Wireless Networks Detection in Mobile Phone Using J2ME Language

Ansam Emad Naji & Sabreen Emad Naji
Department of Mathematics, College of Education for Pure Sciences, Babylon University, Babylon Province, Iraq.
Email : ansamemad@yahoo.com

Abstract

Using J2ME language in NOKIA mobile cellular phones, we detects internet access points names by IAPInfo.

IAPInfo (Internet Access Point Information) is a simple API Java Micro Edition, which is a Nokia proprietary package, which is used by applications to access related information about fundamentals of Internet access points and destination networks. This information is retrieved from the native OS, where all the communication related settings are stored.

This project concerned with detection an Internet access points names and listed in a menu, by using the J2ME language which is the suitable programming languages that deal with mobile phone applications, and it provides convenient environment for these devices.

Finally, the project yields satisfied results when it is tested practically using mobile NOKIA (N78) .

Introduction

1.1 General Introduction

Mobile and wireless computing has the power to change the way business is conducted. It allows employees, partners, and customers to access corporate data from almost anywhere. Universal data access, combined with increased worker productivity and effectiveness, reduced processing costs, heightened accuracy, and competitive advantages, is driving the demand for enterprise mobile applications. As the demand continues to increase, the mobile infrastructure that makes creating sophisticated mobile applications possible is maturing .

1.1.1 Mobile Phones

phones get more and more popular. Since August 2006,more mobile

phones are registered . As the capabilities of these devices increase, they are not simple voice centric handsets anymore; rather they provide mobile computing power that can be used for several purposes .

1.1.2 Java 2 Micro Edition

Java 2 Micro Edition (J2ME)has been developed primarily as a technology for the execution of applications on constrained devices. In this case, the constrained devices are mobile phones, PDAs, TV set-top boxes, in-vehicle telemetry, residential gateways and other embedded devices .

1.1.3 Wireless Networks

wireless technologies are all around us. These technologies have pervaded our homes, businesses, and lives. Life without some of these technologies is almost unimaginable . The wireless communications industry has seen rapid growth in the past several years. This has made wireless communication one of the fastest growing technology areas in the world .

1.1.4 J2ME

Java 2 Platform, Micro Edition (J2ME) is the second revolution in Java's short history. When Java was introduced in 1995, it looked like the future of computing was in *applets*, to understand how Java 2 Micro Edition (J2ME) lies within the wider Java landscape it is best to explore the overall Java architecture.J2ME has been developed primarily as a technology for the execution of applications on constrained devices. In this case, constrained devices are mobile phones, PDAs, TV set-top boxes, in-vehicle telemetry, residential gateways and other embedded devices. J2ME as a whole can be described as the technology that caters for all these devices. Given that many of them have limited resources, it would be imprudent to expect all of these devices to be able to deliver all of the functionality of the few. The Java community therefore decided that these devices should be grouped to best reflect their purpose and capabilities .

J2ME is the newest and smallest addition to the Java family. It is the smaller brother of J2SE (Standard Edition) and the server-based J2EE (Enterprise Edition) . It also describes the packages and classes within the commonly used environments, with special emphasis on MIDP 2.0 .

1.2 Project Objective:

The goal of this project is to detection the available Wireless Networks Access Points using J2ME as a programming language, over Symbian OS of cellular phone platforms.

1.3 Scope and Limitations:

- Scope: The application can used by the internet users on mobile phone.
- Limitation: the application can be used only on mobile phone that support the wireless .

Mobile and Java 2 Micro Edition

2.1 The Mobile and Wireless Definitions

The mobile and wireless definition varies from person to person and from organization to organization. In many cases, the terms mobile and the wireless are used interchangeably, even though they are two different things. Mobile is the ability to be on the move. A mobile device is anything that can be used on the move, ranging from laptops to mobile phones. As long as location is not fixed, it is considered mobile. Wireless refers to the transmission of voice and data over radio waves. It allows workers to communicate with enterprise data without requiring a physical connection to the network. Wireless devices include anything that uses a wireless network to either send or receive data. The wireless network itself can be accessed from mobile workers, as well as in fixed locations. Figure (2.1) depicts the relationship between mobile and wireless area. As can be seen in the most cases, wireless is a subset of mobile; but in many cases, an application can be mobile without being wireless .

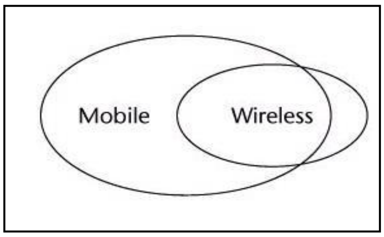


Figure (2.1) The Relationship between the mobile and the wireless

For an application to be considered a mobile or a wireless, it must be adapted to the characteristics of the device that it runs on. Limited resources, low network bandwidth, and intermittent connectivity all these are the factors into the proper design of these applications .

On the other side, there are many examples where the Mobile applications are not wireless. There are many examples where this is the case. Any application that can be used on the move and that does not have wireless connectivity fits into this category. This includes many laptop and personal digital assistant (PDA) applications. Until only a few years ago, it was actually rare to have wireless data access for mobile devices. For these mobile applications, data is often synchronized using a fixed connection and stored on the device for use at a later time. It is worthwhile to note that even though these applications do not require wireless connectivity, they can often benefit from it when it is available .

The use of wireless devices such as cellular phones and two-way pagers has undergone tremendous growth over the past few years. Sun and a group of wireless industry leaders such as Motorola, Nokia, and Palm defined a new set of standards called Java 2 Micro Edition (J2ME) to help develop and deploy the next generation wireless applications that can change the way people live and the way they do business .

2.2 Symbian Operating System

The Symbian have been developed by the Symbian software company for mobile phones. First Symbian OS mobile phone, Nokia Communicator 9210, became available in the first half of 2001 and since then there have been several others from several different manufacturers, for example Sony Ericsson P800, Fujitsu F2102V, Nokia 6600, Siemens SX1, and Sendo X .

Symbian operating system is an open operating system that is specifically designed for data-enabled Second Generation (2G), Second and half Generation (2.5G) and third Generation (3G) mobile phones. It includes multi-tasking kernel, communications protocols, integrated telephony support, advanced graphics support, low-level graphical user interface framework, etc .

Symbian OS has been designed to support many different kinds of mobile phones varying from voice-centric phones to information-centric phones. Symbian vision is to build an operating system for mobile phones that are not only phones but more like personal digital assistants (PDAs) that come closer to a computer than to a traditional mobile phone.The Symbian operating system is designed specifically for mobile phones rather than scaling down a PC operating system or building communication capabilities over a small basic operating system found on many PDAs. Fitting an existing operating system to mobile phone needs would induce so many fundamental compromises that it would be impossible to build a robust and scalable operating system on that foundation. Therefore Symbian OS is built from scratch and designed to be an efficient and robust operating system on various different mobile phones .

2.2.1 The Series 60 Platform

Series 60 Platform builds on the Operating System from Symbian, complementing it with a configurable graphical user interface library and a comprehensive suite of applications plus other general-purpose engines. Series 60 is a complete smartphone reference design.

2.3 Understanding J2ME

J2ME isn’t a specific piece of software or specification. All it means is Java for small devices. Small devices range in size from pagers, mobile phones, and personal digital assistants (PDAs) all the way up to things like set-top boxes that are just shy of being desktop PCs. J2ME is divided into *configurations*, *profiles*, and *optional APIs*, which provide specific information about APIs and different families of devices. A configuration is designed for a specific kind of device based on memory constraints and processor power. It specifies a Java Virtual Machine (JVM) that can be easily ported to devices supporting the configuration. It also specifies a strict subset of the Java 2 Platform, Standard Edition (J2SE) APIs that will be used on the platform, as well as additional APIs that may be necessary. Device manufacturers are responsible for porting a specific configuration to their devices .

Profiles are more specific than configurations. A profile is based on a configuration and provides additional APIs, such as user interface, persistent storage, and whatever else is necessary to develop running applications for the device .

2.3.1 MIDlets and MIDlets Suite

Java applications that run on the MIDP are known as *MIDlets*, and in some ways resemble the J2SE concept of Applets. Like Applets, MIDlets must extend the MIDP-defined abstract class javax.microedition.MIDlet and provide a public default constructor (i.e. a constructor that requires no arguments), which enables the system software to create an instance of MIDlet. MIDlets run in an execution environment within the Java VM that provides a well-defined lifecycle controlled via MIDlet class methods, which each MIDlet must implement.

A collection of one or more MIDlets is packaged together into one JAR (Java archive) file to form a *MIDlet suite*. All the MIDlets in a suite can be installed, uninstalled and removed only as a group. The MIDlets in a suite share both static and runtime resources of their host environment, e.g. classes loaded into their mutual Java VM and persistent storage; in the MIDP 2.0 though, inter-suite access is allowed as for the persistent storage, if explicit permission is given .

2.3.2 MIDlet Lifecycle

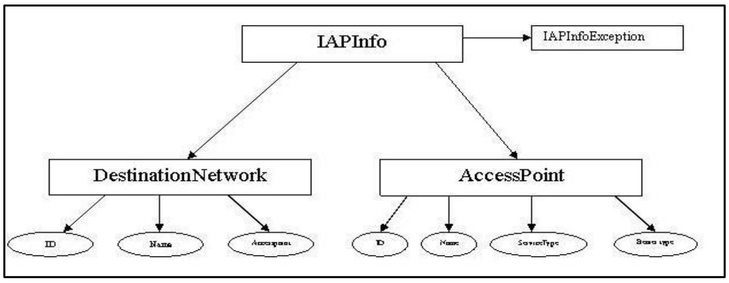
MIDlet has one more analogy to Applet: like the Applet's start, stop and destroy methods, the MIDlet class defines three abstract methods that the system software calls to start, pause, and destroy an application – namely startApp, pauseApp, and destroyApp.

Especially in the case of mobile phone, we must assume that the application would be often interrupted, among other things, by incoming or outgoing phone calls. Therefore it is essential to implement the MIDlet's pauseApp() method, to save important data and free up as much memory and other resources as possible before going into the paused state.

The destroyed state indicates that a MIDlet has terminated and resource associated with it can be freed by the system. Once in the destroyed state, the MIDlet cannot transit back to the other state. Again, developers are to implement the destroyApp method in a proper way so that the important data are saved, all the allocated resources are released, any background threads are terminated and any active timers are stopped as well as all the connections are closed. In fact, finalizes do not exist in the CLDC-based profiles, so the only way to free the resource used by an object is to do it explicitly .

The Program Implementation

IAPInfo API provides information about the Access points and Destination network names that are available in the system . Attributes of an access point such as ID, Name, service type and bearer type can be enquired by using the methods of the Accesspoint class. The available destination networks, their IDs, names and the Ids of the access points available within a destination network can be enquired through the methods avilable in the Destination Network class of the IAPInfoAPI.



Figure(3.1) IAPInfo diagram

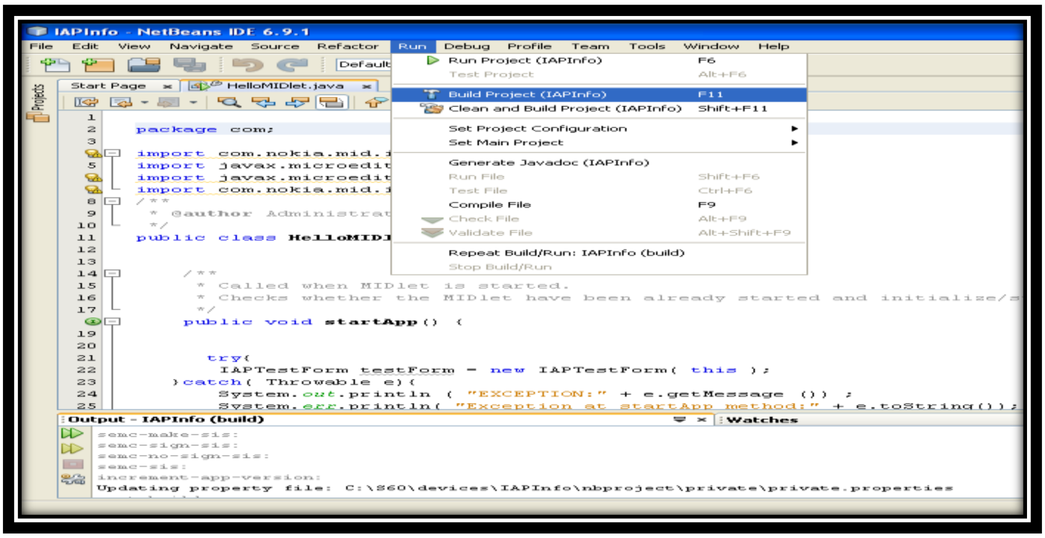
This project concerned detection the names of access point .

3.1 Creating The project Environment

The project environment is constructed using the instructions that come with the Symbian SDK for Nokia devices, for building MIDlets in IDEs or using command line.

3.2 Building and Running in an Emulator

We created all the necessary files for the project, using NetBeans IDE emulator. The figures 3.2 and 3.3 show build and run steps.

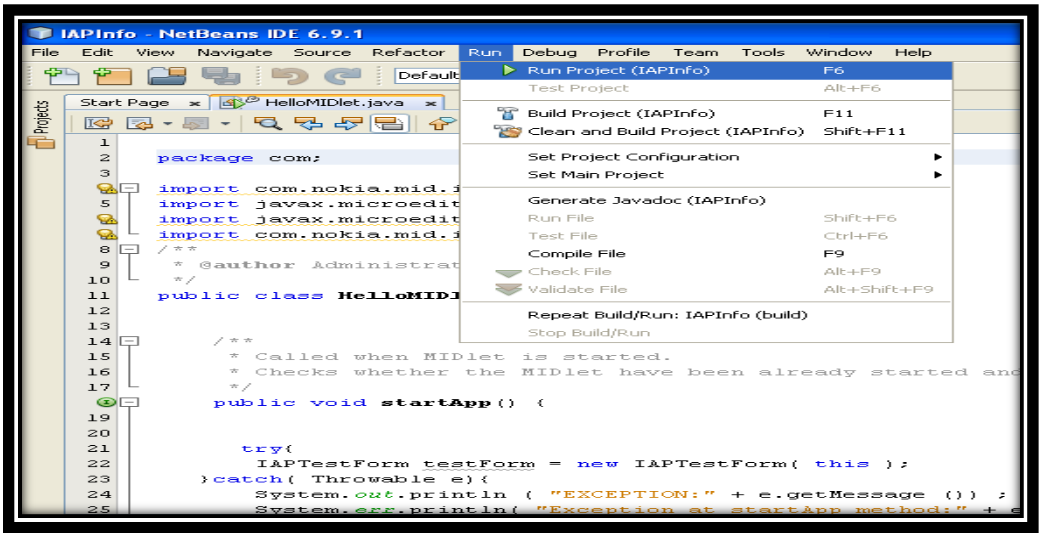


```
package com;

import com.nokia.mid.i
import javax.microedit
import javax.microedit
import com.nokia.mid.i
/**
 * @author Administrat
 */
public class HelloMIDl


    /**
     * Called when MIDlet is started.
     * Checks whether the MIDlet have been already started and initialize/s
     */
    public void startApp() {


       try{
        IAPTestForm testForm = new IAPTestForm( this );
     }catch( Throwable e){
        System.out.println ( "EXCEPTION:" + e.getMessage ()) ;
        System.err.println( "Exception at startApp method:" + e.toString());
```

```
semc-make-sis:
semc-sign-sis:
semc-no-sign-sis:
semc-sis:
increment-app-version:
Updating property file: C:\S60\devices\IAPInfo\nbproject\private\private.properties
```

Figure(3.2) Build the program

Select Run from the toolbar to execute the program as shown in the figure(3.3).



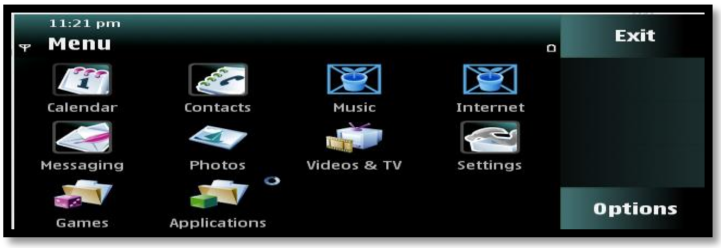
```
package com;

import com.nokia.mid.i
import javax.microedit
import javax.microedit
import com.nokia.mid.i
/**
 * @author Administrat
 */
public class HelloMIDl


    /**
     * Called when MIDlet is started.
     * Checks whether the MIDlet have been already started and
     */
    public void startApp() {


       try{
        IAPTestForm testForm = new IAPTestForm( this );
     }catch( Throwable e){
        System.out.println ( "EXCEPTION:" + e.getMessage ()) ;
        System.err.println( "Exception at startApp method:" + e
```

Figure(3.3) Run the program

3.3 The Setting of Mobile Phone

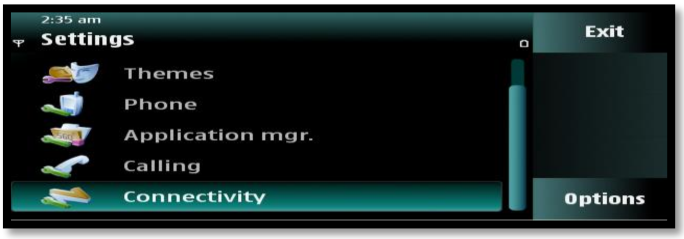
Before execute the program must setting of the mobile phone, where the setting be as the following :

1.Select setting from menu as shown in the figure(3.4).



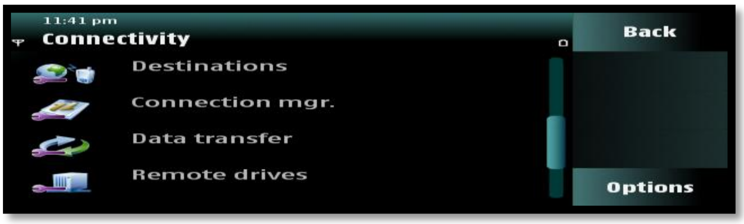
Figure(3.4) show the setting

2.Select connectivity as shown in the figure(3.5).



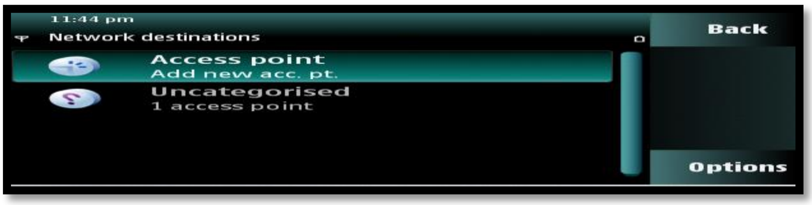
Figure(3.5) show the connectivity

3.Select destinations as shown in the figure(3.6).



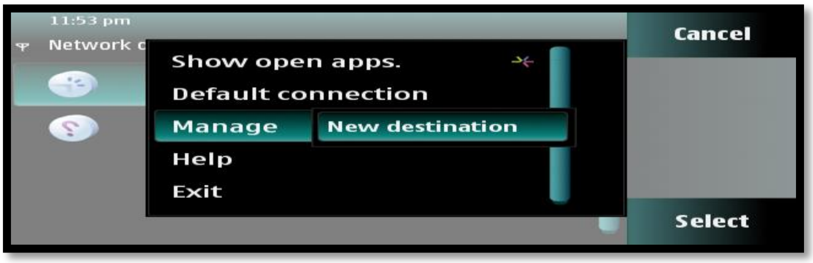
Figure(3.6) show the destinations

4.Select access point as shown in the figure(3.7).



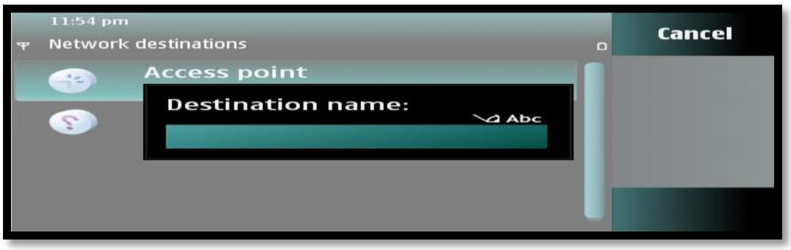
Figure(3.7) show the access point

5.Select manage and then choose new destination as shown in the figure(3.8).



Figure(3.8) show the manage and new destination

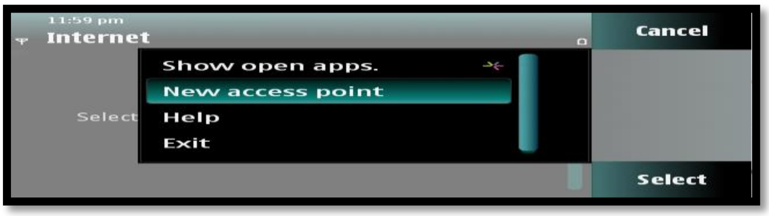
6.enter the destination’s name and then click OK as shown in the figure(3.9).



Figure(3.9) show insertion destination's name

Will add new destination to the menu.

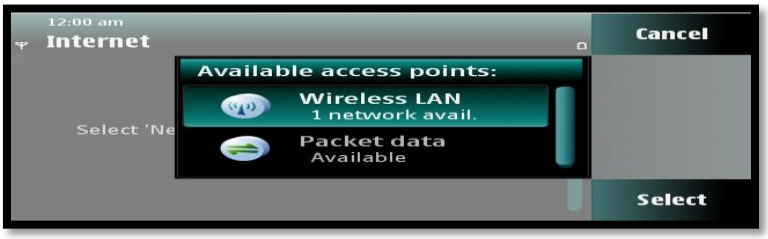
7.from the command option choose new access point as shown in the figure(3.10).



Figure(3.10) show new access point

Then click YES.

8.Select the wireless LAN as shown in the figure (3.11).

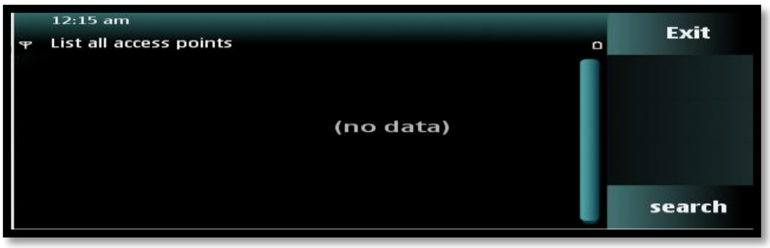


Figure(3.11) show wireless LAN

Then choose first selection and thus will add new access point .

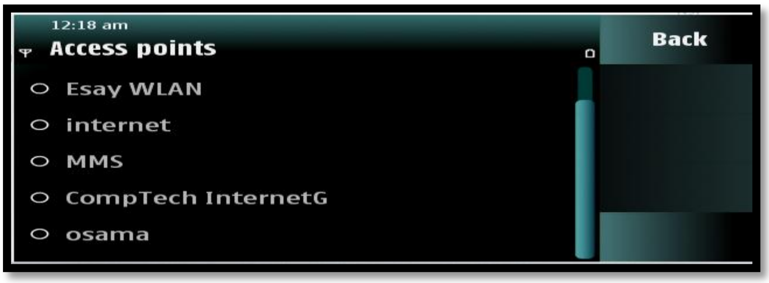
3.4 How to use the IAPInfoMIDlet

1- Start the IAPInfoMIDlet, which then loads the IAPTestForm class. This class contains the command "search" ,and by pressing this command we get the names of access point that are available in the system and also contain the command "exit", and by pressing this command will stop the execution as shown in the figure(3.12) .



Figure(3.12) show the first interface

2- The ACPTest class contains a list of the available access points and the command "back" and by pressing this command will back to the previous interface as shown in the figure(3.13).



Figure(3.13) show the second interface

3- The TestUtil class contains all the test methods that use the IAPInfo component. Each static method contains the usage of a single feature from the IAPInfo API. All methods are static, so the class file can be included in other applications.

References :

1. Martyn Mallick, "Mobile and Wireless Design Essentials", John Wiley & Sons Ltd, 2003.

2. Bulygin Y, Epidemics of mobile worms.New Orleans. IEEE Computer , Society, Los Alamitos, 2007.

3. Martin de Jode, "Programming Java 2 Micro Edition on Symbian OS
(A Developer's Guide to MIDP 2.0)", John Wiley & Sons Ltd, 2004.

4. Roger Riggs, "Programming Wireless Devices with the Java 2
Platform, MicroEdition, Second Edition", Addison Wesley, 2003.

5. Valentino Lee, Heather Schneider and Robbie Schell, “Mobile Applications: Architecture , Design, and Development”, by Prentice Hall., 2004.

6. Yu Feng & Jun Zhu, “Wireless Java Programming with Java 2

Micro Edition”, Sams, 2001.

7. SYMBIAN LTD. Symbian WWW site, 2004.

http://www.symbian.com.

8. “Symbian OS version v7.0s functional description”, Symbian Ltd,

DIXON, K. 2004.

9. SING LI ; JONATHAN KNUDSEN, Beginning J2ME: From Novice

to Professional, Third Edition, Verlag New York, Inc.,2005.

10. H. M Deitel; P. J. Deitel; T. R. Nieto; K. Steinbuhler, “Wireless

Internet & Mobile Business How to Program”, Prentice Hall, 2001.

11. “Programming the MIDP Lifecycle on Symbian OS”, Martin de Jode , Version: 1.00,2004.

12.V.-BITKOM;E. Mehr Handys, Bundesverband Informationswirtschaft
Telekommunikation und neue Medien, als Einwohner in
Deutschland.; http://www.bitkom.de/41015_40990.aspx, 2006.

13.Wale Soyinka,“Wireless Network Administration(A Beginner’suide)”, The McGraw-Hill Companies, 2010.