KEMI-TORNIO UNIVERSITY OF APPLIED SCIENCES

TECHNOLOGY

ZHUANG CHENG

Research and Implementation Of Mobile Phone Dictionary System Based on J2ME

The Bachelor's Thesis Information Technology Programme
Kemi 2010

PREFACE

I would like to thank all the people who have helped me in producing this thesis project, especially my instructor Mr. Thai Bui for his support.

ABSTRACT

Kemi-Tornio University of Applied Sciences

Degree programme	Information Technology
Author	Zhuang Cheng
Subject	Research and Implementation of Mobile Phone Dictionary System Based on J2ME
Type of Study	Bachelor's Thesis
Date	29 November 2010
Pages	30 + 4 appendixes
Instructor	Thai Bui

The objective was to research a new application for mobile users; that application should be helpful and convenient for human beings' usual life. The application was designed to be run on the mobile phones supported with Java.

The research methods were programming mostly. J2ME is the main programming language in this thesis research. This thesis was an individual work. The research was written on the basis of the information which was available and this is the author's view in how the sensible things should be implemented there.

The result of my thesis work is an application called EFDictionary, which attempts to help people to translate languages between English and Finnish. With the help from Mr. Thai Bui, the research progresses to the final results and the mobile application could run well on the emulator.

Key word: Dictionary, Java, J2ME

TABLE OF CONTENTS

1 INTRODUCTION 1
1.1 Objectives of thesis 1
1.2 Principle of project 1
1.3 Structure of the thesis 1
2 J2ME 2
2.1 J2ME Overview 2
2.1.1 Java 2 Platforms 2
2.1.2 Introduction to J2ME 3
2.2 Configurations and Profiles 4
2.2.1 Configuration 4
2.2.2 Profile 5
2.3 J2ME Architecture 6
2.4 MIDP 7
2.4.1 MIDP Architecture 7
2.4.2 MIDP Applications (MIDlets) 7
2.4.3 MIDlet Lifecycle 8
2.4.4 MIDlet development process 9
2.5 J2ME Development Environment 10
2.5.1 J2ME WTK 10
2.5.2 IDE (Integrated Development Environment) 10
3. DESIGN 11
3.1 Motivation and background information 11
3.2 Functionality 12
3.3 Analysis of Application Needs 12
3.4 Target Goals 13
3.5 Considerations During in Design 13
3.6 System Design 15
3.6.1 Activity Diagram 15
3.6.2 Flow Chart 17
3.6.3 Class Diagram 19
3.7 Application Running Platform 20
4. IMPLEMENTATION 21
4.1 GUI (Graphic User Interface) 21
4.1.1 User Interface Architecture 21
4.2 Database Structure 22
4.3 Algorithms of Searching Target Words 24
5. EVALUATION 27
6. CONCLUSIONS 28
REFERENCES 29
Listing the sources: 29
Appendix I

Guide in practical use...I

EXPLANATION OF CHARACTERS AND ABBREVIATIONS

J2ME	Java 2 platform, Micro Edition
J2SE	Java 2 platform, Standard Edition
J2EE	Java 2 platform, Enterprise Edition
PDA	Personal digital assistant
RAM	Random-Access memory
TV	Television
GPS	Global Positioning System
API	Application programming interface
CLDC	Connected Limited Device Configuration
UI	User interface
CDC	Connected Device Configuration
MIDP	Mobile Information Device Profile
PDAP	PDA Profile
RMI	Remote Method Invocation
MID	Mobile Information Device
JVM	Java Virtual Machines
JAD	Java Application Descriptor file
JAR	Java Archive file
WTK	Wireless Toolkit
IDE	Integrated Development Environment
SMS	Short Message Service
OS	Operating system
GUI	Graphic User Interface
AWT	Abstract Windows Toolkit
LCD	Liquid crystal display
2D	Two-dimension
BREW	Binary Runtime Environment for Wireless
AMS	Application Management Software

TERMS

EFDictionary	The application were built for this thesis research
Java	A programming language developed by SUN company
MIDlet	MIDP Applications

1 INTRODUCTION

1.1 Objectives of thesis

The purpose of this thesis is to study and build a convenient mobile application, and make the user could be benefit from using the use of EFDictionary.

EFDictionary was designed to the customers who are unacquainted to either English or Finnish. For the customer, the usability and convenience are very worth considering. So, mobile software is a good method to expand my idea.

The EFDictionary application is worked based on J2ME platform, which means, Java programming language is the most used. And any mobile phones which support Java should be capable to run this application.

1.2 Principle of project

During the thesis work, the knowledge of J2ME platform, process of MIDP Application development, and mobile phone should be involved.

The primary target was to develop a new application of Finnish- English dictionary that can be implemented on mobile phones supporting Java.

1.3 Structure of the thesis

Firstly, there will be a brief about Java and J2ME, some special features regarding J2ME MIDlet development; also some summarization about Netbeans development environment.

After that, my opinion of design is introduced.

Next, I explain how I implement the application, and give a few illustrations of codes.

At the end, a final conclusion of EFDictionay which is according to this thesis research will be give.

2 J2ME

In this chapter, there are some elementary information of J2ME are given.

2.1 J2ME Overview

Background information of J2ME and J2ME platforms are discussed in the next subchapters.

2.1.1 Java 2 Platforms

Recognizing that one structure can not adapt to all the circumstances, therefore, Sun divided Java into three different versions, each version for today's computer industry in a special field.

- J2ME (Java 2 platform, Micro Edition): Specifies several different sets of libraries (known as profiles) for devices which are sufficiently limited that supplying the full set of Java libraries would take up unacceptably large amounts of storage.

- J2SE (Java 2 platform, Standard Edition): For general purpose use on desktop PCs, servers and similar devices.

- J2EE (Java 2 platform, Enterprise Edition): Java SE plus various APIs useful for multi-tier client-server enterprise applications.

Fig.1. Structure of Java 2 platforms /5/

Fig.1 shows the different versions of Java, and their target markets, from the top side of the most high-end to the bottom side of the most low-end. Basically, it identifies their target markets as well as the types of their target devices. J2EE provides enterprise-class computer servers and support, J2SE provides support for desktop and personal computers, J2ME provides high-end and low-end devices.

- High-end device: Typical products from this class of device are like set-top box, video phone, car entertainment/navigation systems and so on. Those devices are capable at processing huge amount of user interfaces and dependent on broadband-Internet connections.
- Low-end device: like mobile phones, PDAs which devices are with ordinary user interfaces, limited RAM and narrowband-Internet or intermittent network communication.

2.1.2 Introduction to J2ME

J2ME was launched by SUN Company in June 1996, as a new Java version to provide application development especially for the small resource-constrained consumer electronics devices. J2ME now has been widely used in cell phones, PDAs, car navigation systems as well as TV set-top boxes and many other devices; it has a very good development prospects.

Its major technical advantages are: has a good cross-platform capability to achieve the superiority of which called "write once, run anywhere"; with a seamless integration with J2EE capacity; retains excellent Java language characteristics, such as simple,

safe; and the existing Java platform, a wide range of development tools, enterprises, developers can provide a good J2ME support.

At this stage, the most popular application for J2ME is game software. Compared to its competitors such as BREW /15/ (an application development platform created by Qualcomm) which needs for expensive special equipments and development tools, J2ME program can be developed on PC and run by emulator device, and then easily deployed to the target device; it makes the development, testing and release become easy and inexpensive. Because once a J2ME application was developed, it could be run on any kind of platforms. In fact, uses of prospects of J2ME are much broader.

2.2 Configurations and Profiles

As a Java development platform, J2ME was specially designed and developed for the mobile devices with small memory capacity and low-performance processor from the advent of beginning. On the surface, it is good for those developers who wish to enhance the portability of the application; however, the term of "mobile device" covers a very wide range of areas, Including PDAs, pagers, smart phones and even GPS equipments and so on, they all have different hardware configuration. Similarly, the market for such equipment is also independent; there is an issue about compatibility between two kinds of device from the same supplier, let alone devices from different manufacturers. This is almost impossible to provide an universal development platform for these wide variety of mobile devices. As a result, J2ME defines two types of specifications which work together to provide a mobile Java platform; these two types of specifications are Configuration and Profile.

2.2.1 Configuration

The J2ME platform covers a large range of different devices types. Due to the large range of different type devices in the J2ME market place, Sun has split the J2ME in configurations. Configurations define virtual machines features, Java language features and Java API classes for each configuration environment. To avoid fragmentation that could lead to confusion, Sun has introduced only two configurations so far, which are:

CLDC: Connected Limited Device Configuration

The CLDC was the first configuration to be defined as it is bound to the main target group: small toys or gadgets, which always are carried around by the user, e.g. mobile phones or PDAs. So the market consisting of personal, mobile, connected information devices is served by the CLDC. This kind of devices require following factors /14/:

- ✓ Very simple user interface (including no UI at all).

- ✓ Low level memory budgets (160 kb to 512 kb).
- ✓ 16 or 32 bit processors.
- ✓ Wireless communication, possibly low bandwidth.
- ✓ Limited power, often battery operation.

CDC: Connected Device Configuration

The CDC serves the market consisting of shared, fixed, connected information devices. In order to get support from CDC, mobile devices must have at least the following conditions /14/:

- ✓ Large range of user interface capabilities (down to and including no UI).
- ✓ Memory budgets in range of 2 to 16 megabytes.
- ✓ 16 or 32 bit processors.
- ✓ Connectivity to some type of network.

2.2.2 Profile

The configurations help already a lot to differentiate different types of devices. But as there plenty types around they still have to be refined further because a configuration might cover devices intended for totally different usage, like mobile phones and washing machines. The mobile phone and the washing machine could belong to the same configuration but it is obvious that a mobile phone application is not expected to run on the washing machine or other devices. Thus, the J2ME framework provides the concept of profiles to achieve application environment portability.

Profiles address the specific demands of device category. Because of this, profiles may include libraries that are far more device category specific than the libraries provided in a configuration. A profile consists of classes that enable developers to implement features found on a related group of small computing devices.

There are several profiles in various stages of development:

- Profiles used with CLDC:
  - Mobile Information Device Profile(MIDP)
  - PDA Profile(PDAP)

- Profiles used with CDC:
  - Foundation Profile
  - Game Profile
  - Personal Profile
  - Personal Basis Profile
  - RMI Profile

2.3 J2ME Architecture

The system structure of J2ME is based on the devices. A class defines particular types of device: mobile phone, Beep-Pager and PDA each is a simple category.

Sun Company designs J2ME system structure as modularized in order to support the flexibility and customization for CLDC devices. J2ME defines the modularization and scalability at the time of a complete application running; the three layers in this model are constructed in the host operating system on the device. Fig.2 shows fours layers of J2ME architecture.

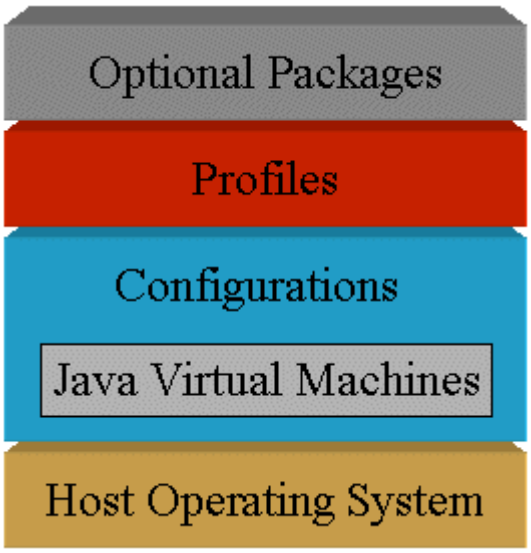


Fig.2. J2ME Architecture /6/

- Host Operating System: this layer for specific hardware device's operating system.
- JVM (Java Virtual Machines): this layer is an implement of JVM; it is made for host operating system of special devices, to support a particular J2ME configuration.
- Configurations: configuration layer defines the JVM's functions; it handles interactions between the profile and the JVM.
- Profiles: it consists of the minimum set of application programming interfaces for the small computing device.

JVM layer, configuration layer and profile layer together for the application, provide a standard operation environment which allows new applications and services to be installed to various end-user devices.

2.4 MIDP

MIDP is the Profile locates in the upper CLDC, which is the relatively mature developing and best-known Profile on J2ME platform

2.4.1 MIDP Architecture

Mobile Information Device Profile (MIDP) defines the Java application environment for mobile information devices (MIDs), such as mobile phones and personal digital assistants (PDAs). MIDP is part of the Java™ 2 Platform, Micro Edition (J2ME™). Fig.3 illustrates the MIDP architecture.

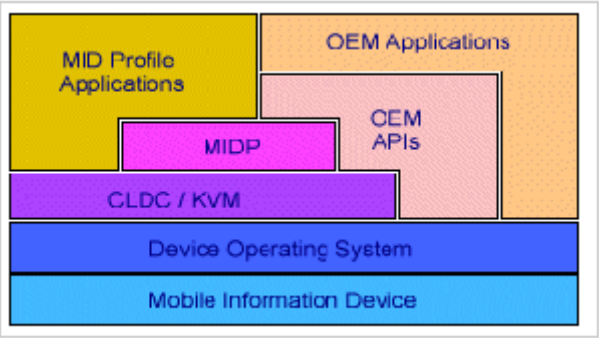


Fig.3. MIDP Architecture /7/

2.4.2 MIDP Applications (MIDlets)

The applications written for mobile information devices such as mobile phones are called MIDlets. A MIDlet is a well behaved MIDP application which lives within the resource constraints which runs and terminates when requested.

A complete MIDP application is consist of a JAD (Java Application Descriptor file) and JAR (Java Archive file). JAD describes the MIDlets that are distributed as JAR files; JAR aggregates many files into one, such as Class files, Resource files and Manifest files. Fig.4 shows the MIDlet packing.

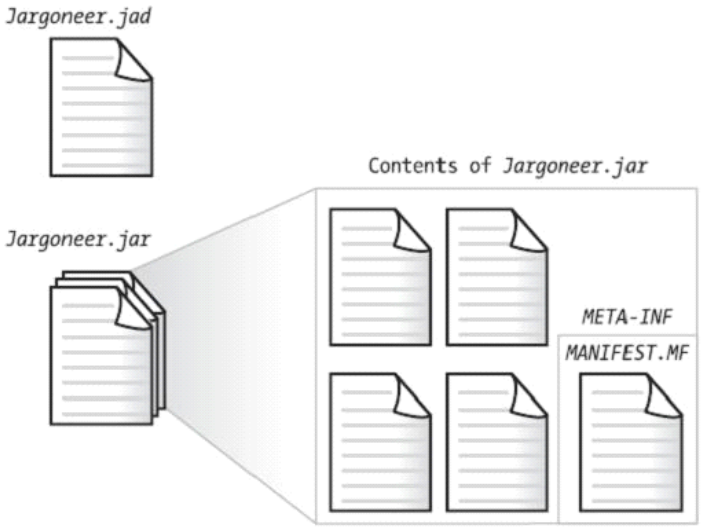


Fig.4. MIDlet packing /8/

2.4.3 MIDlet Lifecycle

A MIDlet can be in one of the three states (Active, paused, destroyed) after it is launched by the Application Management Software.
A MIDlet's main class extends javax.microedition.midlet.MIDlet. The main class defines three life-cycle notification methods: startApp(), pauseApp(), and destroyApp().

There are three possible states in a MIDlet's life-cycle:

- Active: The MIDlet is active. Starting the MIDlet application from the device User Interface when the application is executed by default startApp() method will call.

- Paused: The MIDlet instance has been constructed and is inactive. This state is usually entered in the following ways: From the Active state after the pauseApp() method is called from the AMS (Application Management Software) and Paused from a running state by an interruption event such as an incoming phone call.

- Destroyed: The MIDlet has been terminated and is ready for reclamation by the garbage collector. The MIDlet releases all of its resources and terminated when this state is on. This state is entered possibly by reasons such as from the Active or Paused state, after the detroyApp() or notifyDestroyed() method is called and returns successfully, Destroyed from a running state via a user directive from the MIDlet menu or device was accidently shut off.

Fig. 5 illustrates the switches among different states.

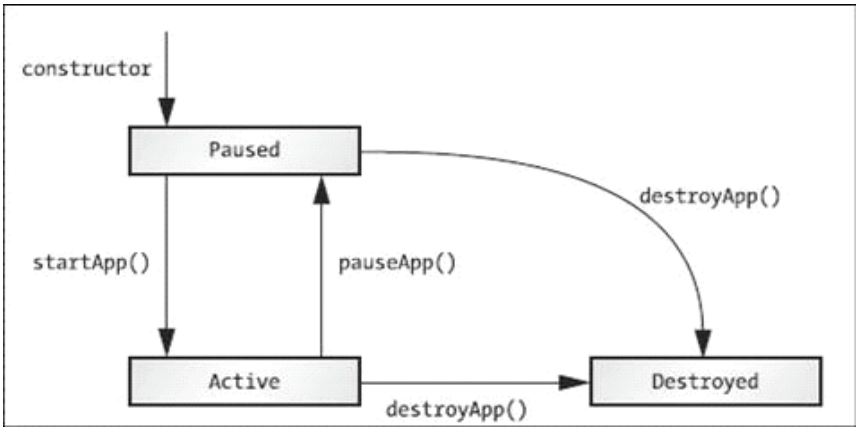


Fig.5. MIDlet Lifecycle /9/

2.4.4 MIDlet development process

During the processes of developing a MIDlet, it could be seven separately steps in total there. By the proper order of creation, there are designing, coding, compiling, preverification, packaging, testing, and deployment.

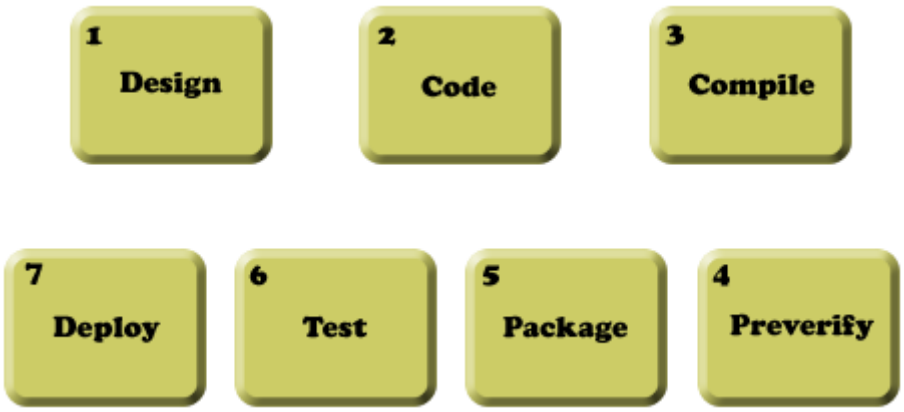


Fig.6. Steps to MIDlet Development /10/

Fig.6 describes the steps of MIDlet development.

1. Design: this part happens before the coding job, mostly sketching for the user interfaces and user interactivity.
2. Code: coding of methods and functions for the expected purposes.
3. Compile: compilation makes programming language translated to machine language, and generating debugging info.
4. Preverify: verification of byte code to ensure that the class file is structurally and conceptually correct as the JVM specification.
5. Package: Manifest file, then JAR and JAD file is created in the step.

6. Test: MIDlet is tested by using emulator device on computer for minimizing errors or bugs.
7. Deploy: Installing the MIDlet into a real mobile device and implement it.

2.5 J2ME Development Environment

2.5.1 J2ME WTK

J2ME Wireless Toolkit (J2ME WTK) is the official MIDP application development tool from SUN Company release. It provides a complete development environment of coding and testing for MIDP applications and an emulation environment for a variety of devices. The J2ME WTK allows the use of any text editor for editing source files.

At the present time, J2ME WTK can work on the platforms of Windows Xp, Windows 7, Linux and Solaris. It is free to use, program developer can download the toolkit from Internet.

The version of WTK used in this thesis project is 2.5.2. Since I used the same version of WTK for the last project I took from school's lecture to build a media player, though I am more familiar to WTK 2.5.2.

2.5.2 IDE (Integrated Development Environment)

There are a good deal of IDE tools are well-known, such as Eclipse, Jbuilder and NetBeans. Besides the essential functions of J2ME WTK, they offer various visual development tools moreover. In respect that NetBeans IDE is a tool used for this thesis project, therefore it is necessary to acquaint ourselves with NetBeans IDE.

The NetBeans IDE is a famous integrated development environment available for Windows, Mac, Linux, and Solaris. The NetBeans IDE is an open-source integrated development environment. NetBeans IDE supports development of all Java application types.

The version of Netbeans IDE used for this thesis project is 6.9. And it is free to download and use from the official web site.

3. DESIGN

In the following chapter, the points of view of idea and concept from the author are explained in detail.

3.1 Motivation and background information

Nowadays, the usability of mobile phones is no longer only limited to phone calling. People use mobile phone for SMS sending or other basic daily functions. More and more users take better consideration on the joy of using their phones. Moreover, the applications such as digital camera, music player and Internet browser make mobile phones more powerful and multifunctional. Meanwhile, those functions bring users preferable convenience and enjoyment. Therefore, developing and implementing an educational application based on mobile phone for its users will be a good choice.

Why chooses J2ME as the development platform? J2ME was launched by SUN Company in June 1996, which as a new Java version to provide application development especially for the small resource-constrained consumer electronics devices. Its major technical advantages are that: has a good cross-platform capability to achieve the superiority of which called "write once, run anywhere"; with a seamless integration with J2EE capacity; retains excellent Java language characteristics, such as simple, safe; and the existing Java platform, a wide range of development tools, enterprises, developers can provide a good J2ME support.

Furthermore, Finnish is a minority language for the foreigners and tourists. Nowadays, I realized that the translation tools about Finnish - English is not adequate in today's market, let alone the application that be run on the mobile devices. In addition, technology about mobile devices is rapidly developing, it has a very good development prospects.

3.2 Functionality

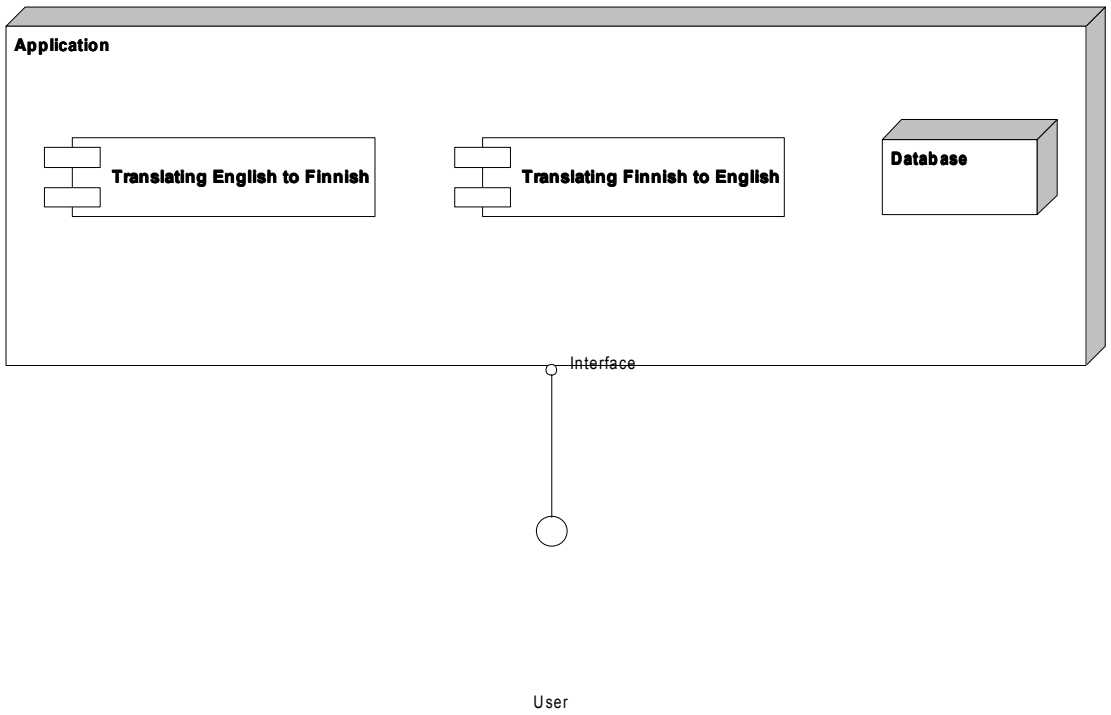


Fig.7. Deployment diagram

The main functionalities of application are shown in the Fig.7. The application was designed as a bidirectional translating tool, it means user can reach:

- Finnish- English Dictionary: Translating a Finnish word meanings in English
- English- Finnish Dictionary: Translating an English word meanings in Finnish

3.3 Analysis of Application Needs

The target application was expected to accomplish following functions:

- Application can let user to input a target word which needs to be translated
- MIDlet is able to retrieve data from existing word-stock files
- MIDlet shows up the translating result on the screen

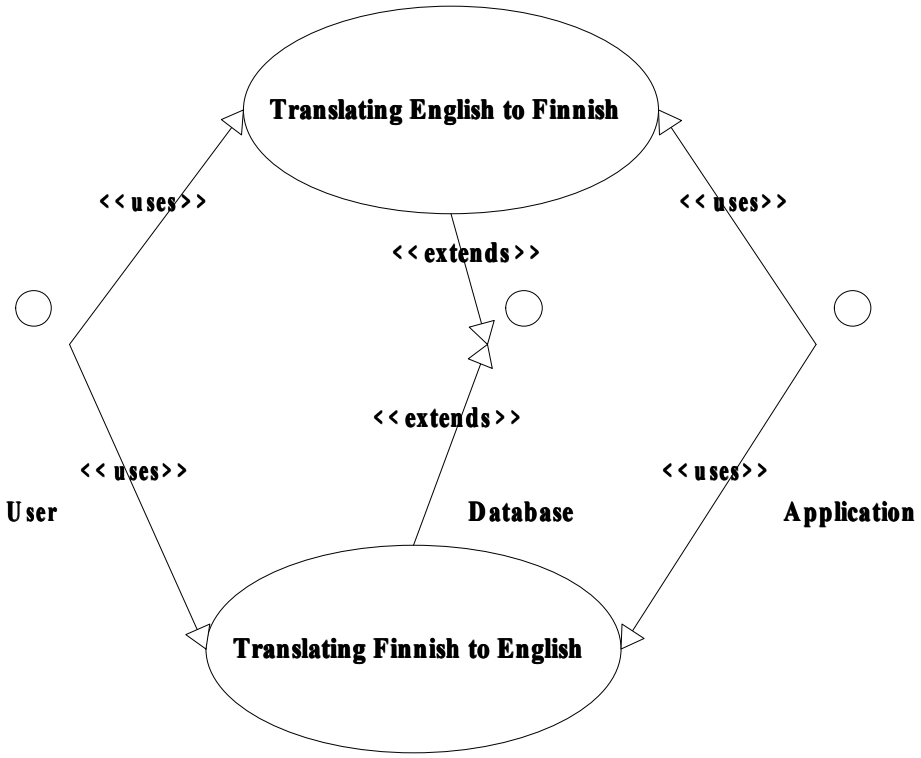


Fig.8. Use Case diagram

Fig.8 describes the relationship between actors and use cases. There are three actors in this diagram, the first one is User, the second one is Application and the third one is Database. In addition, there are two use cases: (a) Translating a Finnish word to English; (b) Translating an English word to Finnish.

3.4 Target Goals

To obtain an application of Finnish- English dictionary that can be implemented on mobile phones those support Java, therefore the application was designed to have these factors:

- Usability: Easy to install and fast to get familiar with.
- Reliable reusability: Make sure application is durable.
- Fine expansibility: Support more types of language for the future work.
- Testability: For in case of bugs happen.
- High performance: Cost memory as low as possible.

3.5 Considerations During in Design

The product of mobile equipments and the technology related to it is developing

rapidly everyday, it is impractical to build applications or supply services for a number of specific models of mobile phone in a small range. In case of doing for that, it will only raise the cost of developing and maintenance. So developing programs on the extensive and universal platform is more reasonable and serviceable. iPhone is just an good example, it has an incredible selling achievement in the market; and behind this there are a huge number of devotee following iPhone. Why? The answer is obvious, the advantageous iPhone OS makes numerous outstanding developers take part in it and they help iPhone OS and Apps to become more excellent. That is why nowadays more and more people are using or going to use iPhone.

J2ME also provides developers of an extensive and universal platform; nevertheless, there are still some factors to be considered.

- Limited processing ability of mobile phone

Nowadays, most of mobile phones support J2ME, but they can only supply very limited processing ability. The hardware of mobile phones mostly are small, this is a big barrier to applications development. Because EFDictionary has a big deal with database process, so making an efficient algorithm is much needed.

- Finite screen size of mobile phone

The screen on mobile phone is quite small compare to computer monitor in despite of the manufactures are making mobile phones' screen bigger and bigger. And there is no such a "standard size" to be defined, which means different models of phone have different sizes of screen. This restriction brings developers more work to do, and in the meantime it reduces the portability. Actually, I only used Low-Level API to develop EFDictionary, and the application works well on the emulator with 480x360 screen resolution.

- Small memory configuration

It needs to consume a lot of RAM for running a program on the mobile phone, but most RAM inside mobile phone is quite small at the same time. Besides some high-level smart phones with powerful configurations, the RAM inside of most ordinary mobile phones is less than 64M. It is bottleneck restrictions to applications development for mobile phones. As the way of solving problem of limited processing ability, here the efficient algorithm could figure it out.

- Portability

With regard to different configurations from various models of mobile phones, portability should be taken good care of. Such as protocols supporting, screen size. On account of J2ME technique is from Java, so thanks to super excellent characteristics from Java and J2ME, it has good portability as its nature.

3.6 System Design

3.6.1 Activity Diagram

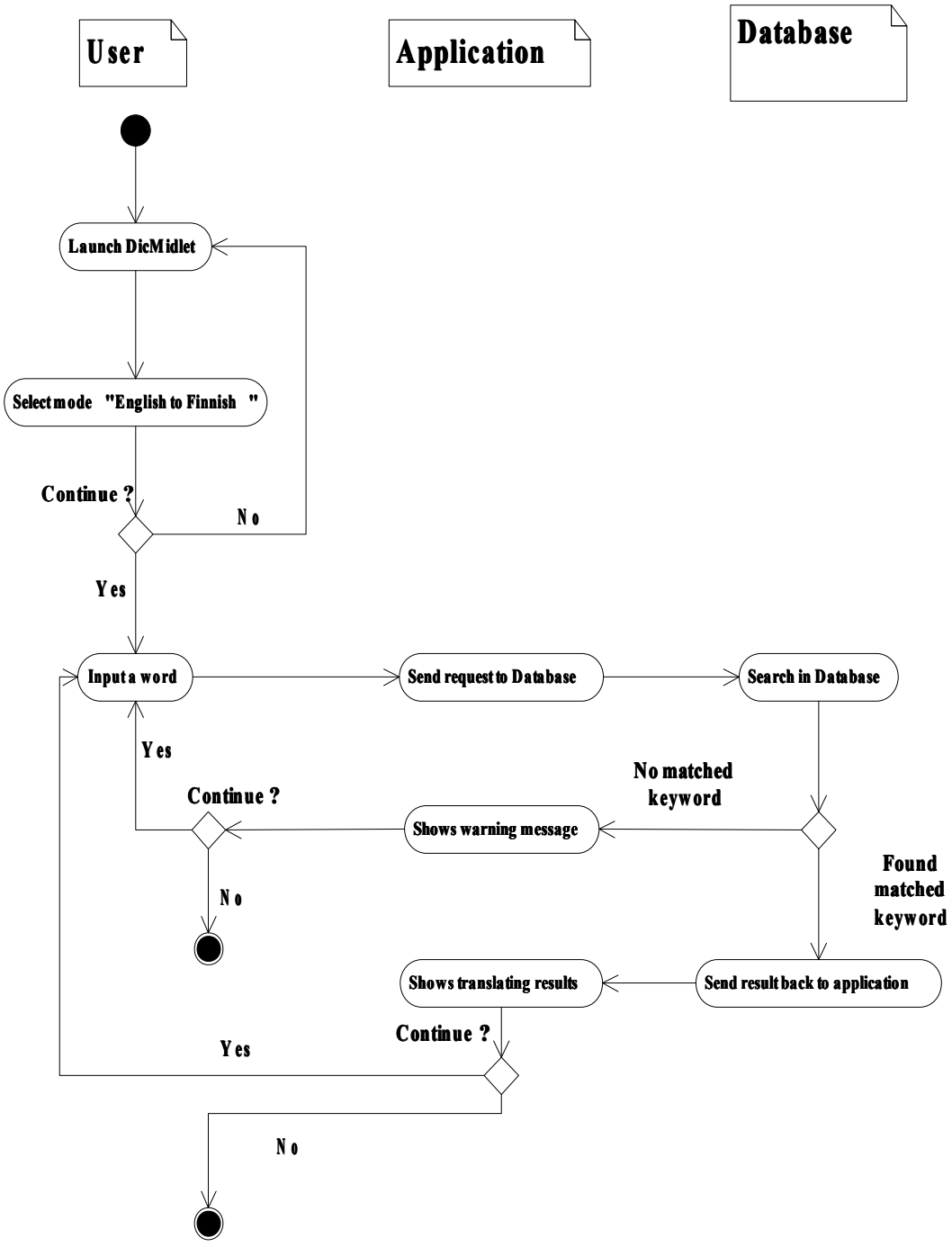


Fig.9. Activity diagram

The Fig.9 is drawn to describe the whole process of application work. Firstly, user can launch EFDictionary on his/her phone. There are two functions to be selected: (a) English to Finnish; (b) Finnish to English. According to user's choice, for instance, the

function of "English to Finnish" is selected. Secondly, user is asked to input a word. Having been inputted, word is stored into memory; the word is sent to database by application and searched inside it. There is a judgement by program to determine what will be done on next step. If there is matched keyword in database, this word will be sent back to application and displayed on screen. Finally, user is asked whether he/she wants to finish or start a new process. If user chooses "no" the application will be finished. Otherwise the application will be back to second step for asking a word. On the other hand, if there is not matched word in database, a warning message will be showed on the screen.

3.6.2 Flow Chart

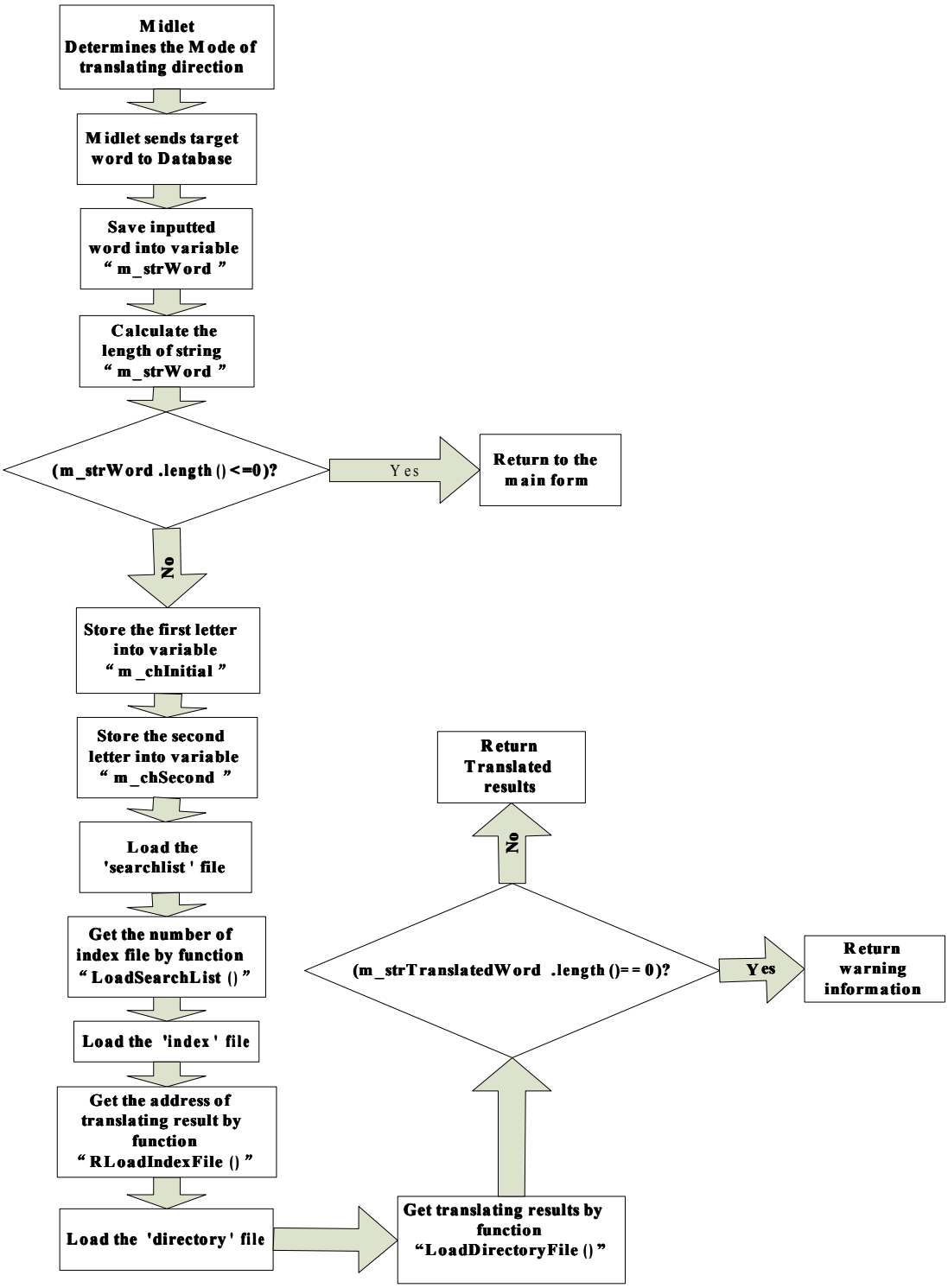


Fig.10. Flow Chart

Fig.10 expresses how the EFDictionary processes with algorithms and relates to database.

There are several key functions which are made for the EFDictionary. And now I am exampling some key functions for my thesis readers to ease understood. Once this MIDlet has been launched, the first decision from user is to select the translating direction from either "English to Finnish" or "Finnish to English". The function I made in MIDlet for detecting the translating directions is by the index position on the listing form which is named "operationList" in codes.

Secondly, there is a user interface with a textbox let users inputting target translating words. In case the textbox is still empty after user click the "OK" button, then the programme will keep staying at the same user interface. Because I made another function to test the length of the content from that text button, after the action of "OK" button actives. The purpose of this function is to avoid unwanted errors happened and to reduce the computation

The most important part in the EFDictionary is the process of trying to find the most matching words for the target translating given by user. Actually there are three steps to be dealt with in this section. More detailed descriptions are given in the section 4.3, since it has more relation with database of words stock. It will become much easier understand after me explaining the structure, contents, and algorithms in database.

Afterwards, on the condition that there are matching words found in the words stock, the step of showing translating results should be followed. I also created a method to detect whether there is translating results found. This function also has the principle with the length calculation method like I mentioned another familiar function for the text box of target translating word inputting. If it was succeed in finding matching target words and its translating results, then shows the results. On the contrary, there will be a warning message shows up if there is nothing found after process in database.

3.6.3 Class Diagram

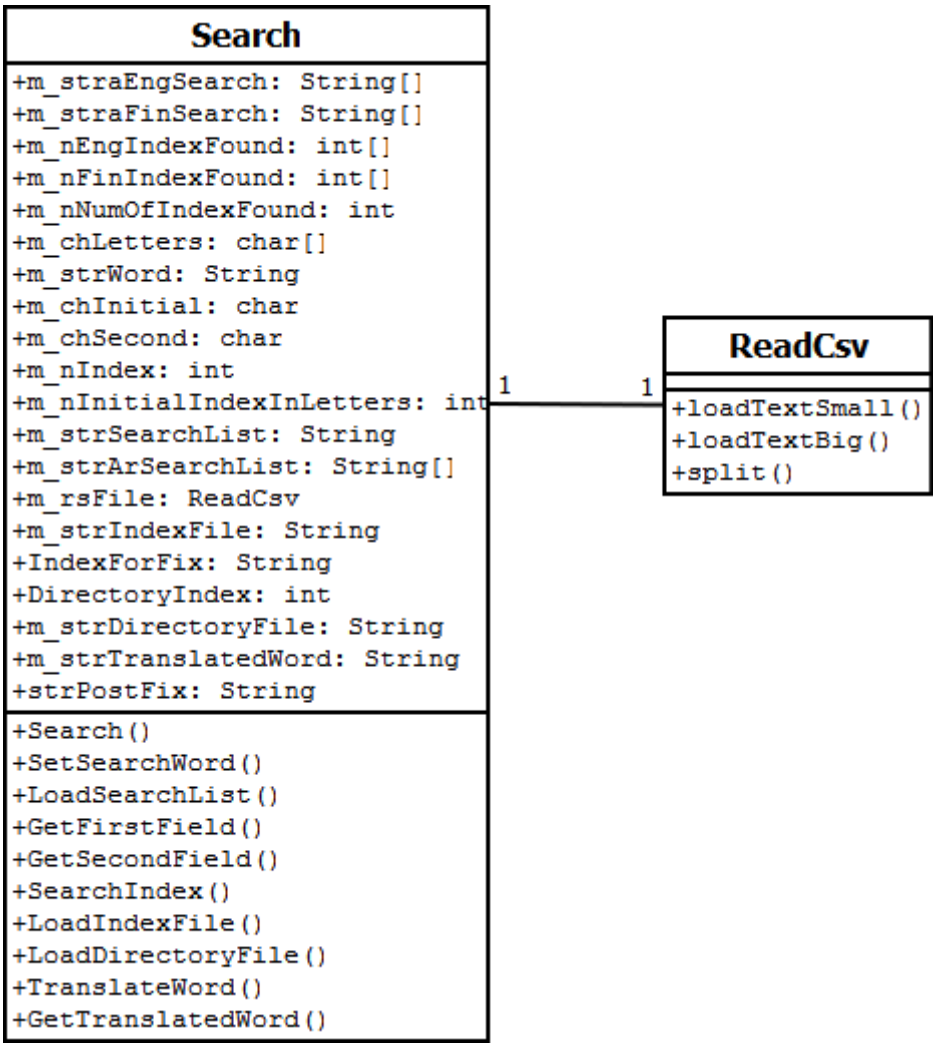


Fig.11. Class Diagram

There are two classes in the Fig.11, which are "Search" and "ReadCsv".

In class of "Search", there are 20 attributes such as m_straEngSearch, m_straFinSearch, m_nEngIndexFound and others. It also has 10 methods: (1) Search(), (2) SetSearchWord(), (3) LoadSearchList(), (4) GetFirstField(), (5) GetSecondField(), (6) SearchIndex(), (7) LoadIndexFile(), (8) LoadDirectoryFile(), (9) TranslateWord(), (10) GetTranslatedWord() respectively.
In "ReadCsv" class, "loadTextSmall()", "loadTextBig()" and "split()" are the only 3 methods. More detailed description about attributes and methods are expressed as comments in the code.

The relationship between Search and ReadCsv is 1 to 1 which means they are unique to each other at one time.

3.7 Application Running Platform

The application can be run on any mobile phones which support Java and specific emulator device. The MIDlet was developed according to J2ME Profiles and Configurations, which can avoid the problem from different characteristics of mobile hardware from various manufactures. Moreover, it has high performance of portability on any mobile systems support Java.

4. IMPLEMENTATION

A brief explanation of methodologies adopted during the process of application development.

4.1 GUI (Graphic User Interface)

A core feature of the MIDP (Mobile Information Device Profile) technology is its support for developing mobile phone user interfaces. The MIDP provides a set of Java APIs known as the LCDUI, which has functionalities being similar to the Java Abstract Windows Toolkit (AWT) and Swing APIs in the desktop world. J2ME supports mobile phone UI development using the MIDP APIs.

4.1.1 User Interface Architecture

MIDP contains user interface classes in javax.microedition.lcdui and javax.microedition.lcdui.game packages. The MIDlet is represented by an instance of Display class, accessed from a method called getDisplay(). The main purpose of Display is to keep track of what is currently shown, which is an instance of Displayable. MIDlets can change the contents of the display by passing Displayable instances to Display's setCurrent() method.

The MIDP user interface classes, found in the library javax.microedition.lcdui, can be considered on two parts: the high-level UI API and the low-level UI API.

The high-level part of UI API gives the maximum portability for the application. The high-level classes are heavily abstracted to provide minimal control over their look and feel, which is left for device on which they are deployed to manage, according to its capabilities.

The low-level part of UI API provides very little protection from the physical device's capabilities. The classes of the low-level group are perfect for MIDlets where precise control over the location and display of the UI elements is important and required. Of course, with more control comes less portability.

For you to be able to show a UI element on a device screen, whether high-level or low-level, it must implement the Displayable interface. A displayable class may have a title, a ticker, and certain commands associated with it, among other things. This implies that both the Screen and Canvas classes and their subclasses implement this

interface. A Displayable class is a UI element that can be shown on the device's screen while the Display class abstracts the display functions of an actual device's screen and makes them available to you. It provides methods to gain information about the screen and to show or change the current UI element that you want displayed. Thus, a MIDlet shows a Displayable UI element on a Display using the setCurrent(Displayable element) method of the Display class.

As the method name suggests, the Display can have only one Displayable element at one time, which becomes the current element on display. The current element that is being displayed can be accessed using the method getCurrent(), which returns an instance of a Displayable element. The static method getDisplay(MIDlet midlet) returns the current display instance associated with your MIDlet method. Fig.12 describes the architecture of user interface in J2ME development.

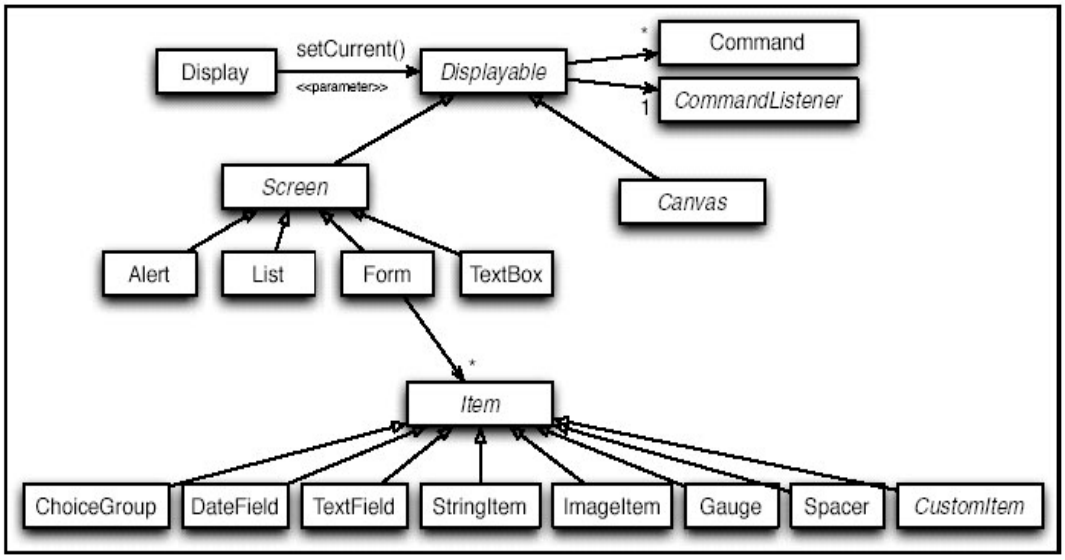


Fig.12. User Interface Architecture /13/

Because my target application is a dictionary tool, so the user interface should be following the objectives of simply and efficient. There are only few Displayable elements highly used, which are List, Form, TextBox, TextField and Button as well.

4.2 Database Structure

As the result after discussing with my thesis instructor, the database I used for my dictionary is an existing one from Internet. The source is downloaded from http://www.dicts.info/java/en/download.php?file=english-finnish and it is free. Actually there are many existing applications developed with lots of languages on that website, but here I only download and use the words stock files as the database for my own application. The file I downloaded is called "dfm-english-finnish.zip" from the

reference link above.

After downloaded and extracted, it is a folder file which contains lots of files under it. There is a folder named" dictionary" in the source file folder. In fact, this"dictionary" folder plays the role of words stock. Actually this folder of words stock was an existing one from another program which is open and free on the Internet, rather than I created it. So before I used this word stock for my own program, I must to figure out the algorithm and principle to use it. Fig.13 shows the contents under folder of Dictionary.

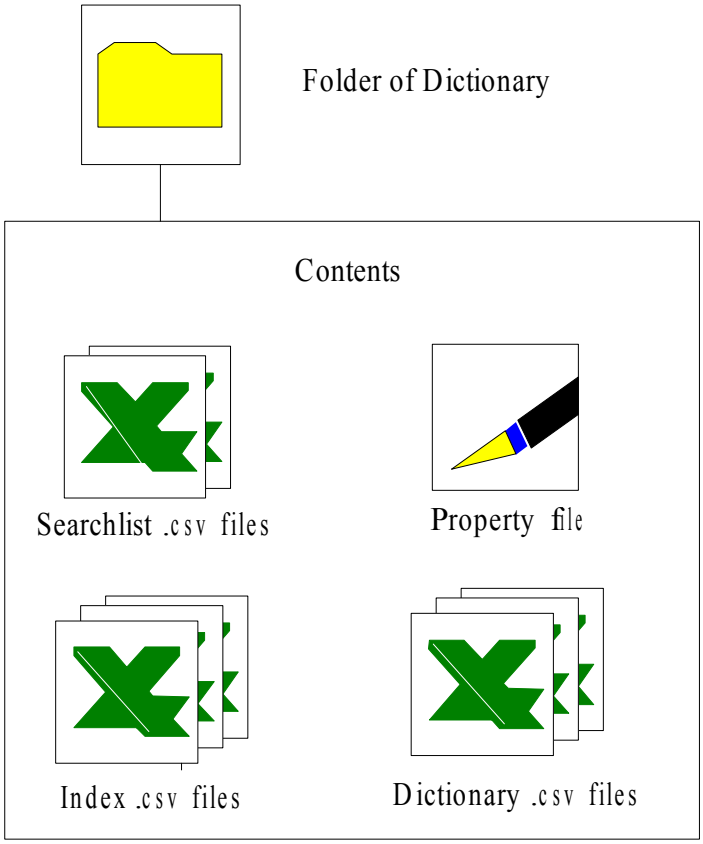


Fig.13. Folder Structure of Database

As there are many files under the "dictionary" folder from the source file, I made a statistics. There are 4 types of file could be considered.

I opened those files with EXCEL software, which could display the original file structure.

Firstly, the files with initial name "searchlist"; there are only two. searchlistfin.csv and searchlisteng.csv. There is only one column which consists of one word and one number. For example "week 39" in this case, number 39 shows that the words after word "week" ordered by lexicographic order are listed in indexeng39.csv. Another example, the keywords "fine14" and "frothy15" also exist in the searchlisteng.csv.

So we can find word ""fine" from indexeng14.csv and the words like "first", "flat" should be found in the same indexeng14.csv file since the lexicographic order. Another reason for these two words existing in the indexeng14.csv rather than indexeng15.csv is because their lexicographic orders are in front of the word "froth". This illustrate that any word ordered between the range of "fine14" and "frothy15" with lexicographic order, which should be found in the file indexeng14.csv.

Secondly, files about index. There are 39 files of indexeng.csv and 43 ones of indexfin.csv. Same example about word "fine", the first row in the file "indexeng14.csv" is for word "fine". There are several columns that each with special numbers and letter, several columns for several translating meaning result according to some words in language have multiple meanings. Like "96-1491-B" which at the first column for word "fine". I figured out that number 96 leads to the file"dictionary96.csv" and code "1491-B" stands for the position of the character begins of the word in the file of "dictionary96.csv".

Thirdly, the files called directory.csv have 288 in all. In the file "dictionary96.csv", I found some combinations like finehyvin, fineihana, finekaunis. So those combinations have a common point, which starts with "fine"; the rest part behinds part "fine" like hyvin, ihana, and kaunis which I think they are the meanings in Finnish.

Finally, DictionaryForMIDs.properties file is the property file.

4.3 Algorithms of Searching Target Words

After understanding well at the structures of database, I moved forward to the work of the algorithms between database and application.

As I have mentioned the algorithms ago in the section 3.6.2 about the flow chart, the EFDictionary application would get a target translating word once users active the event of translating. If the value of the inputted word is not empty, then the MIDlet should to do tasks of searching that word in the database of words stock files and gathering the translating results.

Since there are two ways of translating direction and each way of them have their own words stock files in database, so I thought it would be much efficient to make a sorting job in the beginning. By the reason of the MIDlet has the function of knowing which translating directions user chose from the list at the first interface which shows up on the screen.

Then I tried to assign a value of either 'eng' or 'fin' to a String 'strPostFix' that is a part of content of the path for looking for the file searching the target word.

After that, building functions to find the matching words based on the target word. According to the natural structure of database, there are three steps to be considered. As it was described at the last section, MIDlet firstly process the target word with 'searchlistfin.csv' and 'searchlisteng.csv' depends on the value from String 'strPostFix' which expressed the translating direction. Here I assume translating an English word "first" to Finnish as an example; processing 'first' with in 'searchlisteng.csv' is the first step to get the final result. I realized that the purpose of this step is to get locating of the target word for the next step, and there should be an advanced function to implement my goals. I was thinking what's connection and relationship between the target word and the words in the file 'searchlisteng.csv', somehow any common point they have? Eventually I figured out that I could make comparison between them since the words in word stocks are listed by lexicographic orders.

The following piece of code is given to describe the method of split():

```
public static String[] split(String original, String regex) {
        int startIndex = 0;
        java.util.Vector v = new java.util.Vector();
        String[] str = null;
        int index = 0;
        startIndex = original.indexOf(regex);
        while (startIndex < original.length() && startIndex != -1) {
            String temp = original.substring(index, 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;
    }
```

List.1. Part of Original Source Code /16/

A part of source code is quoted in List.1 as above, it illustrate how the idea is. Now I explain more, the target word 'first' as an original word in codes meanwhile words from stock are as regular expressions. Regular expressions are used when you want to search for specify lines of text containing a particular pattern. A simple example for regular expression here: The simplest character set is a character. The regular expression "the" contains three character sets: "t," "h" and "e". It will match any line with the string "the" inside it. But the original word is compared to only one regular

expression at every turn. The sequence is begins compare to the words from the top down ordered by lexicography. Firstly it compares the initial letter from both two words: the target word 'first' and a regular expression which is a random word from the words stock in this case here, it continues to compare the second letter with the same word if they match or jump to the next word to compare their initial letter again. After the most familiar regular expression was found, they take the value of the number behind that word. That number leads to the next step. In this particular example of target word 'first', the number is 14 which illustrate the result is in the file "indexeng14.csv" for the next step.

The task of next step is familiar as the first one by using the same method of algorithms. But in this step, it makes more calculations and comparisons with 'Index' files in database for a more precision searching since there are much more words to be compared with than the last step. As the result got from the prior step, the Index file here is "indexeng14.csv". The expected goal of this step is to find the matching word exactly. After then, take the index numbers behind the matching words for the final step which obtains the final translating results. In this case, the index numbers behind the target word 'first' are as following: 97-368-B, 97-382-B, 97-397-B, 97-408-B, 97-422-B, 97-441-B. Six results are gained for this example, which means there are possibly 6 multi-meanings existing.

Finally, the MIDlet moves forward to achieve the final translating results in 'directory' files by the index number given from the last step. By the index given, it is easy now to locate where translating results are. The first translating result of target word 'first' here is begins at the 368th position in bytes from file "directory97.csv" due to the combination "97-368-B" is the first index number gained from the prior step. But the issue of multi-meaning should be considered here since one word could be translated to more than one result. The 'directory' files of EFDictionary have a feature that is the translating result is always behind the original target word, any string in there are always made of an English and a Finnish word. So by the same method of split () that was quoted above, finding the matching words first, then take the behind part which is the final translating results.

After obtaining the results, they are shown on the screen if the result was found. Or else, a warning message will be displayed if there was no matching result found. Then user will be given a choice either inputting another target word to translate or exiting this application.

5. EVALUATION

Right now, the outcome of application has been done and working well. But it does not reach all the goals as I expected in the beginning of thesis starts. The application was expected to have the function of showing up matching vocabularies on the screen while user typing initial letters, but I have not done yet because of the limitation of techniques and time.

Anyway, there are both advantages and disadvantages of my application to be shared with you. Since the EFDictionary was developed with the J2ME, so it has good compatibility for the mobile devices from different manufactures. But as a work done from a bachelor student, I think there are lots of imperfections in my work. Such as mobile phones with touch screen have no chance to use my application at this moment, and the quantity of the words stock is limited and so on. During the whole process of thesis, I realized that my programming skill is very amateur, which is a major reason to restrict the level of my product.

There is some future work I am planning to do for improving this application. First, a job of supporting retrieving data from external file that stored in the memory card of mobile phone could be considered, which can expand my application with more target languages. Secondly, trying to work with touch screen phone is expected. Because the touch screen is the trend of development of mobile phone. Furthermore, I should upload my source file onto Internet for sharing with people who are interested, then it will be much easier for them to get it since more and more mobile phones has ability to access Internet.

6. CONCLUSIONS

The research of this thesis illustrated a new solution for dictionary based on mobile phone. This report showed the process of the research, and also features of EFDictionary.

During this thesis work I learned a lot. Most of time I spent on with J2ME programming language and MIDlet development. The highlight of the research is the EFDictionary application which was implemented in the mobile phone. Furthermore, the result was proved successful in theory.

However, there are still some weakness of the application that could be improved in the future work. There are few betterment points in my mind.

1. The matching vocabularies could be shown up on the screen while user typing initial letters
2. EFDictionary works well in the mobile phones with touch screen.

To sum up, the research has been recorded in documentation as an ideal plan, but it could not present whether the principal could be totally accomplished in the real work. At the same time, when the thesis was processed step by step, more and more challenges would come out. At the moment, I realized that it is a good sense for me to study and improve myslef skills.

REFERENCES

Listing the sources:

/1/ Sing Li and Jonathan Knudsen, Beginning J2ME™ platform From Novice to Professional, THIRD EDITION, Apress, 2005

/2/ Sams Publishing, Sams Teach Yourself Java 2 in 21 Days, Second Edition, Sams Publishing, 2000

/3/ Ronald Ashri, Steve Atkinson, Danny Ayers, Marten Haglind, Bill Ray, Rob Machin, Nadia Nashi, Richard Taylor, Chanoch Wiggers, Professional Java Mobile Programming, Wrox Press LTD.®, 2001

/4/ Cay S.Horstmann & Gary Cornell, Core Java 2 Volume I – Fundamentals, The SUN Microsystems press, 19999

/5/ Anyang-window.com.cn, J2ME SERIES OF THEMATIC DEVELOPMENT ENTRY: J2ME OUTLINED, [WWW-document], < http://www.anyang-window.com.cn/j2me-series-of-thematic-development-entry-j2me-outlined/>, 10.09.2010

/6/ Qusay H. Mahmoud, J2ME for Home Appliances and Consumer Electronic Devices, [WWW-document], < http://developers.sun.com/mobility/configurations/articles/cdc/>, January 2003

/7/ Calsoftlabs.com, Java Programming for Wireless devices using J2ME/CLDC/MIDP, [WWW-document], <http://www.calsoftlabs.com/whitepapers/wireless-device-programming.html>, 10.09.2010

/8/ Javanb,com, Packaging MIDlets - Wireless Java Developing with J2ME, Second Edition, [WWW-document], < http://book.javanb.com/wireless-java-developing-with-j2me/LiB0027.html>, 10.09.2010

/9/ Javanb.com, The MIDlet Life Cycle, [WWW-document], < http://book.javanb.com/wireless-java-developing-with-j2me/LiB0024.html>,

10.09.2010

/10/ Chetan Manchanda, J2M2 APPLICATIONS ON MOBILE DEVICES, [WWW-document], < http://www.wiziq.com/tutorial/11610-J2ME>, 10.09.2010

/11/ Wikipedia, MIDP, [WWW-document], <http://en.wikipedia.org/wiki/Midp>, 10.09.2010

/12/ Vikram Goyal, User interfaces with MIDP 2.0, [WWW-document], <http://today.java.net/pub/a/today/2005/05/03/midletUI.html>, 17.10.2010

/13/ Michael Juntao Yuan and Kevin Sharp, Develop rich mobile phone user interfaces using the MIDP UI, [WWW-document], <http://www.javaworld.com/javaworld/jw-05-2005/jw-0516-midp.html?page=1#resources>, 16.10.2010

/14/ Urs Steiner, J2ME: Introduction, Configurations and Profiles, [WWW-document], <http://www.ifi.uzh.ch/~riedl/lectures/Java2001-j2me.pdf/>, 11.11.2010

/15/ WAVE report, QUALCOMM BREW Tutorial, [WWW-document], <http://www.wave-report.com/tutorials/brew.htm>, 20.11.2010

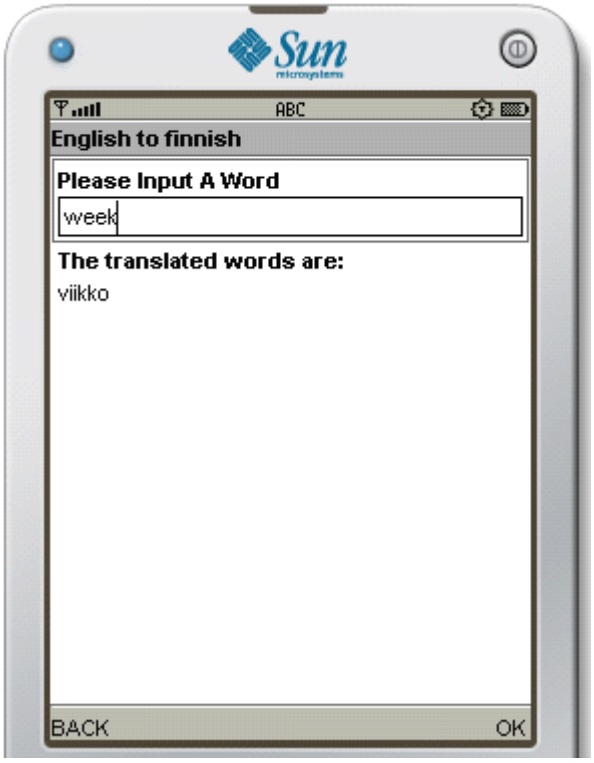
/16/ Softcov, J2ME is implemented in the program split the string, [WWW-document], <http://www.softcov.com/programming-and-testing/j2me-is-implemented-in-the-program-split-the.html>, 22.11.2010

Appendix

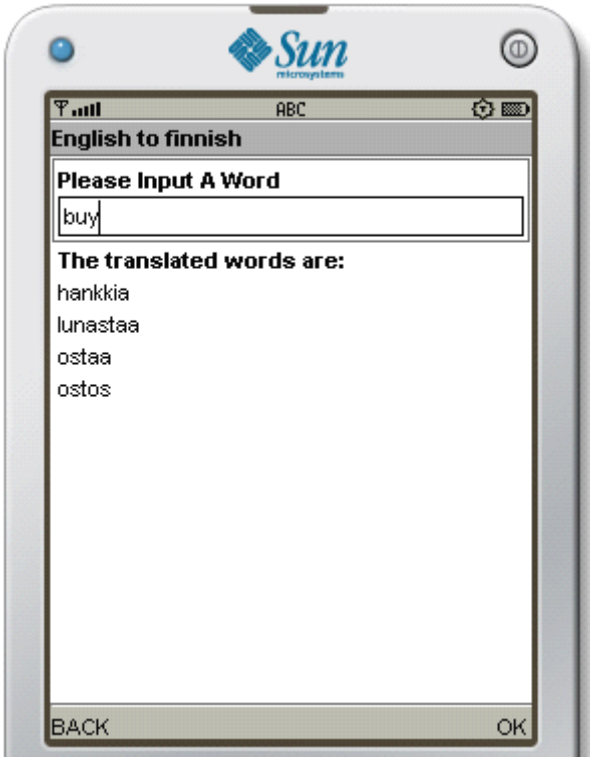
Guide in practical use

Step 1.The translating-mode interface. Select the translating mode (direction) English to Finnish in this example. Click "OK" button to implement, "Exit" button to terminate.

Step 2.Input the target word in the textbox, click "OK" button to implement and shows translating results in the blank space, "BACK" button to return the translating-mode interface.



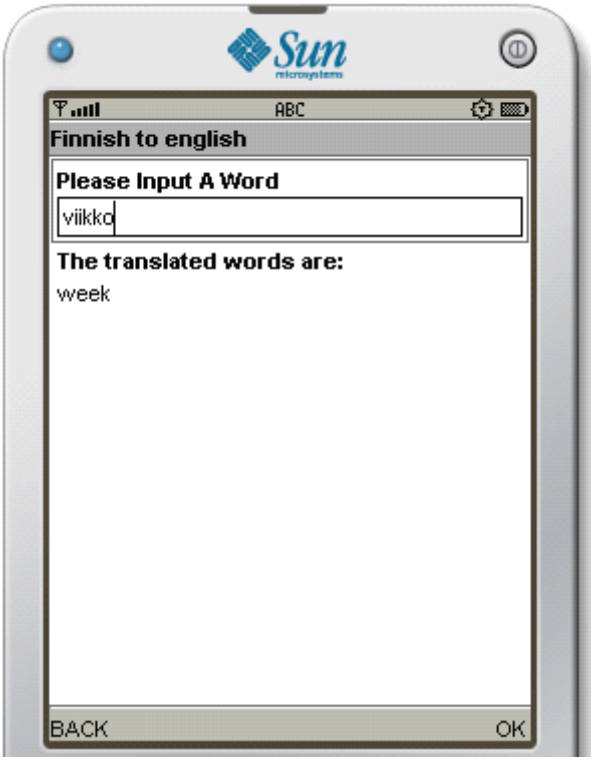
Step 3. Continue to implementing the next target words for translation. Just repeat the same operation as Step 2.



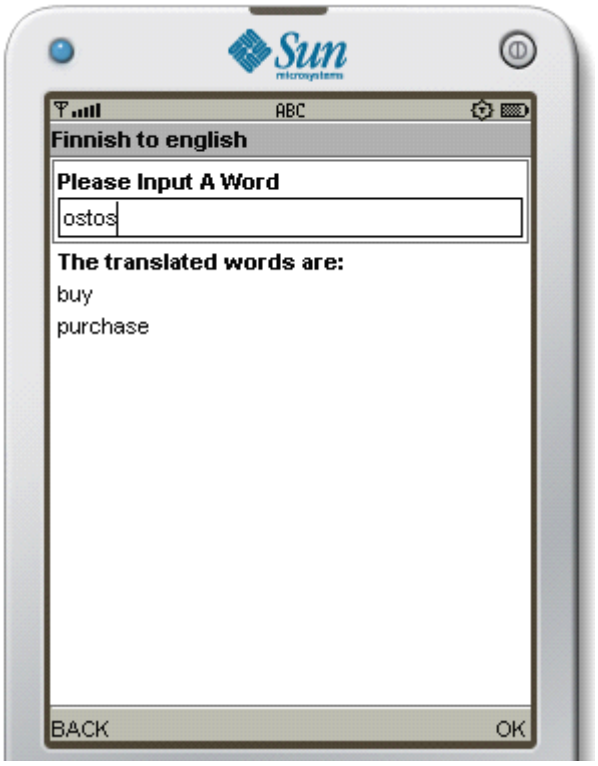
For translation from Finnish to English, the operations are same after implementing the translating- mode interface.

Step 1. Translate Finnish to English

Step 2. Input the target word of "viikko" and shows the translating result "week".



Step 3. The translating result of Finnish word "ostos".


Sun
microsystems
ABC
Finnish to english
Please Input A Word
ostos
The translated words are:
buy
purchase
BACK
OK