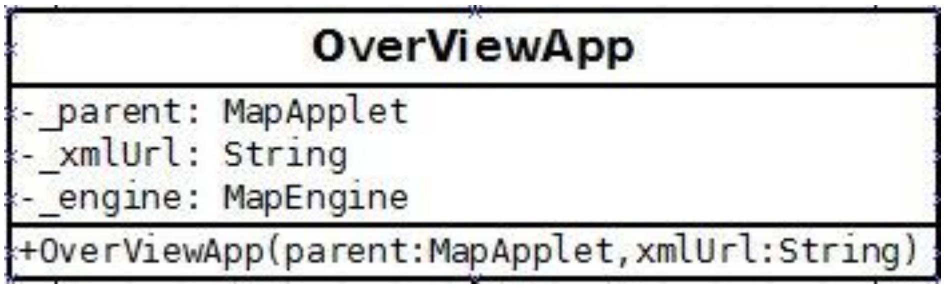
MapApplet.java



SingleViewApp.java

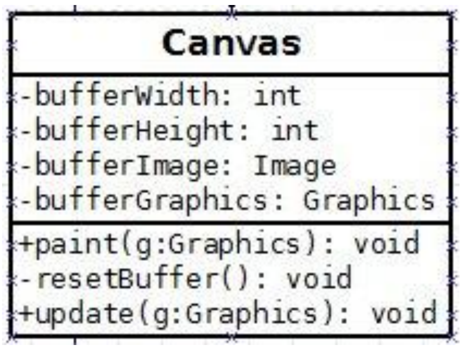


OverViewApp.java

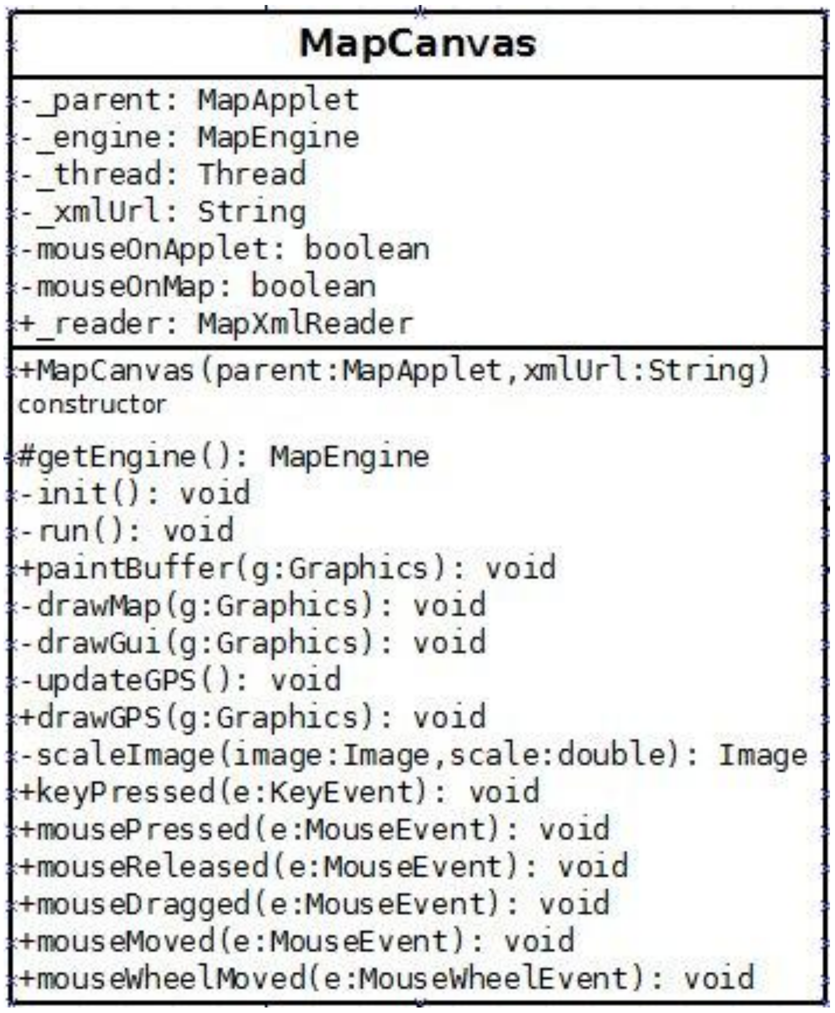


APPENDIX 1/2

Canvas.java



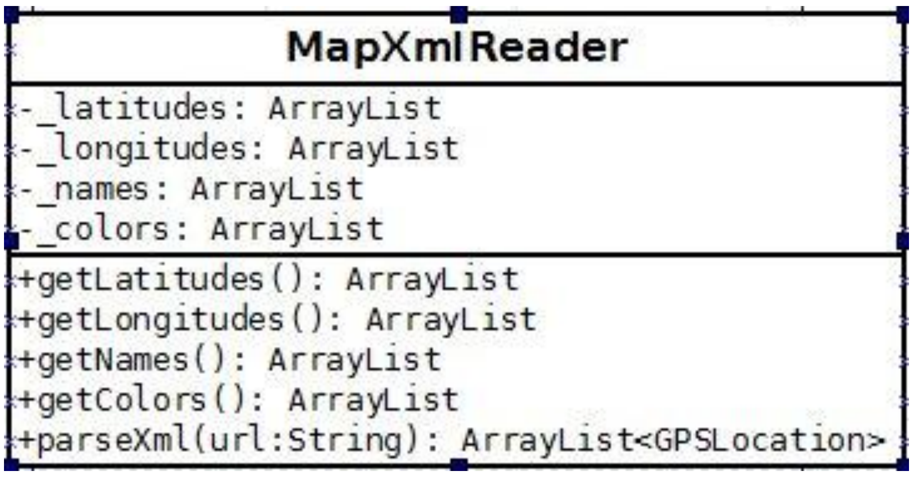
MapCanvas.java



MapCanvas
-_parent: MapApplet
-_engine: MapEngine
-_thread: Thread
-_xmlUrl: String
-mouseOnApplet: boolean
-mouseOnMap: boolean
+_reader: MapXmlReader
+MapCanvas(parent:MapApplet,xmlUrl:String)
constructor
#getEngine(): MapEngine
-init(): void
-run(): void
+paintBuffer(g:Graphics): void
-drawMap(g:Graphics): void
-drawGui(g:Graphics): void
-updateGPS(): void
+drawGPS(g:Graphics): void
-scaleImage(image:Image,scale:double): Image
+keyPressed(e:KeyEvent): void
+mousePressed(e:MouseEvent): void
+mouseReleased(e:MouseEvent): void
+mouseDragged(e:MouseEvent): void
+mouseMoved(e:MouseEvent): void
+mouseWheelMoved(e:MouseWheelEvent): void

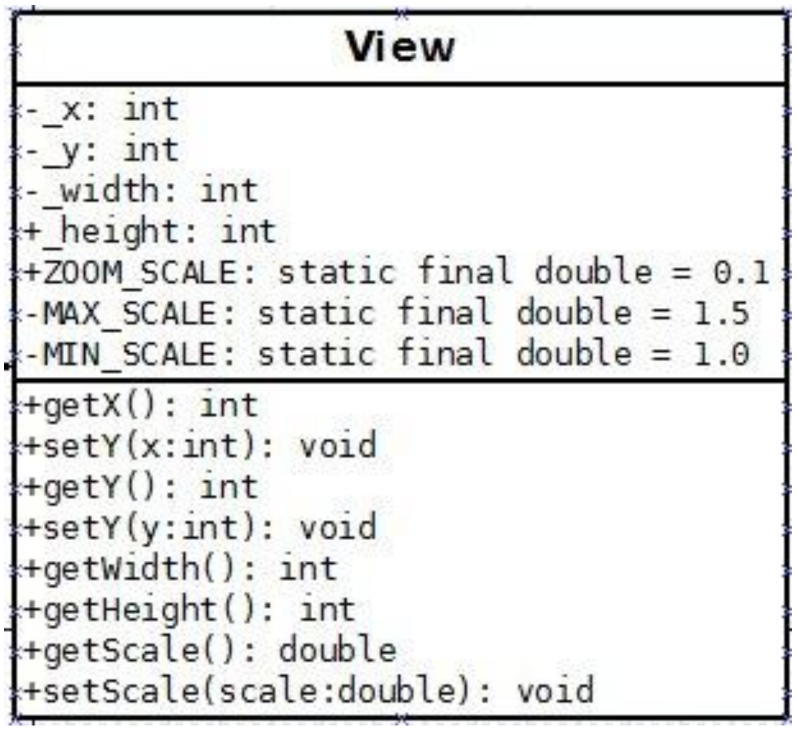
APPENDIX 1/3

MapXmlReader.java



```
MapXmlReader
-_latitudes: ArrayList
-_longitudes: ArrayList
-_names: ArrayList
-_colors: ArrayList
+getLatitudes(): ArrayList
+getLongitudes(): ArrayList
+getNames(): ArrayList
+getColors(): ArrayList
+parseXml(url:String): ArrayList<GPSLocation>
```

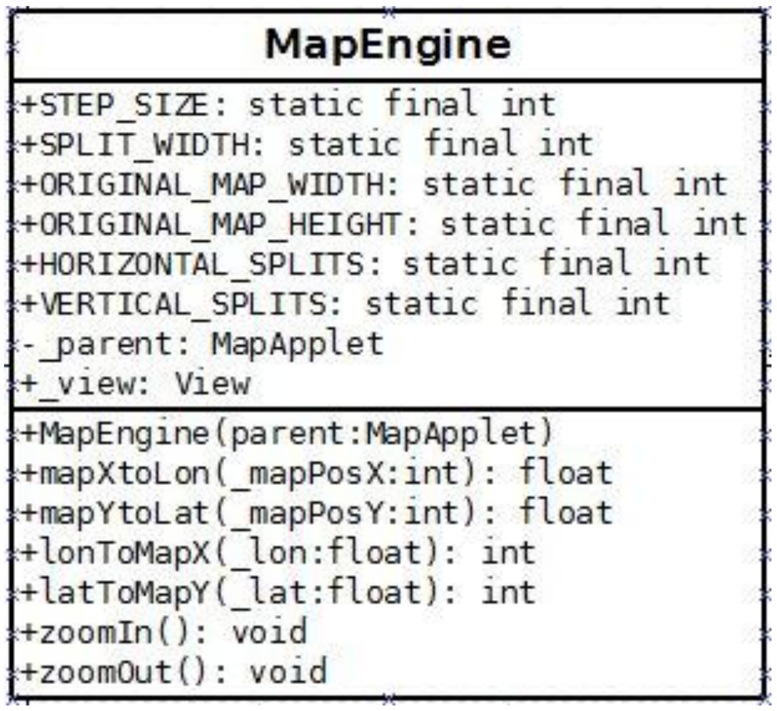
View.java



```
View
-_x: int
-_y: int
-_width: int
+_height: int
+ZOOM_SCALE: static final double = 0.1
-MAX_SCALE: static final double = 1.5
-MIN_SCALE: static final double = 1.0
+getX(): int
+setY(x:int): void
+getY(): int
+setY(y:int): void
+getWidth(): int
+getHeight(): int
+getScale(): double
+setScale(scale:double): void
```

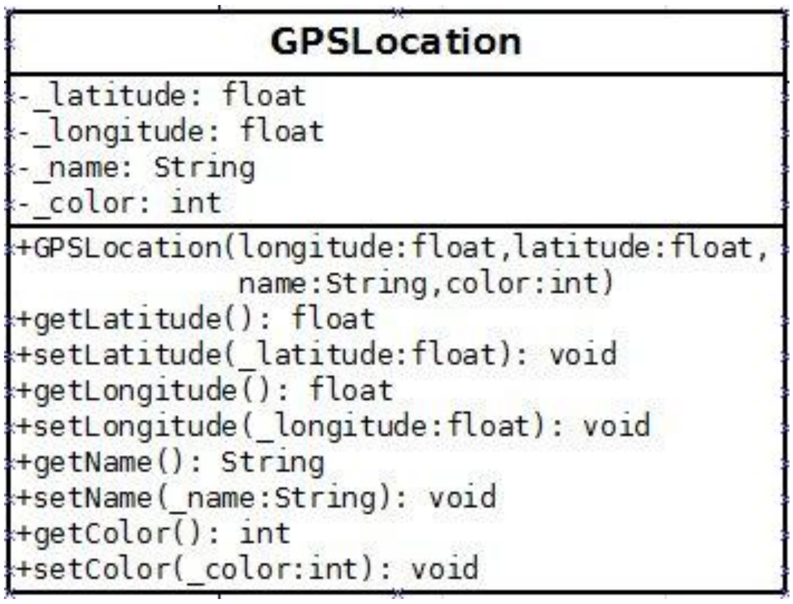
APPENDIX 1/4

MapEngine.java



```
MapEngine
+STEP_SIZE: static final int
+SPLIT_WIDTH: static final int
+ORIGINAL_MAP_WIDTH: static final int
+ORIGINAL_MAP_HEIGHT: static final int
+HORIZONTAL_SPLITS: static final int
+VERTICAL_SPLITS: static final int
-_parent: MapApplet
+_view: View
+MapEngine(parent:MapApplet)
+mapXtoLon(_mapPosX:int): float
+mapYtoLat(_mapPosY:int): float
+lonToMapX(_lon:float): int
+latToMapY(_lat:float): int
+zoomIn(): void
+zoomOut(): void
```

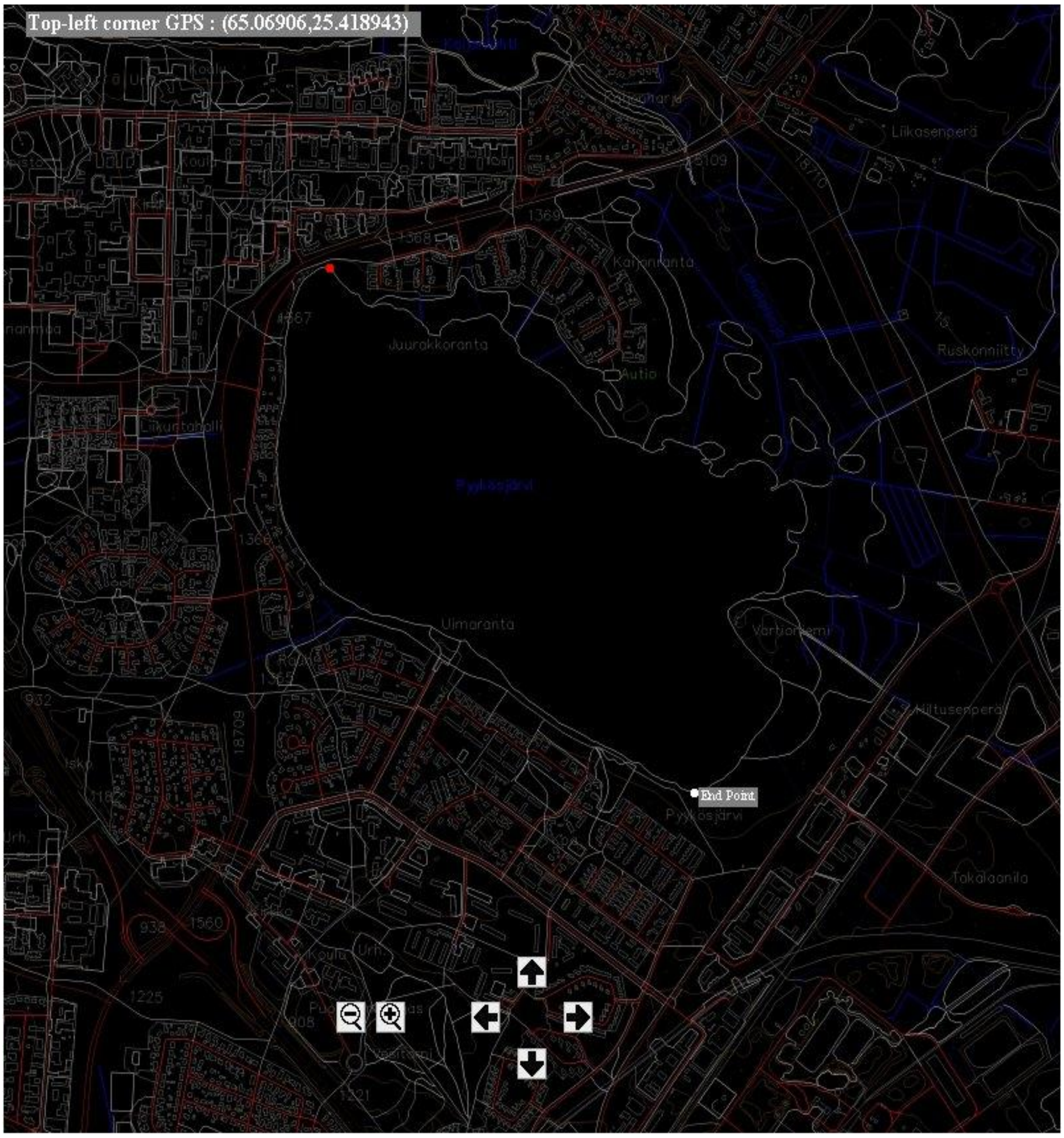
GPSLocation.java



```
GPSLocation
-_latitude: float
-_longitude: float
-_name: String
-_color: int
+GPSLocation(longitude:float,latitude:float,
             name:String,color:int)
+getLatitude(): float
+setLatitude(_latitude:float): void
+getLongitude(): float
+setLongitude(_longitude:float): void
+getName(): String
+setName(_name:String): void
+getColor(): int
+setColor(_color:int): void
```

APPENDIX 2

Screen shot of the map application during final testing


Top-left corner GPS : (65.06906,25.418943)
Juurakkoranta
Pyykösjärvi
Ulmaranta
End Point
Liikasenperä
Ruskonniitty
Vartioniemi
Hiltusenperä
Takalaanila