

Zi Wang

Mobile Phone Programming

Piano Player Game

Bachelor's Thesis

Information Technology Programme


May 2010



MIKKELIN AMMATTIKORKEAKOULU

Mikkeli University of Applied Sciences

DESCRIPTION

 MIKKELIN AMMATTIKORKEAKOULU Mikkeli University of Applied Sciences		Date of the bachelor's thesis 24.5.2010
Author(s) Wang Zi	Degree programme and option Information Technology	
Name of the bachelor's thesis Mobile Phone Programming Piano Player Game		
Abstract <p>Nowadays, a mobile phone is playing an increasingly essential role in human life. With the development of the high technology and mobile phone industry, mobile phones can be used as computers and for simulating sorts of devices mostly. So entertainment is a promising market in the mobile phone technology.</p> <p>Nokia E66 could support the software compiled on the Java ME platform. So in my thesis, I edited a piano player game program, which can simulate the keys in a real piano. The program was compiled in Netbean Java ME and then downloaded into Nokia E66. The buttons in the mobile phone represent the keys in the piano. They are Do, Re, Mi, Fa, So, La, Ti, Do. In this software, the users could play songs, which like that on a piano.</p>		
Subject headings, (keywords) Symbian, Windows Mobile, Android, BlackBerry, iPhone, Palm, SDK, J2ME, MIDP, MIDlets		
Pages 38	Language English	URN URN:NBN:fi:mamk-opinn 2010A8585
Remarks, notes on appendices 		
Tutor Matti Koivisto	Employer of the bachelor's thesis Mikkeli University of Applied Sciences	

ACKNOWLEDGEMENT

I would like to thank all the people who helped me and inspired me during the final thesis period.

At the beginning, I would like to give my honest thanks to my tutor, meanwhile, my dissertation's supervisor, Matti Koivito. He not only instructs me the academic knowledge, but also the way how to handle the problems in life. When I encountered difficult, his patience and professional skills give me a lot of power to overcome the adverse circumstances.

Secondly, I would like to thanks my Java programming's tutor, Timo Mynttinen. Without his course, I cannot complete my thesis so soon. The most important thing is that his optimistic spirit inspired me a lot. Whenever I cannot compiled a function successfully and feel frustrating, Timo's optimistic spirit would affected me.

Third of all, my friends assisted me a lot during the diploma research study, especially the friends studying computer science. They suggest some good recommendation for me in order to make my program more efficient and occupy the less space.

In the end, I will thanks my beloved parents that encourage me from another remote country.

Here is my deepest thanks again for all of you I mentioned above.

LIST OF FIGURES

Figure 2.1 Interface of Palm OS	3
Figure 2.2 Windows Mobile OS	4
Figure 2.3 Anroid Third-Party Software	5
Figure 2.4 iPhone Interface	5
Figure 2.5 BlackBerry OS	6
Figure 2.6 Maemo Interface in Different Applications	7
Figure 2.7 S60 platform, Main Menu Interface	8
Figure 2.8 Representation of Symbian OS GT Components	9
Figure 2.9 Apperance of E66	11
Figure 3.1 Java Edition	13
Figure 3.2 Generic J2ME Architecture	14
Figure 3.3 MIDP Architecture	14
Figure 3.4 MIDlet Life Cycle	15
Figure 4.1 Permit Protocol	19
Figure 4.2 Custom Installation	20
Figure 4.3 Installing	20
Figure 4.4 Finish Installing Completely	21
Figure 4.5 First step of Netbeans' installation	21
Figure 4.6 Specified Position to Install Netbeans IDE	22
Figure 4.7 InstallShield Wizard	23
Figure 4.8 License Agreement	24
Figure 4.9 Choosing Location	24
Figure 4.10 Copying Files	25
Figure 4.11 Setup Status	25
Figure 4.12 Installation Complete	26
Figure 4.13 Management of Java Platform	27
Figure 4.14 Choosing Type of Platform	27
Figure 4.15 Open S60 SDK	28
Figure 4.16 Searching SDK Platform	28
Figure 4.17 Complete Adding Procedure	29

Figure 4.18 Create A New Project.....	30
Figure 4.19 Choosing S60 Platform.....	30
Figure 4.20 S60 Emulator in J2ME.....	31
Figure 5.1 Connection Between Mobile Phone and PC.....	34
Figure 5.2 Installing Application.....	35
Figure 5.3 Import Software into Mobile Phone.....	35

CONTENTS

1. INTRODUCTION	7
2. OPERATING SYSTEM OF MOBILE PHONE	3
2.1 OVERVIEW OF THE MOBILE PHONE OPERATING SYSTEM.....	3
2.1.1 Palm Operating System	3
2.1.2 Windows Mobile Operating System.....	4
2.1.3 Android Operating System	4
2.1.4 Iphone Operating System	5
2.1.5 BlackBerry Operating System.....	6
2.1.6 Symbian Operating System	6
2.2 SYMBIAN OS AND SERIES 60 PLATFORM.....	7
2.2.1 The structure of Symbian OS	7
2.2.2 What is Series 60	8
2.2.3 Developing for Series 60 Platform.....	9
2.3 MULTIMEDIA: AUDIO	10
2.4 NOKIA E66 SYSTEM OS	11
3. INTRODUCTION OF THE ENVIRONMENT.....	12
3.1 ABOUT JAVA	12
3.2 JAVA 2 MICRO EDITION	13
3.2.1 Why choose J2ME.....	13
3.2.2 The View of the Architecture.....	13
3.3 MOBILE INFORMATION DEVICE PROFILE(MIDP)	14
3.4 BASICS OF MIDLETS	15
3.4.1 Creating a MIDlet.....	15
3.4.2 The display of MIDlet	17
3.4.3 MIDlet Suite	17
4. PROCESSING OF MY APPLICATION.....	19
4.1 CONFIGURE THE NETBEANS SOFTWARE	19
4.2 S60 SDK FOR SIMULATION	22
4.2.1 Introduction of S60 SDK.....	22
4.2.2 Installing the S60 SDK	23
4.2.3 Adding S60 SDK Emulator to Netbeans IDE	26
4.3 HOW TO CREATE A NEW PROJECT	29
4.4 THE MAIN CODE WITH THE ANNOTATION.....	31
5. RUNNING IN THE REAL ENVIRONMENT	34
5.1 INTRODUCTION OF THE PC SUITE.....	34
5.2 DOWNLOAD THE PIANO GAME.....	34
5.3 HOW TO PLAY THE GAME	36
6. CONCLUSION.....	37
REFERENCES	38

1. INTRODUCTION

Nowadays, a mobile phone is playing an increasingly essential role in human life. According to a recent report of the mobile market, the usage of mobile phones is increasing by 4.2% per season, especially of the phones with intelligent systems. Users not only apply mobile phones in communication, but also treat them as multi-functional devices, such as a translator, a video player, a game box, and a navigator etc. With the development of mobile phone programming, it can be used as a computer and used for simulating all sorts of devices. At the present time, users can even play games on mobile phones.

The mobile phone device market is opening up for software application developers and content creators. Creating an application, downloading it to a phone, and connecting it to the world provides a new business opportunity for the software industry. Investing in a standards-based platform, which enables deployment of the same software on different types of phones and communicators, is a priority for software companies. So entertainment is a basic function of mobile phones already at present, absolutely in the future.

Considering the promising and wide markets, my target was to design a piano player game for Nokia mobile phones. The buttons in the mobile phone, from 1 to 9, represent the keys in the piano. They are Do, Re, Mi, Fa, So, La, Ti, Do. In this software, the users could play songs like with a real piano. Because most of the present platforms can support Java, I compiled it with Java ME Netbeans software. After that, I will download it to the Nokia E66, where I can run it in the real environment.

The structure of my work is as follows:

In Chapter 2, I will introduce some popular intelligent mobile systems widely used around the world, such as Symbian40, Symbian60, Linux and so on. Consequently, I illustrate the S60 system used mainly in my design.

In Chapter 3, the basic illustration of J2ME MIDP and MIDlet are the main content

associated with the method that explains how to use them.

In Chapter 4, I will give the details of how to install the programming environment into my own PC. I also display the code of my piano player game with annotation.

In Chapter 5, a booklet of the piano player game is given to the users in order to make consumers know how to download it, install it, and play it in their own mobile phones.

Chapter 6 will conclude the thesis. I will give some details about the difficulties through the design process and some judgment about the piano play game.

2. OPERATING SYSTEM OF MOBILE PHONE

2.1 Overview of the Mobile Phone Operating System

Typically, a mobile phone operating system is applied in high-level intelligent phones. At present, with the attendance of more and more mobile phone's manufacturers, the competition between different companies is quite intensive. Nowadays, the operating systems mainly used in mobile phones are Palm OS, Symbian, Windows mobile, Linux(Android), iPhone OS and Blackberry Operating System.[1]

2.1.1 Palm Operating System

Palm OS is a mobile phone operating system initially developed by Palm for personal digital assistants (PDAs) in 1996. Palm OS is designed for ease of use with a touchscreen-based graphical user interface. [2] It is based on the simple-used interface, meanwhile, the Palm OS does not require a large amount of memory, the CPU just occupies a little source. For this reason, Palm OS runs in a high speed comparably, which can fulfill the users' requirement. Unfortunately, Palm OS does not support a multithreading function. It means that users cannot apply several softwares at the same time in one Palm mobile phone. Figure 2.1 is the interface of the Palm operating system.



Figure 2.1 The interface of Palm OS

2.1.2 Windows Mobile Operating System

Windows Mobile is created by Microsoft, mostly used in intelligent mobile phones and mobile devices. Windows Mobile's core is Windows 5.2 kernel, and it features a suite of basic applications developed using the Microsoft Windows API.[3] Previously, Windows Mobile produced as the Pocket PC2000 Operating System. Most of the devices come with a stylus pen for entering the command by touching the screen. Nowadays, Microsoft has developed a new type of a platform for mobile phones, Windows Phone 7 on February 15, 2010, at the Mobile World Congress in Barcelona. Additionally, Windows Mobile operating system is able to support third-party software development and the users can purchase the software at the Windows Marketplace.

The following Figure 2.2 is the interface of Windows Mobile operating system.



Figure 2.2 The Windows Mobile OS

2.1.3 Android Operating System

Android is a kind of open source operating system created by the Linux kernel. It allows developers to write managed code in the Java language. Originally, Android is developed by Google, then Open Handset Alliance keep on working on its design. It is based on the software stack API, mainly divided into three sections. The lowest layer only supports the basic function. Other third-party software is created by the company, written by the Java language.



Figure 2.3 The Anroid third-party software

2.1.4 Iphone Operating System

The famous America’s product-iPhone, developed and marketed by Apple. Mac operating system makes itself become No. 3 in the mobile phone’s market. Mac OS has surpassed the Windows Mobile OS. It is based on the Darwin and the API has four layers, the Core OS layer, the Core Services layer, the Media layer, the Cocoa Touch layer. Users can control the mobile phone through swiping, tapping, pinching and reverse pinching. This perfect design makes the iPhone become more convenient than others’ system. The screen is shown as following diagram Figure 2.4/



Figure 2.4 The iPhone interface

2.1.5 BlackBerry Operating System

BlackBerry OS is the proprietary software platform made by Research In Motion for its BlackBerry line of handhelds. BlackBerry OS provides multi-tasking, and makes heavy use of the device's specialized input devices, particularly the trackball, trackpad or touchscreen. The current OS 4 provides a subset of MIDP 2.0, and allows complete wireless activation and synchronization with Exchange's e-mail, calendar, tasks, notes and contacts.[4] BlackBerry is also a kind of mobile phone system that is mostly used by Americans. The Figure 2.5 displays the interface of BlackBerry.



Figure 2.5 The BlackBerry OS

2.1.6 Symbian Operating System

Symbian OS is widely used in China. It is an operating system designed for mobile devices and smartphones, with associated libraries, user interface, frameworks and reference implementations of common tools.[5] It has two platforms for users, respectively Series and UIQ which support the touchscreen function. My mobile phone program is based on the Symbian Operating System, so I will introduce it in detail later.

2.2 Symbian OS and Series 60 Platform

2.2.1 The structure of Symbian OS

Symbian OS, a mobile Operating System from Symbian Ltd is an open, highly robust operating system for data-enabled mobile phones. Symbian OS (formerly called EPOC) is a 32-bit preemptive multitasking operating system that is central to the success of Series 60, and other user interface platforms such as Series 80 and Series 90, the communicator platforms of Nokia, and UIQ of UIQ Technology AB, a division of Symbian. Also the Maemo™ platform, introduced on Nokia Internet Tablet devices, realises a vision to bring PC-like features and user experiences to mobile devices. Maemo 5 is a kind of platform designed for Nokia Nseries mobile phones, which are represented by powerful hardware, nice UI principles, and internet centric philosophy. Based on well known open-source Linux components and the simplicity of Qt development, Maemo 5 gives the developers a new experience to create innovative mobile application.[6]

As the following Figure 2.6 shows the interface of Maemo



Figure 2.6 Maemo Interface in different applications

In the real world, events often happen simultaneously and with timing that is unpredictable, which is usually termed asynchronous behavior. Symbian applications are designed to behave reliably, and interact smoothly with other applications and with

the numerous asynchronous services. For example, a phone call may interrupt a user composing an email message, a user may switch from Messaging to a Calendar application in the middle of a telephone conversation, or an incoming SMS may cause the user to access the Contacts database and forward the SMS onward. By complying with the platform architecture and software design guidelines, application designers can routinely manage such occurrences in the daily lives of smartphone users.[6]

2.2.2 What is Series 60

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. Symbian C++ APIs enable extremely efficient multitasking and memory management. Following Figure 2.7 displays the Symbian's main menu.



Figure 2.7 S60 platform, the main menu interface

A set of robust components and APIs are provided for developers in Series 60 SDKs. The APIs provided are widely used by the suite of “standard” applications that are an integral part of Series 60 Platform. However, the extensive APIs were designed for use by third-party application developers as well.[6] Series 60 adds the extensive Avkon UI layer, a full suite of applications based on the Avkon and Uikon libraries plus a number of key application engines as in the following Figure 2.8.

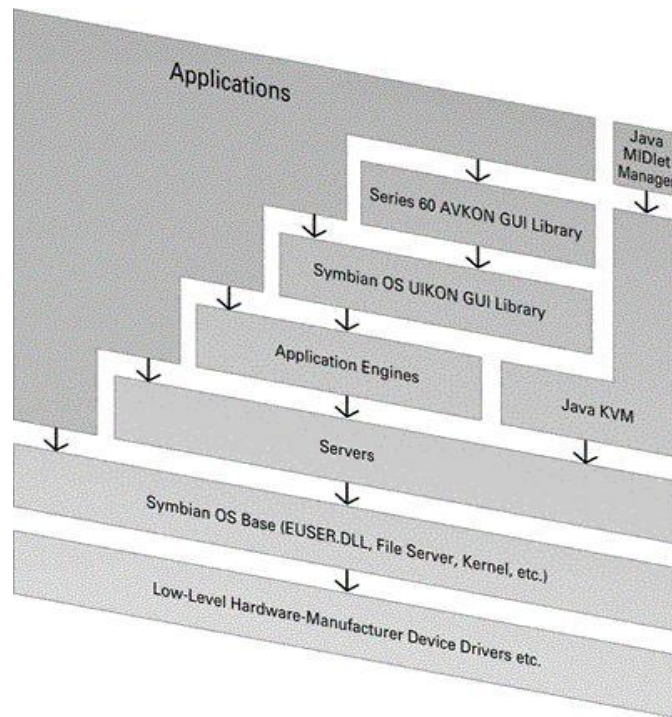


Figure 2.8 Representation of the Symbian OS generic technology(GT) components[7]

2.2.3 Developing for Series 60 Platform

Series 60 was originally designed for one-hand operated smartphones, based around a large color screen and an intuitive user interface(UI). By using standard technologies and open standards it ensures interoperability between different terminal and infrastructure manufacturers.[8] Symbian OS is written largely in C++, the language therefore represents a strong development choice for third parties. According to the Figure 2.8, Series 60 Platform supports to design products and serviced in Java 2 Micro Edition(J2ME). Series 60 has been designed to ensure a safe investment in creating applications for this new mobile market. Symbian implements a wide range of Java APIs including location information, SIP, multimedia, Scalable Vector Graphics and Sensor.[9]

Forecasts, from several trusted industry sources, calculate that during 2007 around 200million smartphones had been shipped to customers and by then the total number of Symbian OS based devices in use were about 500 million. [8] The Symbian OS has by far the largest market share in the smartphone operating systems. That is the reason why I chose the S60 platform as the simulation environment.

2.3 Multimedia: Audio

My software-piano player is based on the audio function. So in this part of my thesis I will introduce some sound effects in Symbian OS. The multimedia architecture is used for performing audio operations, such as recording and playing. A variety of different audio formats for example, wav and midi are supported for use within applications.

To handle the various formats, a plug-in architecture is used to match the file format to an appropriate codec(coder/decoder). Besides this, it is also possible to explicitly select a codec, which is necessary when dealing with raw audio data.

Utility classes are available for carrying out the essential audio tasks, and provision is made for recording, streaming, converting between audio formats, tone playing and audio file playing. From a technical standpoint, an observer mechanism is used, with each utility having a dedicated observer class and callbacks. This allows applications to respond to the asynchronous nature of interacting with the audio utility classes.[8]

Consequently, there are two libraries of interest-MediaClientAudio.lib and MediaClientAudioStream.lib. First library is used to record and play sounds, meanwhile, it can be converted between audio formats. The second library allows multimedia clients to stream audio data.The various aspects of Series 60 sound are illustrated across two example applications, they are AnsPhone and Audio.

The playing of audio data, such as .wav and .midi, is made possible by the CmdaAudioPlayerUtility class and its associated observer, MmdaAudioPlayerCallback. This observer has two functions to inform the client application of the current status of CmdaAudioPlayerUtility:MapcInitComplete() indicates that the data source has been opened successfully, while MapcPlayComplete() signals that playing has been concluded. As with the callback functions used in tone playing, both accept a Tint parameter to indicate if any errors have taken place.[8]

2.4 Nokia E66 System OS

E66 is a kind of business mobile phone that apply Symbian OS 9.2 + S60 Version3.1 platform. There is a CPU that implements in the speed of 369MHz. Even though it is a little low compared with the speed of a CPU used in a PC, the E66 can run the software very well.



Figure 2.9 The appearance of E66

Since my E66 is not a kind of touch-screen mobile phone like the iPhone, it only can use the keyboard to play the “piano”, and the audio equipment located on the bottom of back of the mobile phone.

3. INTRODUCTION OF THE ENVIRONMENT

3.1 About Java

A set of Java bytecodes are transformed into Java programming language by Java compiler, which are instructions for an abstract computing machine referred to as a virtual machine for running a Java program.[10] Java code can be compiled and run straight into native machine binary code.

Java's advantages are the following:

- Inherent portability: execute it without having to recompile the program on various operating systems
- Security infrastructure: Java enables it to download securely and execute third-party code safely from verification to integrity of the generated bytecode.
- Simple: with a large amount of methods(function) supported by different classes
- Exception: manage the flow of the programming

Java editions are the following[10]:

Standard Edition(J2SE): Developed for the workstation computers and some devices like the desktop.

Enterprise Edition(J2EE): Designed to run on Servlets, JSP, and XML, targeted for server-based applications

Micro Edition(J2ME): Built for devices with limited memory space, display and power

Figure 3.1 shows various Java editions

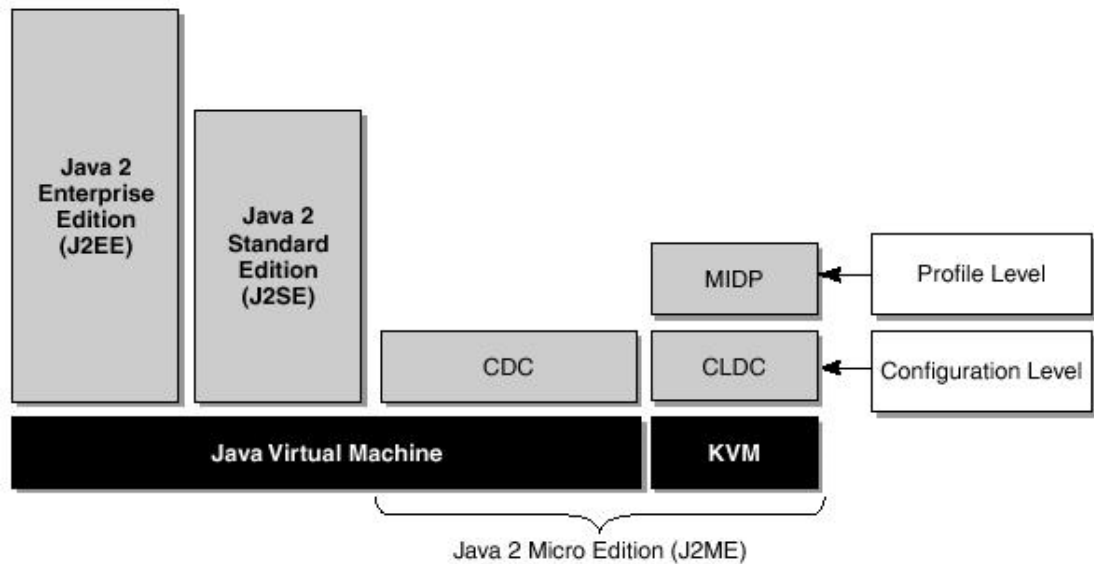


Figure 3.1 Java edition[11]

3.2 Java 2 Micro Edition

3.2.1 Why choose J2ME

J2ME is developed to be used at consumer devices with limited horsepower such as a mobile phone or pager and so on. Many such devices are disable to install or download software that was not configured during the period of manufacturing. With the introduction of J2ME, micro devices can be more attractive than what they were in nature.[10] It means that the devices with limited memory and horsepower can be developed advance and deeply. It is not the same way to download Java applets like a web browser does. The implementation of J2ME on a device supports the option to browse, download and install Java applications.

3.2.2 The View of the Architecture

If only users have compiled some Java source code into one or more class files, and sometimes included them in a Java Archive file, then the JVM will change the type of class files to another way, the machine code for the platform running the JVM. Of course, the allocating, providing security and freeing memory are also the

responsibility of JVM. That is what lets the Java programs go.

At last, let's put all the information about J2ME together into two separate scenarios. The following chart shows a generic software architecture.

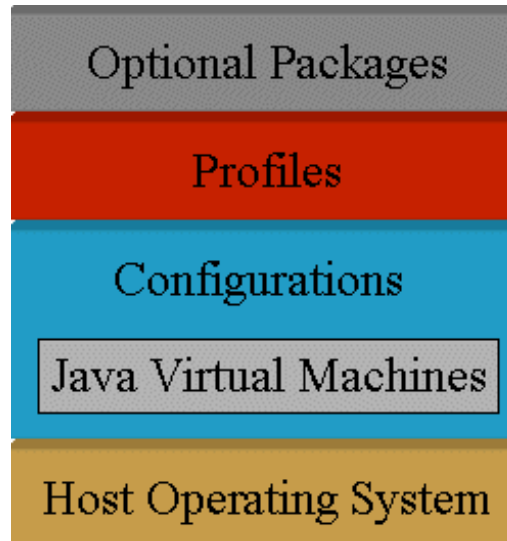


Figure 3.2 Generic J2ME architecture[12]

3.3 Mobile Information Device Profile(MIDP)

The MIDP architecture is based on the host operating system. The following Figure 3.3 shows the architecture of MIDP

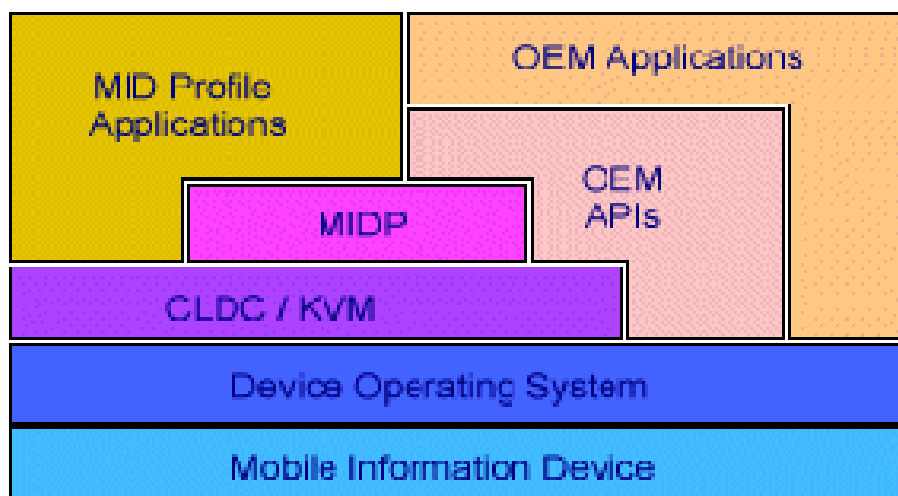


Figure 3.3 MIDP architecture[13]

As we can see in Figure 3.3, it begins with the hardware which is displayed as Mobile Information Device at the bottom of the diagram. One step up, there is the device

operating system, based on the hardware. The CLDC or KVM is installed on the device operating system, meanwhile, it is the foundation for MIDP. Notice that MIDP applications have access to both of the libraries of CLDC and MIDP.

OEM APIs (original equipment manufacturer) are provided by the manufacturer of the device. And the OEM applications may access MIDP and/or OEM APIs.

3.4 Basics Of MIDlets

3.4.1 Creating a MIDlet

A MIDlet is an application that runs on the MIDlet class. In the class, there are some methods that the application manager can use to communicate with a MIDlet. The most important thing is that the communication not only can answer, but also can request. For instance, the application manager can pause a MIDlet (to allow a user to answer an incoming phone call), at the same time, a MIDlet can occur a requirement to be paused (after the phone call, the application can keep on working)

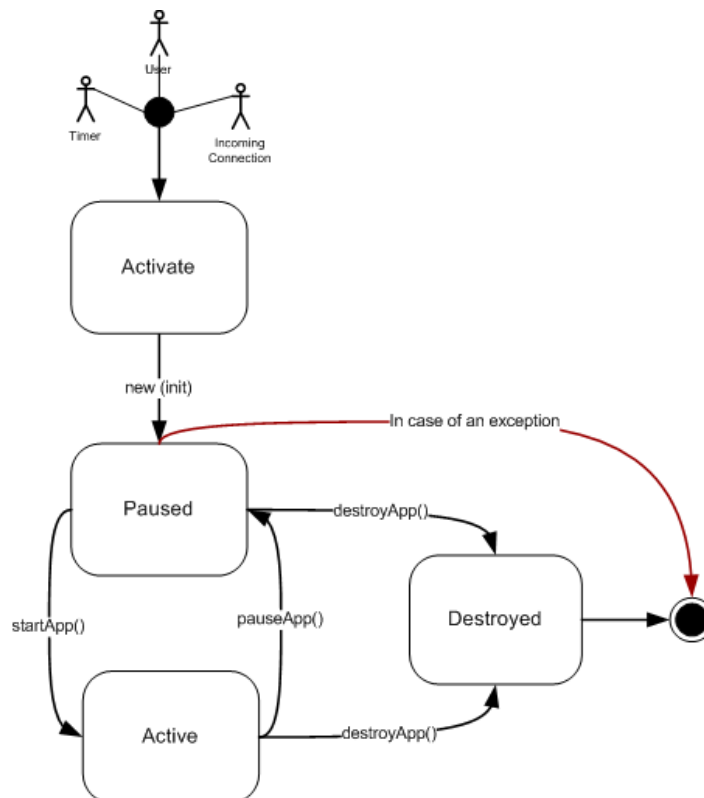


Figure 3.4 The MIDlet life cycle[14]

There are several phases in the whole MIDlet life cycle. At any given time, a MIDlet is in one of three states: Paused, Active, or Destroyed. [15]A state diagram that shows how these states are related and the legal state transitions are shown in Figure 3.4

The manager creates a MIDlet by extending the MIDlet class. This class includes three abstract methods, *startApp()*, *destroyApp()*, and *pauseApp()*.

Following is the shell of a MIDlet. It includes all the methods required by the MIDlet class[15]:

```
Public class Shell extends MIDlet
{
    // This method (constructor) is not required
    Public Shell( )
    {
    }

    // The application manager will invoke this method to start the MIDlet
    Public void startApp( )
    {
    }

    // The application manager will invoke this method before pausing the MIDlet
    Public void pauseApp( )
    {
    }

    // The application manager will invoke this method to prior to shutdown
    Public void destroyApp(boolean unconditional)
    {
    }
}
```

3.4.2 The display of MIDlet

Every MIDlet has a reference to one Display object. This object can retrieve some information about the current display, such as the range of colors supported, and includes methods for requesting that objects (Forms, TestBoxes and so on) be displayed. The Display object is the best display controlling for what is shown on the device and when it should be displayed.

The following shows the Display object held for the lifetime of the MIDlet in a variable[10]:

```
Public class DisplayStats extends MIDlet
{
    Private Display display; // Reference to Display object

    // MIDlet constructor
    Public DisplayStats( )
    {
        Display = Display.getDisplay(this)
        .....
        .....
    }
    .....
}
```

3.4.3 MIDlet Suite

MIDlet suite is a kind of package of the MIDlet. In other words, MIDlet suite consists of one or more MIDlets software, source files and JAR manifest together. All of the content is packed for a Java Archive (JAR) file.

MIDlet suite is developed as a model for solving the problem of MIDlet controlled access and shared resources. Considering the security problem, the MIDlet and source files located in the MIDlet suite cannot be installed, deleted or updated independently.

MIDlet suite must be treated as a whole package for its operations.

4. PROCESSING OF MY APPLICATION

4.1 Configure the Netbeans Software

Netbeans is software especially created for Java programming. The versions for Windows, OS2, OpenVMS and Linux operating systems have the same function although the execution environment is different. This software contains all kinds of models used for compiling various types of application. The most important thing is that manager can create a new MIDP application program. Moreover, MIDlet also can be tested on Netbeans. So I used Netbeans to design and implement my own application.

However, before the Netbeans software's installation, another tool Java Development Kit needs to be installed in the PC. Java Development Kit enables the Java compiler and the application to create Java Archive files(jar.exe). Without it, a compiler cannot create applications for the mobile devices.

Here are the steps of the JDK installation:

First of all, double click the coffee icon. Then, a window will be displayed on the screen just like shown in Figure 4.1. It is a protocol I need to accept. Then, click the accept_(A) button in order to proceed to the next step.



Figure 4.1 Permit protocol

At this interface, I can choose the optional functions I want to install, and change the path in which they want to save the software. I just click the next_(N) button to install the JDK to the default destination. Consequently, the JDK will be installed

automatically. Just like the following figures show.



Figure 4.2 Custom installation



Figure 4.3 Installing

At last, the Java Development Kit has been installed completely by clicking the finish(F) button.



Figure 4.4 Finish installing completely

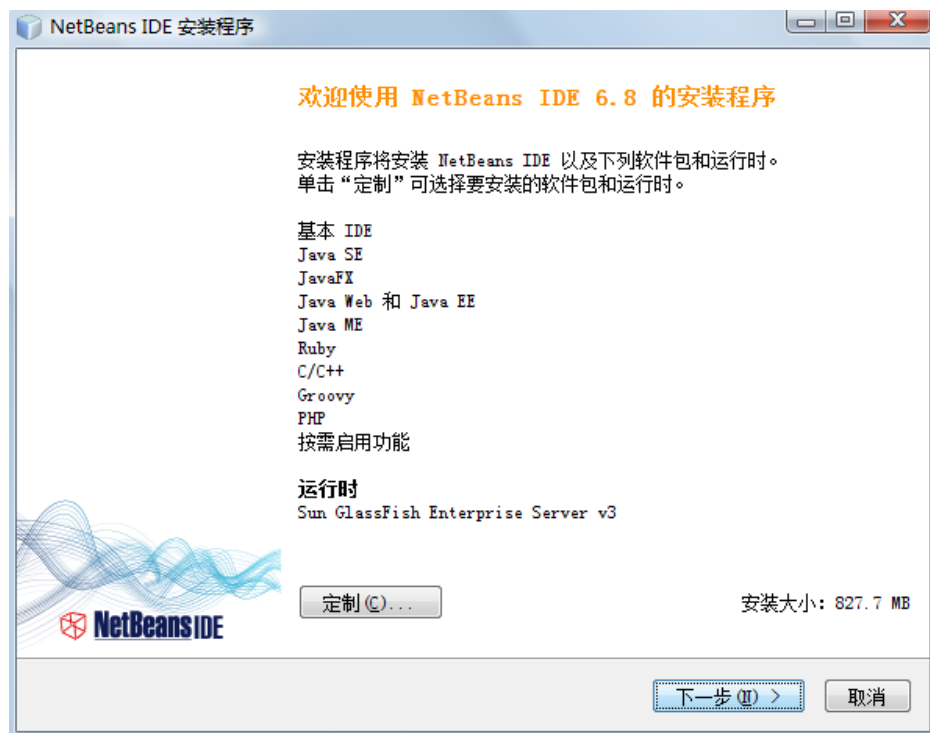


Figure 4.5 First step of Netbeans' installation

Then, I can start to install the Netbeans IDE, the main environment of design. Firstly, double click the Netbeans IDE icon and a window shows the different software packages concluded in the Netbeans IDE such as Java ME, Java Web and C/C++, etc. (Figure 4.5 above). Then, click the next_(N) button to keep on running.

Consequently, it is also the permit protocol step like what is shown in Figure 4.1. So, tick the selection of agreement and press the next(N) button again.

I have installed the JDK before, thus, it can be found in the disk C directly. Otherwise, the Netbeans IDE cannot be installed to the specified position. Click the next(N) button again and again.

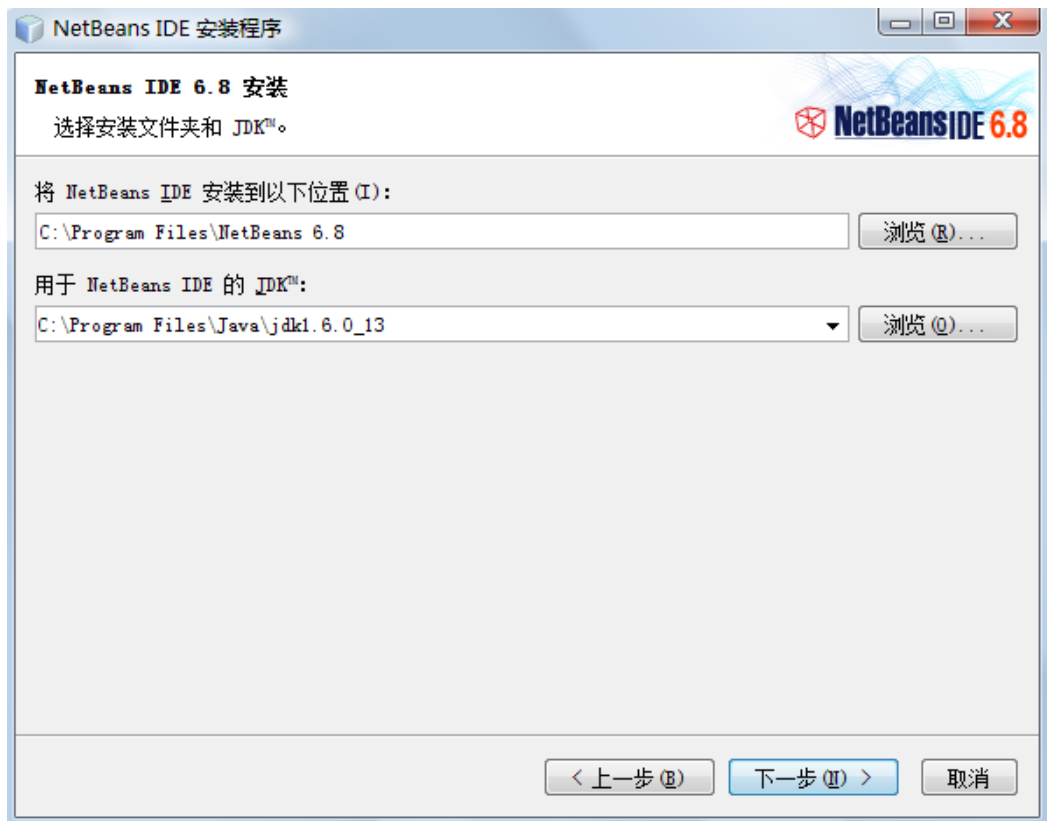


Figure 4.6 Specified position to install Netbeans IDE

In the end, the Netbeans IDE has been installed completely.

4.2 S60 SDK for simulation

4.2.1 Introduction of S60 SDK

S60 SDK is a kind of the Nokia's official development tool. With the help of SDK I can design and implement the MIDP applications for S60 platform smartphones on my own PC. Since I have installed the Netbeans IDE(Integrated Development Environment), together with it the SDK provides all the functions such as APIs,

sample code and files needed for developing the new S60 MIDlets, that is, applications that conform to the MIDP.[16] S60 SDK is the third edition Nokia platform emulator that can simulate everything except calling outside.

4.2.2 Installing the S60 SDK

Installation of the S60 SDK includes the following steps:

Run the installation executable(setup.exe).

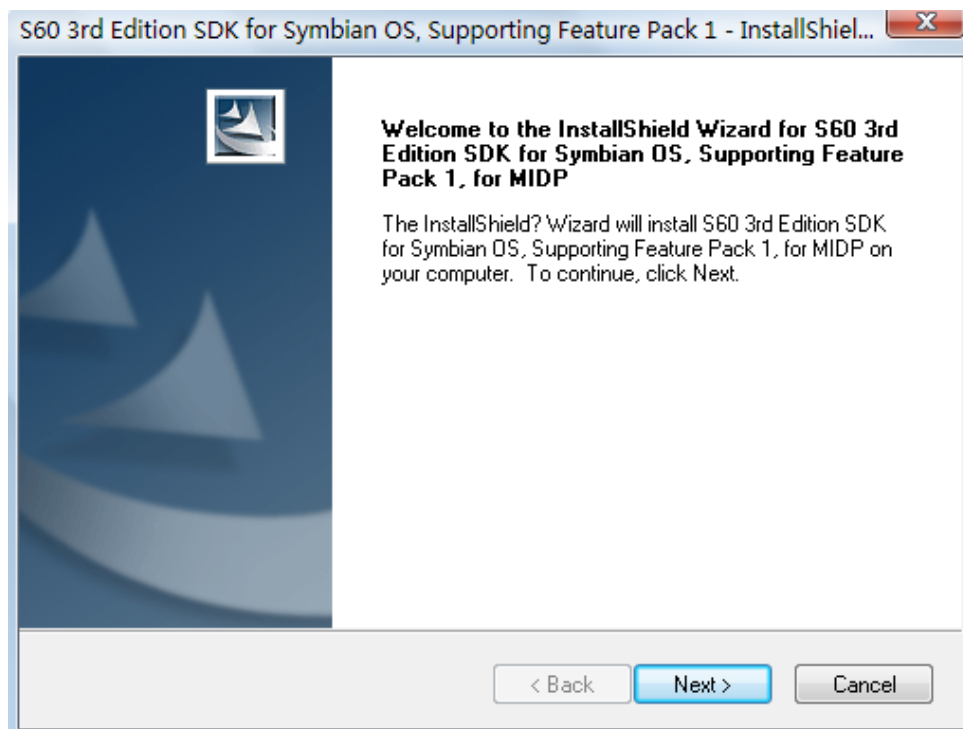


Figure 4.7 InstallShield Wizard

Click Next then the License Agreement is displayed as follows

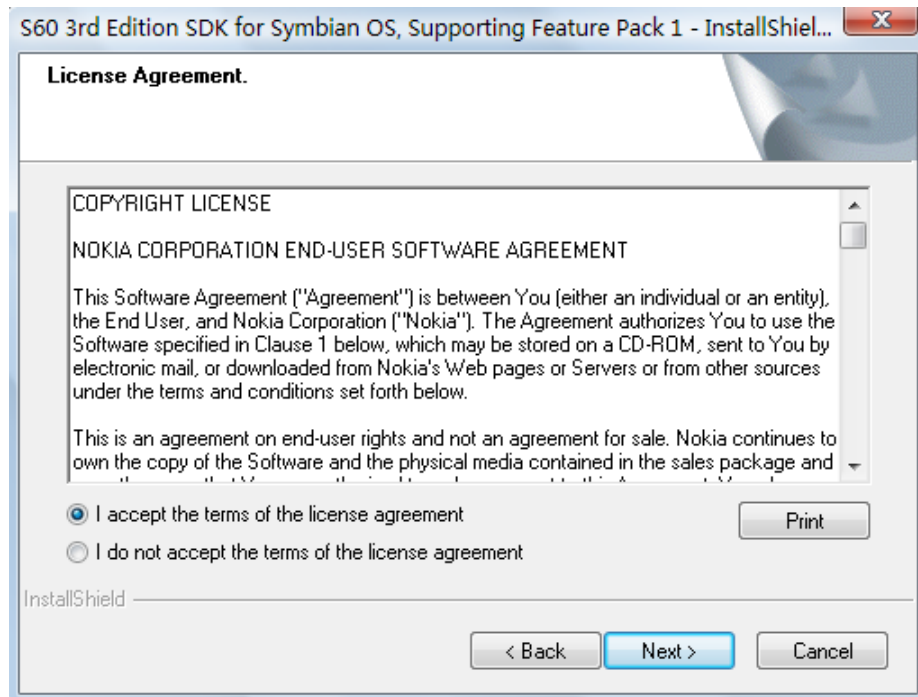


Figure 4.8 License Agreement

Tick the option of “I accept the terms of the license agreement”, then click the Next button. After choosing the destination location, click the Next button again.

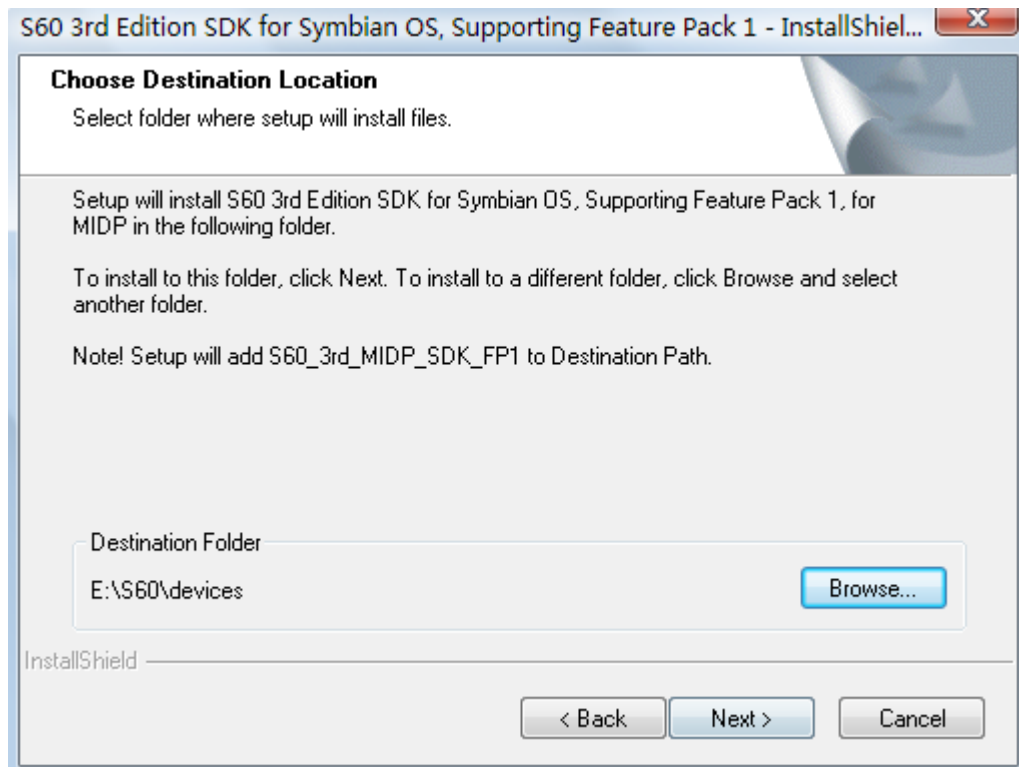


Figure 4.9 Choosing the location

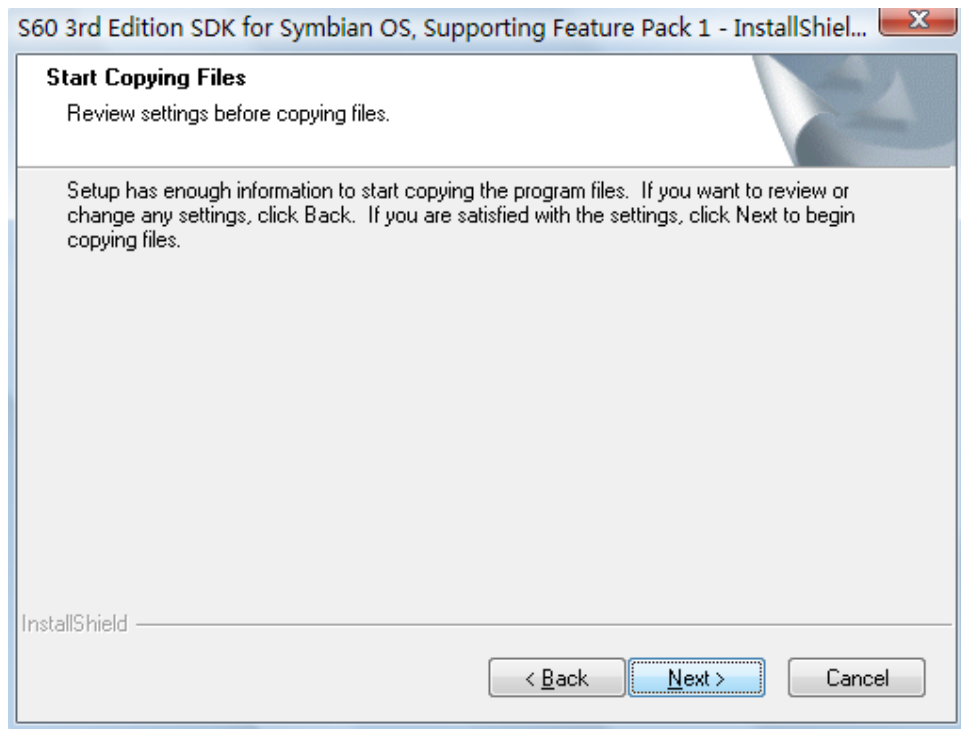


Figure 4.10 Copying Files

Click Next and the installshield wizard start to setup the SDK.

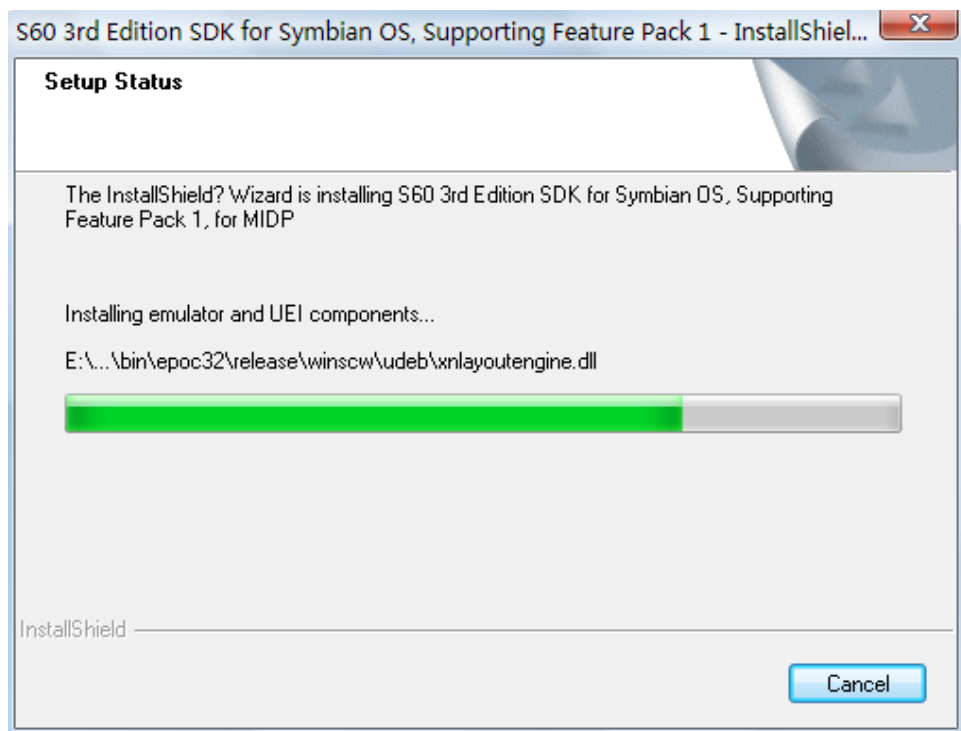


Figure 4.11 Setup Status

Once the installation is complete, the following dialog is displayed. To complete the installation, click the Finish button.

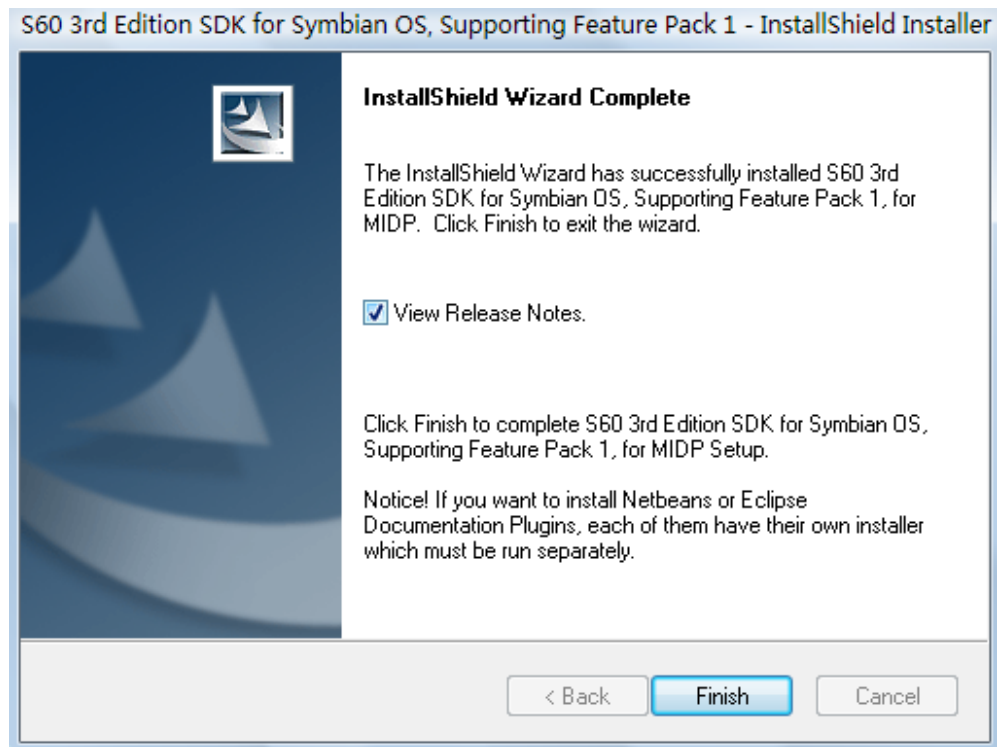


Figure 4.12 Installation complete

4.2.3 Adding S60 SDK Emulator to Netbeans IDE

Adding S60 SDK Emulator to Netbeans IDE includes the following steps:

In order to integrate the SDK emulator with Netbeans IDE, I need to add the SDK into the Netbeans IDE software. To be able to do that, double click the Netbeans icon, select the Java Platform in the Tool(T) option. After that action, the window shown in Figure 4.13 will be displayed, then you can see three platforms in the blank. Since my application is based on the J2ME, choose the J2ME and click the adding platform button on the bottom left.

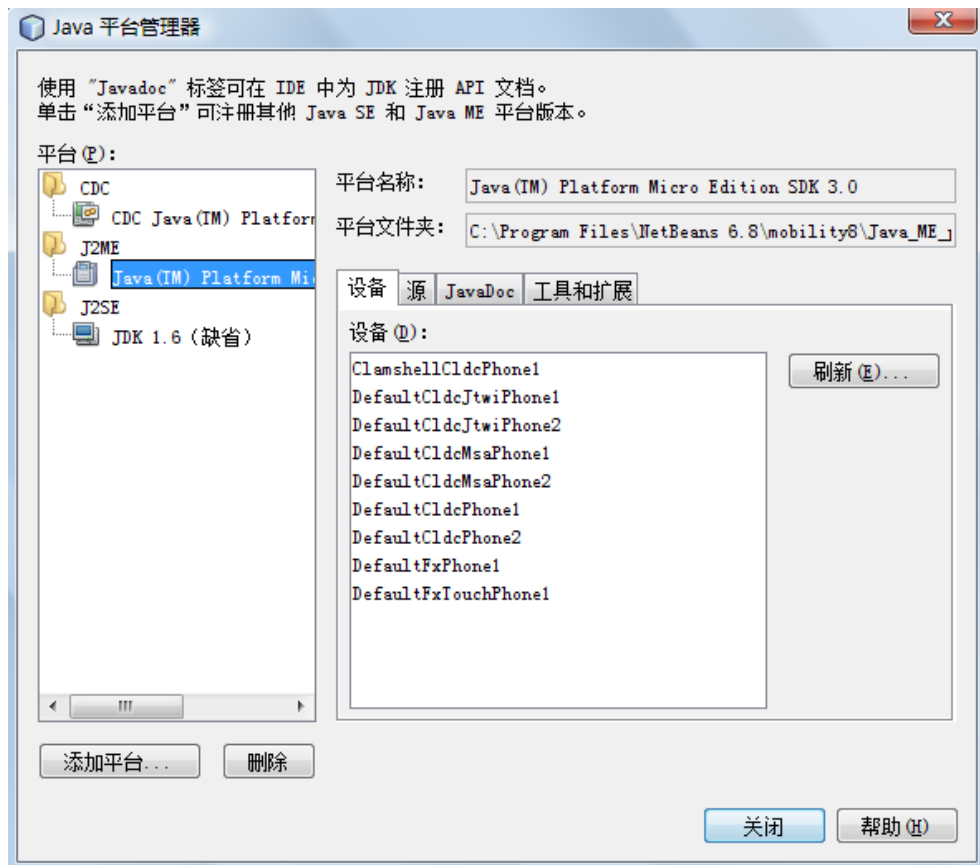


Figure 4.13 Management of Java Platform

In the next step, tick the Java ME MIDP platform emulator and the Next button.

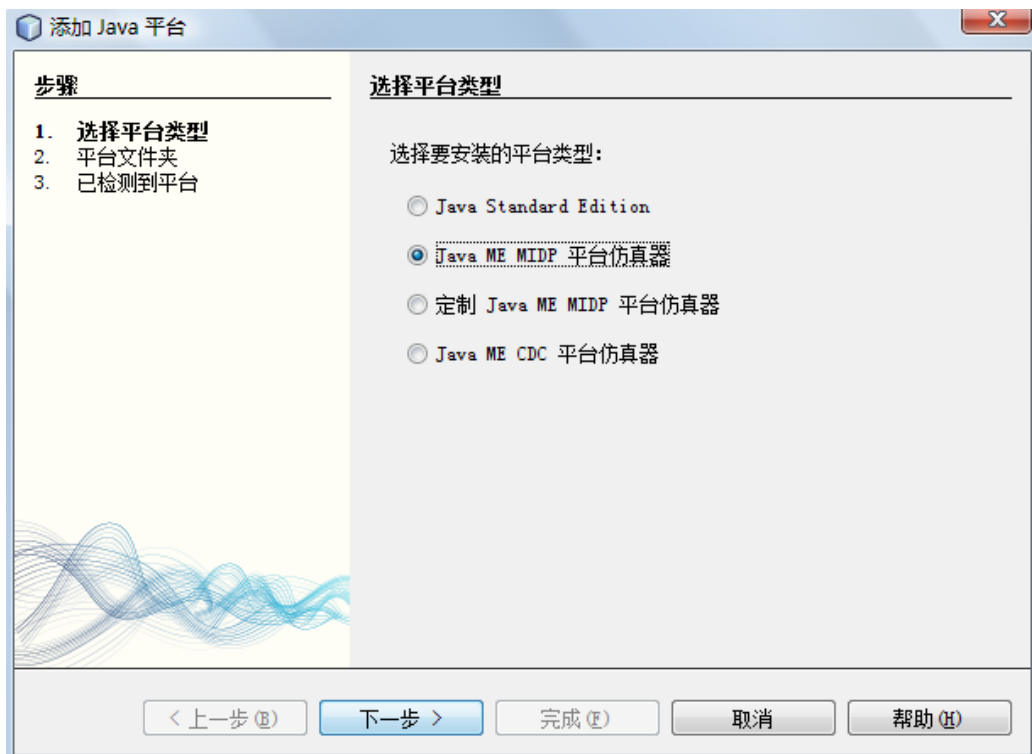


Figure 4.14 Choosing the type of platform

Now you have to find out the file where the S60 SDK is located and open it. Consequently, the software wizard will search the SDK platform automatically just

like in the following diagrams Figure 4.15 and Figure 4.16.

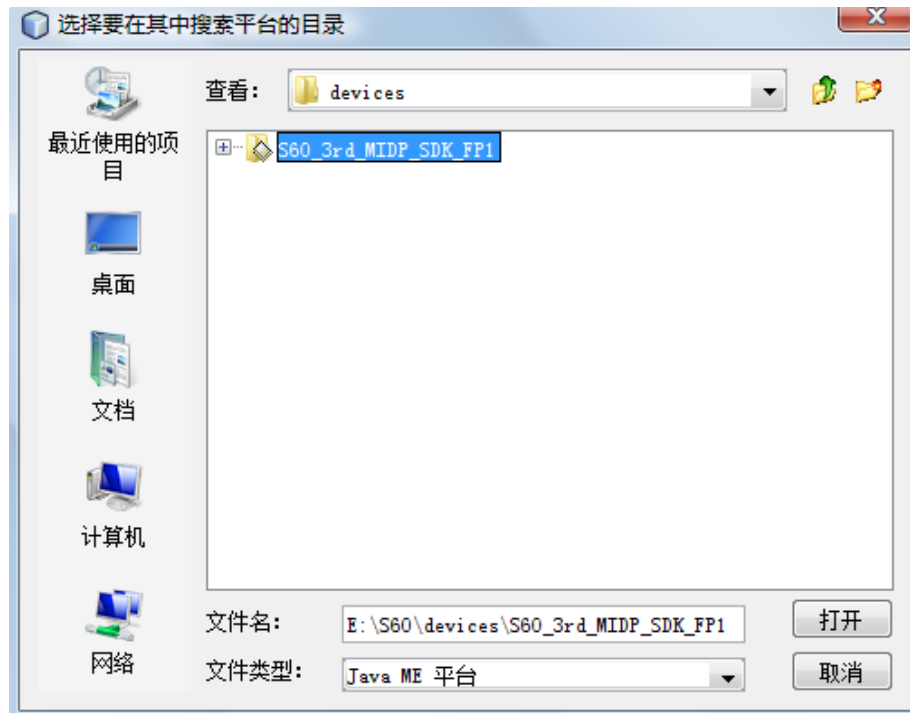


Figure 4.15 Open the S60 SDK

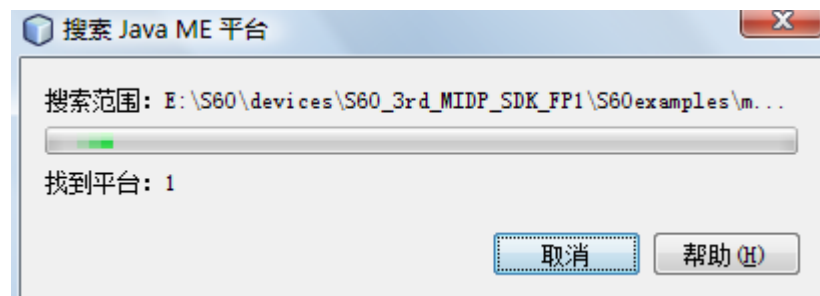


Figure 4.16 Searching the SDK platform

At last, the wizard finds the SDK platform, then tick that option. And the adding procedure has been completed after press the Finish button.

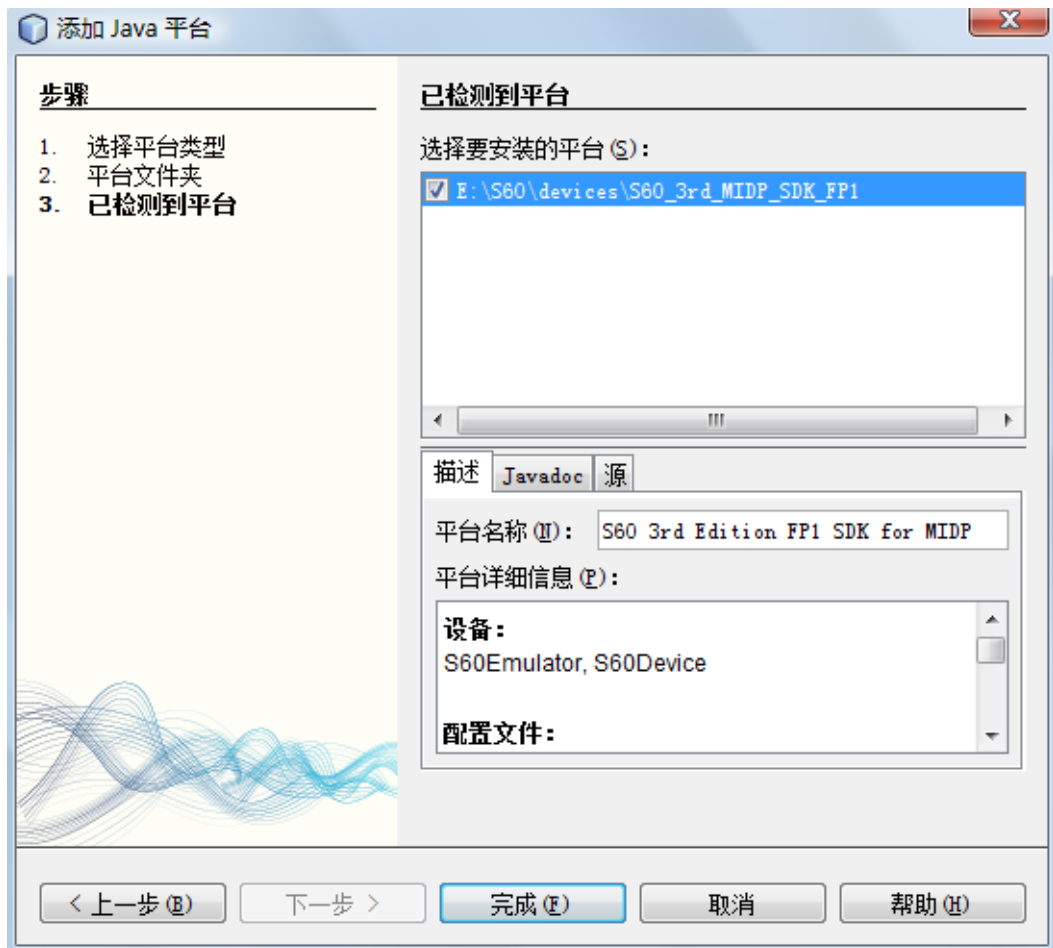


Figure 4.17 Complete the adding procedure

4.3 How to create a new project

Creating a new project includes the following steps:

Open the Netbeans IDE software by double click the icon. In order to open a new project, select the File(F) → New project, then the following diagram Figure 4.18 will be displayed. Choose the Java ME platform and the Mobile application program option.



Figure 4.18 Create a new project

Then the most important step is choosing the S60 3rd Edition SDK platform as the emulator. With the CLDC-1.1 configuration and configure MIDP-2.0 as the device file as the Figure 4.19 shows. Figure 4.20 represents the S60 emulator.



Figure 4.19 Choosing the S60 platform



Figure 4.20 S60 emulator in J2ME

4.4 The main code with the annotation

The following code used to get the current time of the system,, meanwhile, transfer the time unit to millisecond. The interval of the next action is about 100 milliseconds. At the same time, it redraw the graph that displayed on the mobile phones' screen, the code "getWidth" and "getHeight" can get the width and height of the screen:

```

public void run( )
{
    long T1 = System.currentTimeMillis();
    long T2 = T1;
    while(m_bRunning)
    {
        T2 = System.currentTimeMillis( );
        if( T2 - T1 > 100 )
        {
            T1 = T2;
            repaint(0, 0, getWidth(), getHeight());
        }
    }
}

```

```
}  
}
```

The following code means that when a user presses the Number 1 button in the mobile phone, then the mobile phone will sounds like Do. When the user presses the Number 2 button in the mobile phone, then the mobile phone will sounds like Re. Pressing the Number 3 button in the mobile phone, then the mobile phone will sounds like Mi:

```
protected void keyPressed(int keyCode)  
{  
  try{  
    switch(keyCode)  
  {  
    case KEY_NUM1:  
      Manager.playTone(60, 500, 100);  
      break;  
    case KEY_NUM2:  
      Manager.playTone(62, 500, 100);  
      break;  
    case KEY_NUM3:  
      Manager.playTone(64, 500, 100);  
      break;  
      .....  
      .....  
  }  
}
```

The following code means filling the screen with black color:

```
protected void paint(Graphics g)  
{  
  g.setColor(0x00000000);  
  g.fillRect( 0, 0, getWidth(), getHeight() );
```

```
}
```

Once there is a income calling or other application's running result to the pause of the piano game, it can be paused temporarily as the following code's implementation:

```
protected void pauseApp()  
{  
    m_MainCanvas.Stop();  
}
```

```
protected void destroyApp(boolean arg0)  
throws MIDletStateChangeException {  
    m_MainCanvas.Stop();  
}
```

5. RUNNING IN THE REAL ENVIRONMENT

5.1 Introduction of the PC Suite

Nokia PC Suite is an application designed for Nokia mobile phones based on the Windows System platform. Through the connection of a cable or wireless, it can be used for transmitting data between computers and mobile phones. For instance, editing the user's phone book, deleting the client's messages, and so on. This program applies to all kinds of Nokia mobile phone models. According to the user's different types of Nokia mobile phones, PC Suite can make users synchronize, edit and backup the files inside their mobile phones, moreover escalating users' firmware of the Nokia mobile phones.

5.2 Download the Piano Game

First of all, connect the mobile phone to the PC with the data cable like in the following Figure 5.1. Then choose the PC Suite option from the mobile phone.



Figure 5.1 Connection between mobile phone and PC

Double click the PC suite icon and choose the icon which is marked by a red circle like the following Figure 5.2. It is used for installing Java or Symbian SIS applications.



Figure 5.2 Installing application

Once users implement the installing application, the following diagram will be displayed. Then select the music.jar file and click the array icon to import the piano game into my own mobile phone.

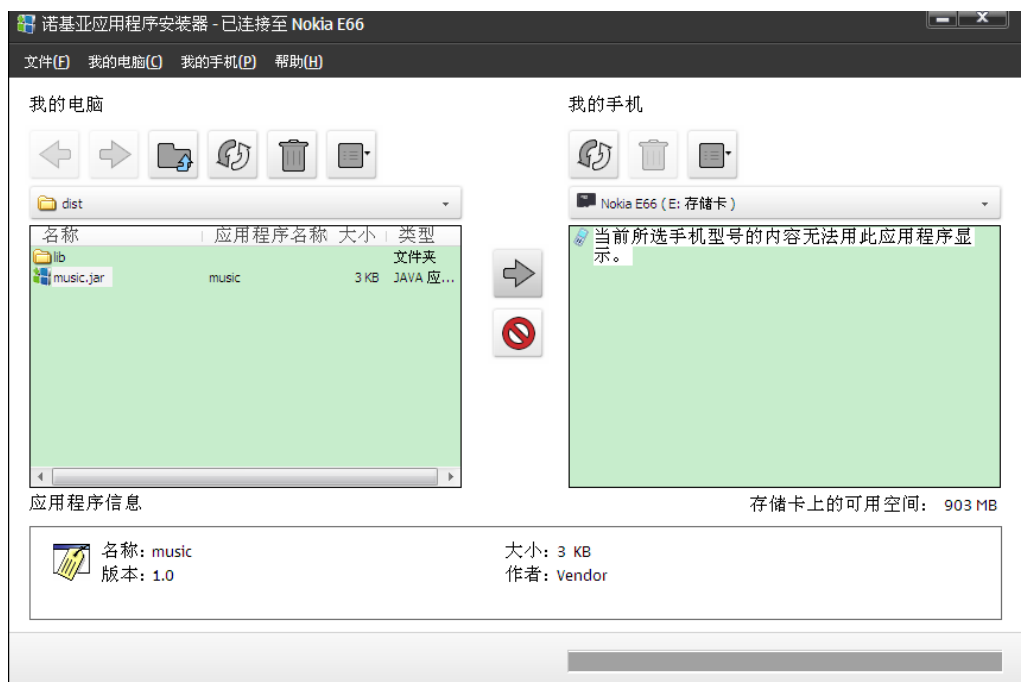


Figure 5.3 Import software into mobile phone

5.3 How to play the game

After downloading the game into a mobile phone, find out the location of the software stores and open it. In this software, the users could play songs like with a real piano. You will see that the mobile phone's screen is filled with black. In this situation, the buttons in the mobile phone, from 1 to 9, represent the keys in the piano. They are Do, Re, Mi, Fa, So, La, Ti, Do. For instance, if you press the Number one button, the mobile phone will make a sound like the note Do. Pressing the Number two button, it will make a sound like the note Re. The regulation is what I have explained above.

6. CONCLUSION

In fact, the piano game has some disadvantages as well. The main drawback is the time delay in the execution. For example, when a user presses the Number one button, he/she cannot hear Do immediately. So the user must wait for a while. However, when I simulated the piano game on the S60 SDK emulator platform, it run without any problems. Maybe that's the problem of CPU run in the mobile phone, which has less speed than the CPU run in the PC.

In addition, there is another disadvantage exists in the piano game: users cannot play chords. For instance, users cannot press more than one button at the same time. Even though you do that, you can only hear one note from the mobile phone. Maybe in the future design, I can give more modifications on this point.

Through the whole process of designing mobile programming, I have understood the steps how to create a mobile application from the very beginning to the end. What kind of developing software I can use to compile corresponding application. What's more, I have already been acknowledged with the job as a mobile phone program developer.

References

- [1]<URL <http://baiki.baidu.com/view/148382.htm>
- [2]<URL http://en.wikipedia.org/wiki/Palm_OS
- [3]<URL http://en.wikipedia.org/windows_mobile
- [4]<URL http://en.wikipedia.org/wiki/BlackBerry_os
- [5]<URL http://en.wikipedia.org/wiki/Symbian_os
- [6]Developing Series 60 Applications, 2001 by Edwards
- [7]<URL http://images.cnblogs.com/cnblogs_com/nickong/s60.JPG
- [8] Developing Series 60 Applications, 2001 by Edwards
- [9]<URL http://www.forum.nokia.com/Technology_Topics/Devices_Platforms/s60/
- [10] Mobile Information Device Profile for Java 2 Micro Edition,2001 by Enrique Ortiz and Eric Giguere
- [11]<URL
<http://www.developer.com/img/articles/2002/08/06/coreJ2ME/CoreJ2ME01.jpg>
- [12]<URL
http://developers.sun.com/mobility/configurations/articles/cdc/images/cdc_fig1.gif
- [13]<URL <http://www.calsoftlabs.com/whitopapers/images/midp.gif>
- [14]<URL http://raud.ut.ee/~tec/thesis/MastersThesis_files/image022.gif
- [15]<URL http://onjava.com/pub/a/onjava/excerpt/j2menut_3/index3.html
- [16]S60 3rd Edition SDK Supporting Feature Pack1, for MIDP, User's Guide