CERTIFICATION OF APPROVAL

Wireless Mobile Application for F&B Ordering System

by

Asrul Sani Bin Abd Hamid

A project dissertation submitted to the
Information Technology Programme
Universiti Teknologi PETRONAS
in partial fulfillment of the requirement for the
BACHELOR OF TECHNOLOGY (Hons)
(INFORMATION TECHNOLOGY)

Approved by,

[illegible]

Anang Hudaya b Muhamad Amin
(FYP Supervisor)

t
QA
76.73
.J38
A798
2005

UNIVERSITI TEKNOLOGI PETRONAS
TRONOH, PERAK
MEI 2005

1. Java (Computer program language)
2. Wireless

CERTIFICATION OF ORIGINALITY

This is to certify that I am responsible for the work submitted in this project, that the originality work is my own except as specified in the references and acknowledgements, and that the original work herein have not been undertaken or done by unspecified sources or persons.

(ASRUL SANI BIN ABD HAMID)

Abstract

The topic for the Final Year Project that has been proposed is "Wireless Mobile Application for Food & Beverages Ordering System". The topic of the project is to build a system that will be developed using Java 2 Micro Edition (J2ME). Currently, most of today's applications running on consumer devices, such as mobile phones, PDAs, and TV set-top boxes, as well as a broad range of embedded devices have the Java 2 Platform, Micro Edition (J2ME) embedded within it. Basically, the sole purpose of the topic chosen is to create an additional application which is the ordering system and installing it in the mobile devices. The main focus of this project is to create a wireless mobile application for ordering food and beverages in a restaurant or a food court. Users can order their food directly from the mobile device at the table directly to the kitchen where a mini server is installed for the cook use. Therefore, the cook can identify their customer order and instantly preparing for them, which makes the process a whole lot quicker. The applications used are devices using Wireless Java 2 Micro Edition via Bluetooth. Nowadays, the use of Bluetooth is growing rapidly and most of Java enabled devices has integration with the Bluetooth technology.

Acknowledgement

The author would like to gives the solemn gratitude to several individuals that have contributed to the development of the project, who help directly or indirectly in order to finish the project. Their contribution and support has really helped the author to manage in overcoming the challenges and difficulties encountered during the phases of the project.

First of all, the author would like to thank his family who has been supporting continuously and giving full encouragement to him. Without their help, and most of all their moral and spiritual support that help the project to conclude as it is.

The second person who I wanted to thank is the author supervisor, Mr Anang Hudaya bin Muhamad Amin, for his guidance and tolerance against my work for the past 5 months of hard work.

And last but not least, the author would like to thank Mr Khairul Zarir and Ms. Afreda Sutina for the help and support that has been given, to teach about J2ME programming language, which if is not for the help given, will give the author the end of the road to the project.

TABLE OF CONTENT

LIST OF FIGURES

ACRONYMS AND ABREVIATIONS

APIs - Application Programming Interface

CDC - Connected Device Configuration

CLDC - Connected Limited Device Configuration

F&B - Food & Beverages

I/O - Input/Output

Java VM - Java Virtual Machine

J2ME - Java 2 Micro Edition

J2EE - Java 2 Enterprise Edition

J2SE - Java 2 Second Edition

JCP - Java Community Process

JDK - Java Development Kit

JRE - J2SE Runtime Environment

MIDP - Mobile Information Device Profile

PDA - Personal Digital Assistant

WYSIWYG - What You See Is What You Get

XML - eXtensible Markup Language

CHAPTER 1: INTRODUCTION

1.1 BACKGROUND OF STUDY

Ordering systems of a typical restaurant nowadays still used the old business process model of taking order manually from the customer and then sending the order to the kitchen to be prepared. This old business system is proven to be out-dated as the process is done manually by the worker. Not to mention the phases needed from ordering to preparing and serving the food to the customers. Today, there is simple solution for this problem. The solution is The Java 2 Platform, Micro Edition (J2ME) .

The Java 2 Platform, Micro Edition (J2ME) provides a robust, flexible environment for applications running on consumer devices, such as mobile phones, PDAs, and TV set-top boxes, as well as a broad range of embedded devices. Like its counterparts for the enterprise (J2EE), desktop (J2SE) and smart card (Java Card) environments, J2ME includes Java virtual machines and a set of standard Java APIs defined through the Java Community Process, by expert groups whose members include leading device manufacturers, software vendors, and service providers.

J2ME delivers the power and benefits of Java technology to consumer and embedded devices. It includes flexible user interfaces, a robust security model, a broad range of built-in network protocols, and extensive support for networked and offline applications that can be downloaded dynamically. Applications based on J2ME specifications are written once for a wide range of devices, yet exploit each device's native capabilities.

The J2ME platform is deployed on millions of devices, supported by leading tool vendors, and used by companies worldwide. In short, it is the platform of choice for today's consumer and embedded devices.

1.2 PROBLEM STATEMENT

The use of Java enabled mobile phone is increasing but the usage of the technology is still not maximized to its full potential. Though the device vendor of the devices includes some application in their product such as games, phonebook and other standard application, there is still room for maximization of the current standard of technology. Today, there are after market software that is sold to customers to upgrade their mobile devices, which more than half of those are consists of J2ME architecture.

1.2.1 Problem Identification

As we all agree, one of the main problem in food and beverages is the amount of time taken to process an order to customer and to prepare it. Many restaurants nowadays are still using the manual way of taking order using pen and notes, and then be given to the cook to prepare the dishes. This process makes it hard to keep track of all the orders and the right queue for the order. There are still humanly mistake to be made by the waitress, e.g. poor writing of the order which may cause the wrong order to be prepared. From the causes of the problems, many of the problems are caused by natural human mistake, so the current system must be improvised so that minimal mistake can be avoided.

1.2.2 Significant of the Project

By computerizing the F&B ordering system, the problems stated above can be removed. This new system will improve the efficiency and reliability compared to the old system used. Nevertheless, it will also improve the service given to the customer to maximize the restaurants profit. It can also increase the usability of the current application from the mobile phone. The system of ordering their food using this technology can also save time by cutting the period of the order getting from the customer to the cook.

1.3 OBJECTIVE & SCOPE STATEMENT

1.3.1 Objectives

- To create a system for a restaurant / food court customer to order their meal using a mobile devices.
- To use the Bluetooth technology as a base for connection between the mobile devices.

1.3.2 Scope of Study

The scope of the study is a wide range of Java enabled PDA that can be used to interact between customer and kitchen. The connection of the devices is via Bluetooth which are more applicable rather than infra red. The system should substitute the current customer-waiter-cook which gives the customer to interact directly to the kitchen to order their food.

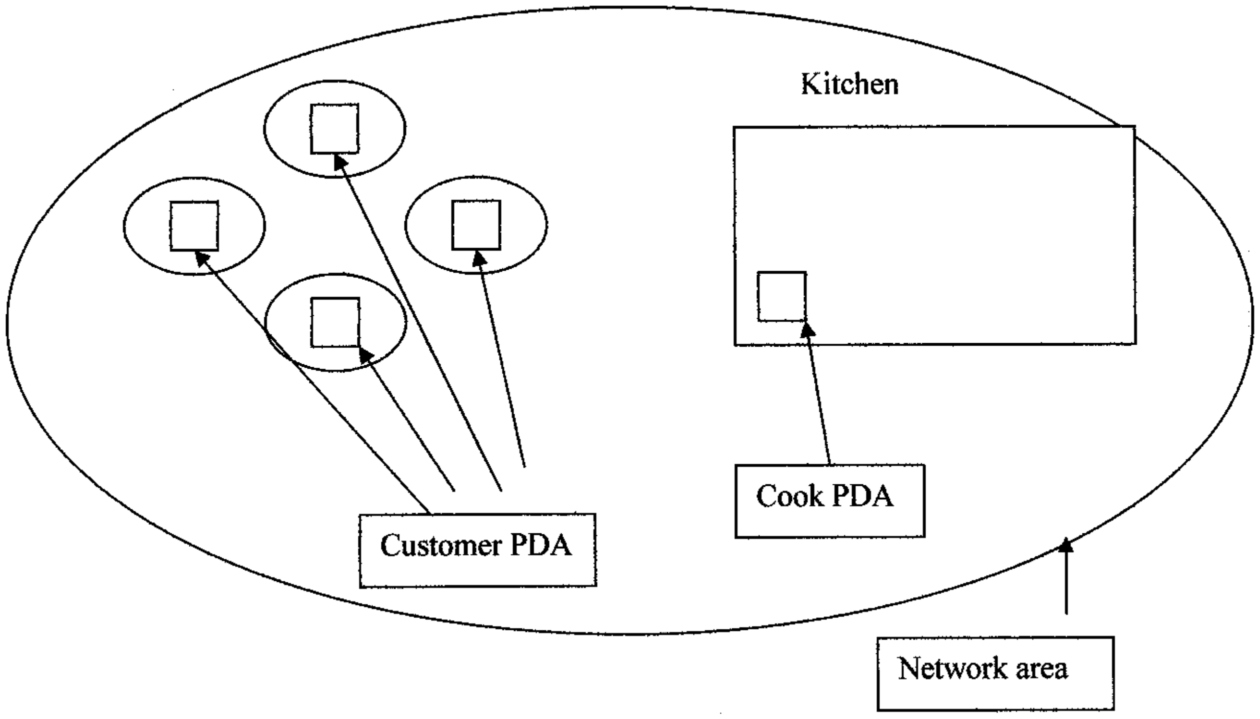


Figure 1.1 : Overview of the project

1.4 RELEVANCY OF THE PROJECT

In the context of the topic, the full usability of today's technology in small handheld devices are still small in numbers and still not maximized to the maximum limit. There are so little companies that are still skeptical of pursuing this technology, because of the lack of introduction to the true capability of the J2ME programming in terms of mobility to the fullest. Thus, this project will give the business environment the impact of mobile application and how it can change their perspective of the benefits and capabilities of wireless technology.

1.5 FEASIBILITY OF THE PROJECT

The proposed project will be given a duration of time frame at about 14 weeks, which the allocated time frame covers the need to develop a prototype of the system according to the system requirements. The management of time used to develop the project is crucial so that every aspect of the project area is covered and the work flow should be consistence with the duration of the entire project. The progress of the project is to be monitored so that the system can be delivered on time.

CHAPTER 2: LITERATURE REVIEW

At this early stage, the project will basically look into the research of the utilization of J2ME system simulation. A thorough literature review would be done through reference books, internet and journals to further understand in order to determine the development procedure and to learn the basic process that is used to develop the project.

Simplicity™ Version 3.0 for Palm OS® makes it easy to build great applications using Java technology that will run on palmOne™ devices, Treo™ smartphones, and many other devices that use the Palm OS® Platform. – ***Data Representation Inc.***

The JavaTM 2 Platform, Micro Edition (J2METM) is the Java platform for consumer and embedded devices such as mobile phones, PDAs, TV set-top boxes, in-vehicle telematics systems, and a broad range of embedded devices – ***Sun Microsystems***

Imagine an enterprise solution that allows Java developers to have true WYSIWYG integration of front and back-end applications. Dream no more! With Simplicity for Mobile Servers, now you can develop, view, and test Servlets and MIDlets simultaneously. – ***Data Representation Inc.***

Java technology readily harnesses the power of the network because it is both a programming language and a selection of specialized platforms. As such, it standardizes the development and deployment of the kind of secure, portable, reliable, and scalable applications required by the networked economy. Because the Internet and World Wide Web play a major role in new business development, consistent and widely supported standards are critical to growth and success. – Java.sun.com

Java Programming Language

The Java programming language lets you write powerful, enterprise-worthy programs that run in the browser, from the desktop, on a server, or on a consumer device. Java programs are run on -- interpreted by -- another program called the Java Virtual Machine (Java VM). Rather than running directly on the native operating system, the program is interpreted by the Java VM for the native operating system. This means that any computer system with the Java VM installed can run a Java program regardless of the computer system on which the application was originally developed.

Java Platform

The Java platform is a software-only platform that runs on top of other hardware-based platforms. Because hardware-based platforms vary in their storage, memory, network connection, and computing power capabilities, specialized Java platforms are available to address applications development for and deployment to those different environments.

Java technology has grown to include the portfolio of specialized platforms listed below. Each platform is based on a Java VM that has been ported to the target hardware environment. This means, for example, in the case of Desktop Java, desktop applications written in the Java programming language can run on any Java VM-enabled desktop without modification.

<u>Java 2 Platform, Standard Edition (J2SE)</u>

Java 2 Platform, Standard Edition (J2SE), provides an environment for Core Java and Desktop Java applications development, and is the basis for Java 2 Platform, Enterprise Edition (J2EE) and Java Web Services technologies. It has the compiler, tools, runtimes, and Java APIs that let you write, test, deploy, and run applets and applications.

There are two principal products in the J2SE platform family: J2SE Runtime Environment (JRE) and J2SE Development Kit (JDK). The JRE provides the Java APIs, Java virtual machine, and other components necessary to run applets and applications written in the Java programming language. It is also the foundation for the technologies in the Java 2 Platform, Enterprise Edition (J2EE) for enterprise software development and deployment. The JRE does not contain tools and utilities such as compilers or debuggers for developing applets and applications. The JDK is a superset of the JRE, and contains everything that is in the JRE, plus tools such as the compilers and debuggers necessary for developing applets and applications. This conceptual diagram illustrates all the component technologies in J2SE platform and how they fit together.

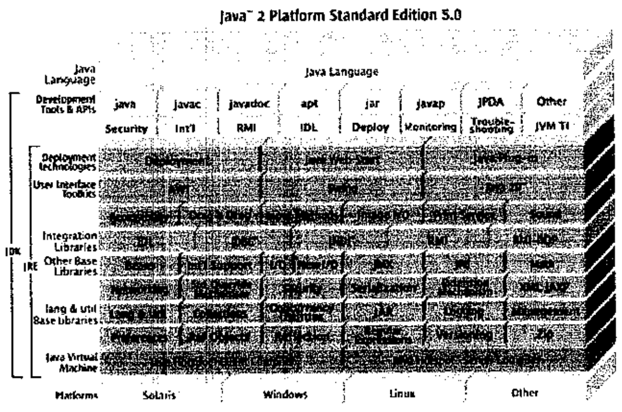


Figure 2.1 : Java™ 2 Standard Edition Platform

Java 2 Platform, Enterprise Edition (J2EE)

Java 2 Platform, Enterprise Edition (J2EE), defines the standard for developing component-based multitier enterprise applications. It is based on J2SE and provides additional services, tools, and APIs to support simplified enterprise applications development.

The Java 2 Platform, Enterprise Edition (J2EE) defines the standard for developing multitier enterprise applications. The J2EE platform simplifies enterprise applications by basing them on standardized, modular components, by providing a complete set of services to those components, and by handling many details of application behavior automatically, without complex programming.

The J2EE platform takes advantage of many features of the Java 2 Platform, Standard Edition (J2SE), such as "Write Once, Run Anywhere" portability, JDBC API for database access, CORBA technology for interaction with existing enterprise resources, and a security model that protects data even in internet applications. Building on this base, the Java 2 Platform, Enterprise Edition adds full support for Enterprise JavaBeans components, Java Servlets API, JavaServer Pages and XML technology. The J2EE standard includes complete specifications and compliance tests to ensure portability of applications across the wide range of existing enterprise systems capable of supporting the J2EE platform. In addition, the J2EE specification now ensures Web services interoperability through support for the WS-I Basic Profile.

Enterprise Application Model

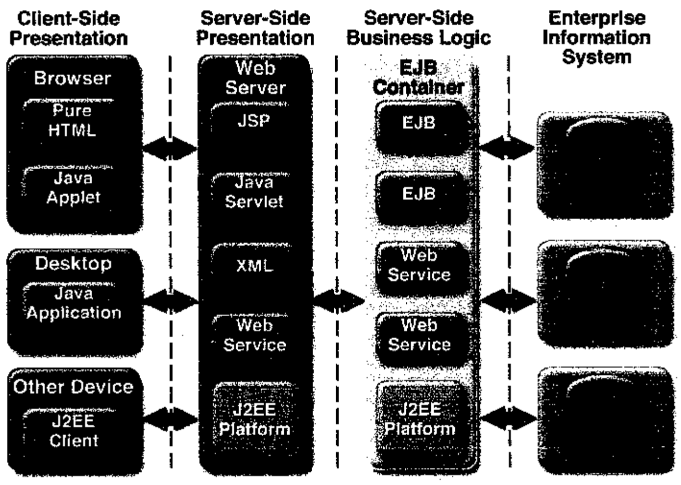


Figure 2.2 : Java^{TM} 2 Enterprise Edition Platform

The Enterprise Java BluePrints for the J2EE platform describe the J2EE application model and best practices for using the J2EE platform. Building on the J2SE platform, the J2EE application model provides a simplified approach to developing highly scalable and highly available internet or intranet based applications.

Java 2 Platform, Micro Edition (J2ME)

Java 2 Platform, Micro Edition (J2ME), is a set of technologies and specifications targeted at consumer and embedded devices, such as mobile phones, personal digital assistants (PDA's), printers, and TV set-top boxes.
The Java 2 Platform, Micro Edition (J2ME) provides a robust, flexible environment for applications running on consumer devices, such as mobile phones, PDAs, and TV set-top boxes, as well as a broad range of embedded devices. Like its counterparts for the enterprise (J2EE), desktop (J2SE) and smart card (Java Card) environments, J2ME includes Java virtual machines and a set of standard Java APIs defined through the Java Community Process, by expert groups whose members include leading device manufacturers, software vendors, and service providers.

The J2ME Architecture

The J2ME architecture comprises a variety of configurations, profiles, and optional packages that implementers and developers can choose from, and combine to construct a complete Java runtime environment that closely fits the requirements of a particular range of devices and a target market. Each combination is optimized for the memory, processing power, and I/O capabilities of a related category of devices. The result is a common Java platform that takes full advantage of each type of device to deliver a rich user experience.

Configurations

Configurations comprise a virtual machine and a minimal set of class libraries. They provide the base functionality for a particular range of devices that share similar characteristics, such as network connectivity and memory footprint. Currently, there are two J2ME configurations: the Connected Limited Device Configuration (CLDC) and the Connected Device Configuration (CDC).

Optional Packages

The J2ME platform can be extended by adding various optional packages to a technology stack that includes either CLDC or CDC and an associated profile. Created to address very specific application requirements, optional packages offer standard APIs for using both existing and emerging technologies such as database connectivity, wireless messaging, multimedia, Bluetooth, and web services. Because optional packages are modular, developers can avoid carrying the overhead of unnecessary functionality by including only the packages and application actually needs.

J2ME delivers the power and benefits of Java technology to consumer and embedded devices. It includes flexible user interfaces, a robust security model, a broad range of built-in network protocols, and extensive support for networked and offline applications that can be downloaded dynamically. Applications based on J2ME specifications are written once for a wide range of devices, yet exploit each device's native capabilities.

The J2ME platform is deployed on millions of devices, supported by leading tool vendors, and used by companies worldwide. In short, it is the platform of choice for today's consumer and embedded devices.

Introduction to Connected Limited Device Configuration (CLDC)

The Connected Limited Device Configuration (CLDC) is a fundamental part of the architecture of the Java 2 Platform, Micro Edition (J2ME). J2ME technology is delivered in API bundles called *configurations*, *profiles*, and *optional packages*. A J2ME application environment includes both a configuration like CLDC and a profile like the Mobile Information Device Profile (MIDP). In addition, optional packages provide capability in specific areas of functionality, such as wireless messaging and multimedia capture and playback. The ability to choose from among the various bundles enables product designers and developers to match software capabilities with hardware capabilities very closely. They can use APIs that give them easy access to the components a particular kind of device actually has, without the overhead of APIs designed for capabilities the device doesn't support.

The CLDC Configuration

A *configuration* provides the most basic set of libraries and virtual-machine features that must be present in each implementation of a J2ME environment. When coupled with one or more profiles, the Connected Limited Device Configuration gives developers a solid Java platform for creating applications for consumer and embedded devices.

Introduction to Mobile Information Device Profile (MIDP)

The Mobile Information Device Profile (MIDP) is a key element of the Java 2 Platform, Mobile Edition (J2ME). When combined with the Connected Limited Device Configuration (CLDC), MIDP provides a standard Java runtime environment for today's most popular mobile information devices, such as cell phones and mainstream personal digital assistants (PDAs). The MIDP specification was defined through the Java Community Process (JCP) by an expert group of more than 50 companies, including leading device manufacturers, wireless carriers, and vendors of mobile software. It defines a platform for dynamically and securely deploying optimized, graphical, networked applications.

CLDC and MIDP provide the core application functionality required by mobile applications, in the form of a standardized Java runtime environment and a rich set of Java APIs. Developers using MIDP can write applications once, and then deploy them quickly to a wide variety of mobile information devices. MIDP has been widely adopted as the platform of choice for mobile applications. It is deployed globally on millions of phones and PDAs, and is supported by leading integrated development environments (IDEs). Companies around the world have already taken advantage of MIDP to write a broad range of consumer and enterprise mobile applications.

Specifications

- **MIDP 2.0** (JSR 118) is a revised version of the MIDP 1.0 specification. New features include an enhanced user interface, multimedia and game functionality, more extensive connectivity, over-the-air provisioning (OTA), and end-to-end security. MIDP 2.0 is backward-compatible with MIDP 1.0, and continues to target mobile information devices like mobile phones and PDAs.

- **MIDP 1.0** (JSR 37) is the original specification, which provides core application functionality required by mobile applications, including basic user interface and network security.

Java Card

Java Card technology enables smart cards and other devices with very limited memory to run small applications, called applets that employ Java technology. It provides smart card manufacturers with a secure and interoperable execution platform that can store and update multiple applications on a single device. Java Card technology is compatible with existing smart card standards.

The technology enables developers to build, test, and deploy applications and services rapidly and securely. This accelerated process reduces development costs, increases product differentiation, and enhances value to customers. In a manner complementary to the Standard, Enterprise, and Mobile editions of the Java 2 Platform, Java Card technology makes it easy to integrate security tokens into a complete Java software solution.

Benefits

Smart card vendors and issuers benefit from several unique features of Java Card technology, which is:

Interoperable: Applets developed with Java Card technology will run on any Java Card technology-enabled smart card, independently of the card vendor and underlying hardware.

Secure: Java Card technology relies on the inherent security of the Java programming language to provide a secure execution environment. Designed through an open process, the platform's proven industry deployments and security evaluations ensure that card issuers benefit from the most capable and secure technology available today.

Multi-Application-Capable: Java Card technology enables multiple applications to co-exist securely on a single smart card.
Dynamic: New applications can be installed securely after a card has been issued, enabling card issuers to respond to their customer's changing needs dynamically.
Compatible with Existing Standards: The Java Card API is compatible with international standards for smart cards such as ISO7816, or EMV. Major industry-specific standards such as Global Platform and ETSI refer to it.
Developers creating Java Card applications enjoy all the advantages of working in the Java programming language:

- Object-oriented programming yields greater code modularity and reusability, leading to higher programmer productivity.
- Protection features characteristic of the Java programming language apply to Java Card applets, enforcing strong typing and protection attributes.
- Powerful off-the-shelf development tools are readily available.

Components
Sun Microsystems publishes the Java Card Platform Specification and the Java Card Development Kit, which includes a reference implementation based on the specification. Providing the basis for cross-platform and cross-vendor applet interoperability, version 2.2.1 of the specification includes three documents:

- **The Java Card Virtual Machine Specification** defines the features, services, and behavior that an implementation of the Java Card technology must support. It includes the instruction set of a Java Card Virtual Machine (VM), the supported subset of the Java language, and the file formats used to install applets and libraries into smart cards and other devices that host Java Card technology.

- **The Java Card Runtime Environment Specification** defines the necessary behavior of the runtime environment (RE) in any implementation of the Java Card technology, which must include implementations of the Java Card Virtual Machine, the Java Card API classes, and runtime support services such as the selection and de-selection of applets.
- **API for the Java Card Platform** complements the Java Card Runtime Environment Specification, and describes the application programming interface of the Java Card technology. The API is compatible with formal international standards and industry-specific standards. It contains the class definitions required to support the Java Card VM and the Java Card RE.

The Java Card Development Kit is a suite of tools for designing implementations of the Java Card technology, and for developing applets based on the Java Card API Specification:

- **C-JCRE** is a reference implementation of the Java Card Runtime Environment written in the C programming language. It also includes the Java Card Virtual Machine interpreter.
- Off-card platform components such as the **Java Card Converter** and the **Java Card Verifier** complement C-JCRE, to provide a complete development chain.
- Additional design and testing tools enable developers to prototype and test applications.

Java Card "S"

The Java Card "S" program enables Java Card licensees to derive fixed-function smart cards from existing Java Card technology-based products. Java Card "S" products have all the functionality and security of standard Java Card smart cards except the dynamic download capability: applications cannot be added or removed after the device has been issued. Java Card "S" products bring the value proposition of Java Card technology to an extended range of smart card products. Card issuers can capitalize on the wealth of applications already developed and certified for the Java Card platform, in smart card products with minimal memory capacity. They can now purchase cards with a wide range of prices and capabilities, while still reaping the benefits of Java Card technology, and using Java Card applets that have already proven their worth. This option dramatically reduces the cost and complexity of application development, functional testing, security evaluation, and application life-cycle management.

CHAPTER 3: METHODOLOGY

3.1 PROCEDURE IDENTIFICATION

3.1.1 Problem Definition and Analysis

The J2ME architecture comprises a variety of configurations, profiles, and optional packages that implementers and developers can choose from, and combine to construct a complete Java runtime environment that closely fits the requirements of a particular range of devices and a target market. Each combination is optimized for the memory, processing power, and I/O capabilities of a related category of devices. The result is a common Java platform that takes full advantage of each type of device to deliver a rich user experience.

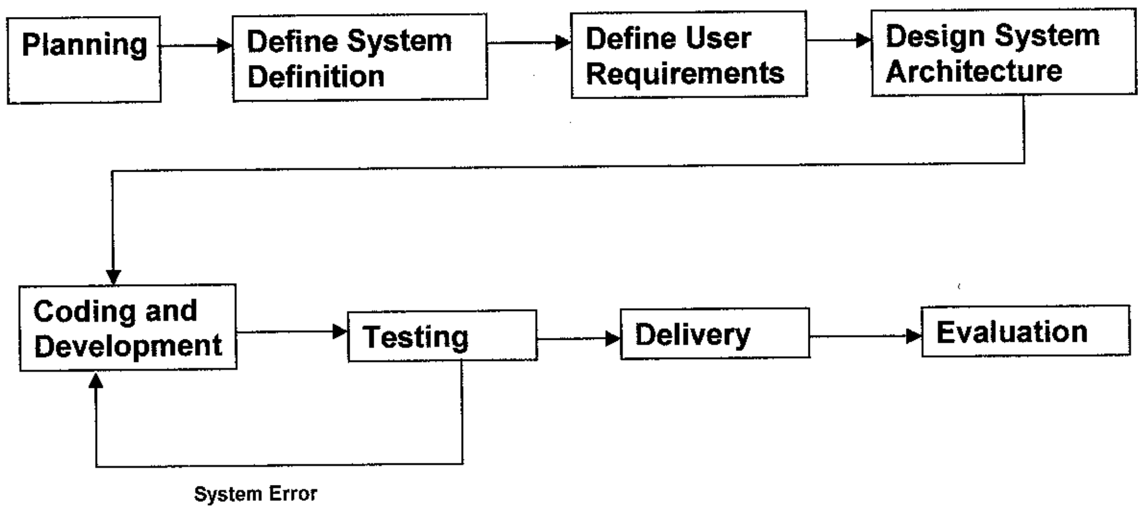


Figure 3.1 : Methodology – Hybrid Model

3.2 PROCEDURE DESCRIPTION

The methodology used in this specific project is the incremental development process model, which is a hybrid development model which support different approach to software development whereby the creator of the system can change the work flows to suit any alteration to the change in the system requirements.

3.2.1 Planning

The first process of developing the project is with a simple planning of the flow of the system. By doing the planning of the project, it allows the author the chance to define any problem statement, design the layout of the system and define the initial scope of the project. In this phase, the project sources are also sorted out to ease the author of where to find the right information regarding the outline definition of the project.
In this planning phase, the author also can identify the tasks and procedures that has to be done and create a draft of Gantt chart to help to cope with the time frame given.

3.2.2 Define System Definition

In this phase, the lay out of the flow of the project are designed to give the author the rough sketch of the layout design. This phase will also be helpful in order to define any user requirements in the next phase, whereby the author can balance the requirements and the capabilities of the project. By doing system definition also the author can analyze any problems that may be encountered during the process of designing the system. After analyzing any problems that may be encountered and fixing the problems, the next phase is to proceed to define the requirements of the system.

3.2.3 Define User Requirements

After the author finished with the definition phase, then the next phase, which is the user requirements definition, is initiated. In this phase, the author will be asking the user, which is the supervisor, of the expectation that he/she may desire within the project. In this phase, there should have been a detailed structure of the system and all the function must fulfill the basic requirement of the project.

3.2.4 Design System Architecture

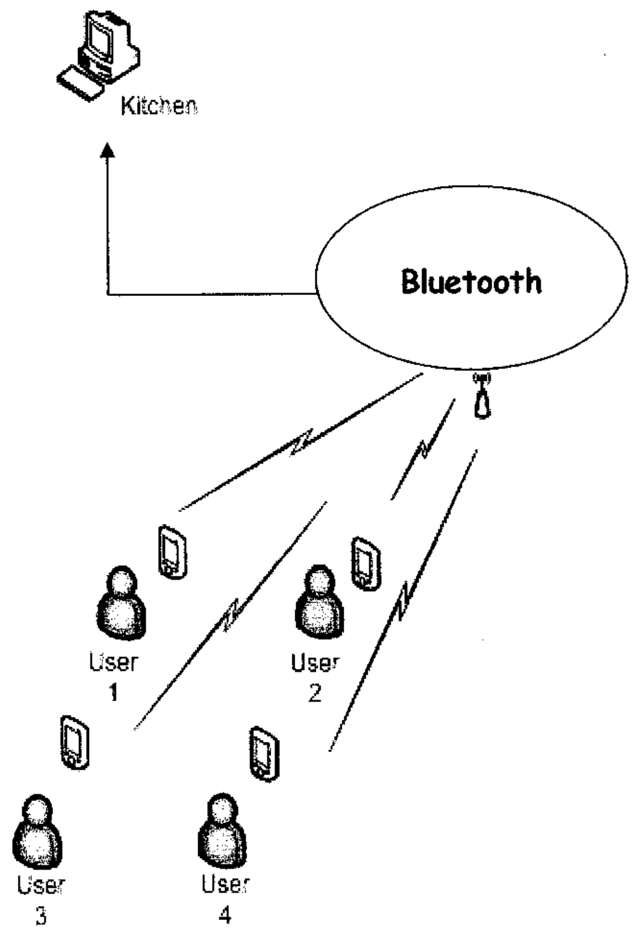


Figure 3.2 : System Architecture of the System

This is basically the system architecture design of the system. Users are seated in tables which have a PDA which is used to order their food. Then the users are connected to the kitchen terminal through a wireless network. The manager terminals are also connected to the wireless network which is connected to the router which routes the order from. All the PDAs are preconfigured to connect to the wireless LAN and will load the F&B ordering system.

3.2.5 Coding and Development

This phase is the core of the development of the project, which will take the longest time to finish, and will ensure that the author move to the next phase. In this phase, the author must finish the coding and development before testing for any errors in the coding.
The purpose of the development phase is to build a system that fulfills the system requirements and design specifications. Among tasks that have to be done in this phase is the development of the user interface and the connection via Bluetooth.

3.2.6 Testing

Testing the system to ensure the functionality of the system meets is specification. This testing phase is usually implemented in a testing area whereby any errors can be detected. After the testing phase and the system satisfy the need for the application, then the system should be implemented in the real world and any errors of the system cannot be altered anymore. If there should be any errors occurring, then the process will reverse back to the previous phase, which is the coding and development process.

3.2.7 Delivery

The delivery phase is done when there are no errors occurring in the testing phase anymore. In this phase, the implementation in a controlled environment is done to test whether the capabilities of the system are similar of the required by the user.

3.2.8 Evaluation

This is the final phase where the system will be evaluated by the users and examiners to check for the progress and the fulfillment of the early user requirements given to the author. Apart from quality aspect, the practicability is conformed to check whether the system produces are suited to substitute with the system currently used.

UML Use Case Diagram

From the figure, there are three actors that play a major role in the system application, which is the customer, chef and manager. Customers can choose from various food that is applicable in the menu, and also the beverages and desert that they wish to order. They also have the capabilities to submit order directly to the chef. Apart from that, the can also alter their order if any mistake are done and cancel any order that they wish to. And after they finish their meal, they can use the system to calculate the bill and pay them directly to the manager. The chef only has the authority to retrieve the order and then update their order queue. Then they will subsequently prepare the order of the customer and serve them. The manager has the total responsibility of updating menu of the food, beverages and dessert in the system. They can also delete any products that have been erased from the menu. And the manager also plays the role of the cashier where they are responsible to acquire the bills from the customer.

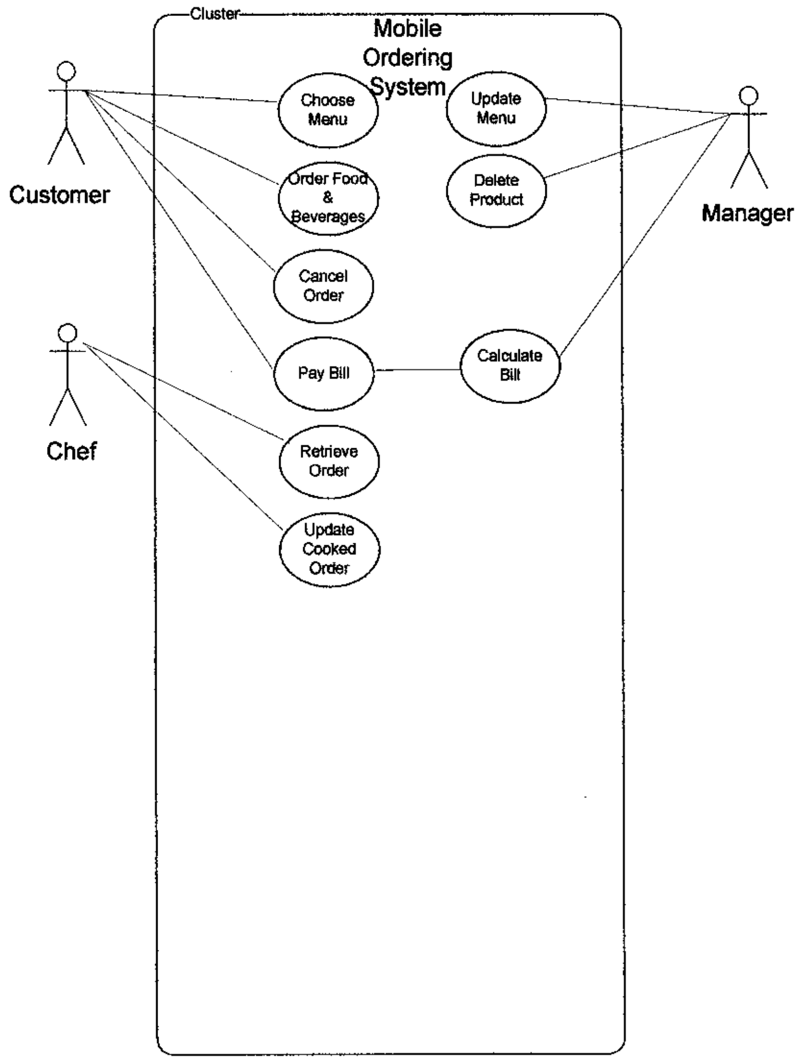


Figure 3.3 : UML Use Case Diagram

3.3 TOOLS REQUIRED

There are two main categories that are needed in other to build and run the system. Those categories are the; the hardware and the software. These tools are also required to simulate the application of the system to give the user the idea of how the system operates.

Hardware

- PDA with Java enabled features
- Complete computer with Bluetooth technology
  - Pentium 4 2.8 Ghz
  - 512 MB DDRAM
  - At least 10GB of hard disk space for operating system
  - 10/100 MBps network card
  - Bluetooth Card
- Bluetooth device (mobile phones, PDAs, hands free, etc.)

Software

- Microsoft Windows XP Professional Edition
- Java j2sdk1.4.2_03 development kit
- J2ME Wireless Toolkit (Ktoolbar)
- Microsoft Visual Studio.net

CHAPTER 4: RESULT AND DISCUSSION

4.1 RESULTS

Below are some of the screenshots of the system. The system has been developed using the layout design from the design processes and fulfilling the user requirements of an F&B ordering system. The flow process of the system is shown in the print screen of the application and all the behaviors are informed of what each picture represents and the manual of the application page.

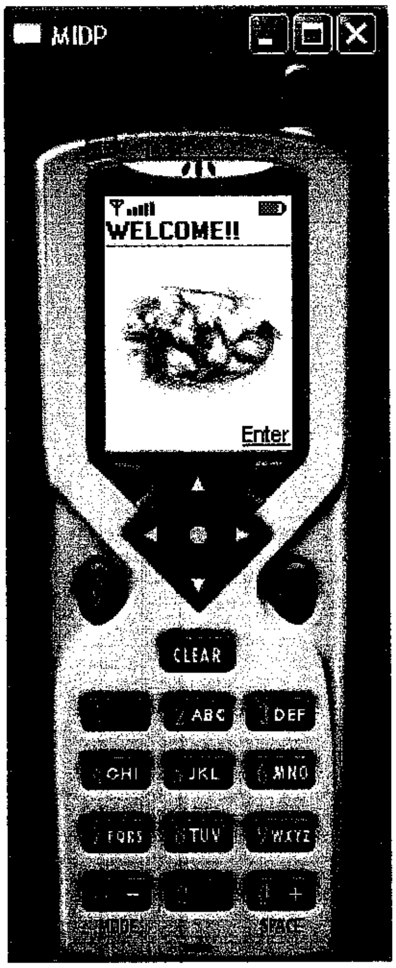


Figure 4.1 : F&B Ordering System - Intro Page

This is the screenshot of the first interface of the system. It is basically a picture of a food with the complimentary MIDlet named "WELCOME". This page purpose is to welcome the users to the system, and the user will enter the system using the button "Enter" situated in the right bottom corner of the screen.

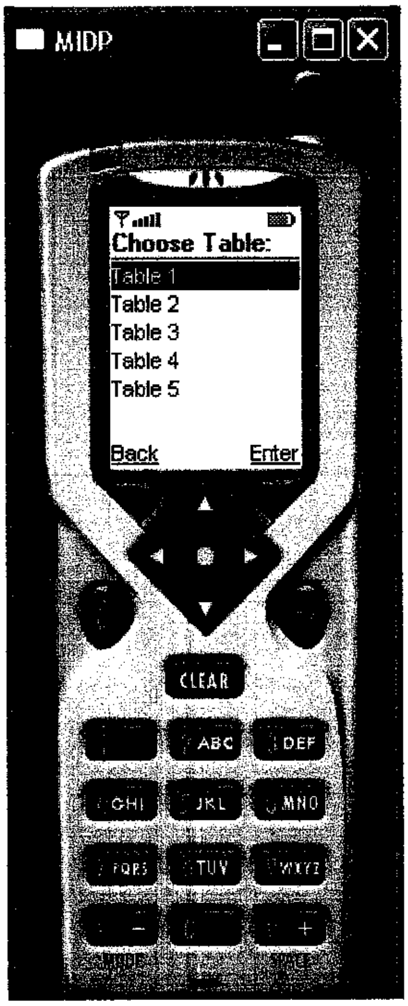


Figure 4.2 : F&B Ordering System - Table Number Page

In figure 4.2, the interface will ask the user to choose the table number where they are located. The user will enter the number and press the "Enter" button to proceed to the next page, or just click the "Back" button to return to the Welcome page.

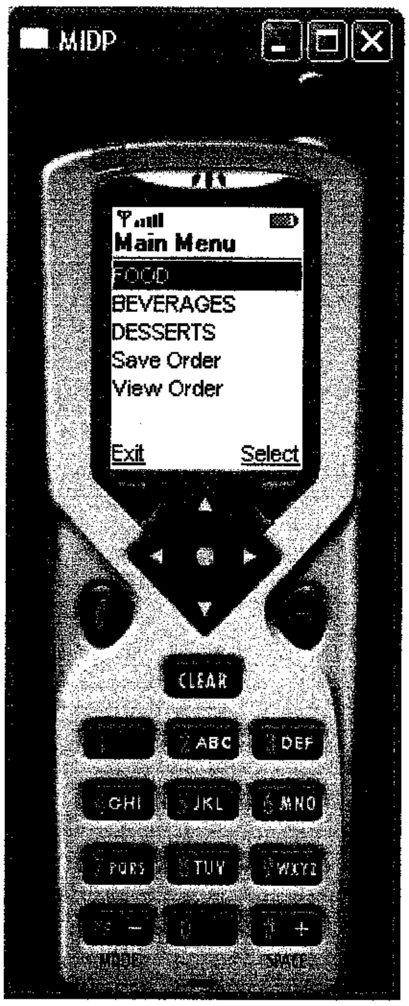


Figure 4.3 : F&B Ordering System - MainMenu Page

This is the main page of the system, whereby the user will start ordering their order for the day which is being offered by the restaurant. There are three sub-menu from this page, the “Food”, “Beverages” and “Deserts”. If the user choose one of any from the list, they will be taken to a page of they desire.

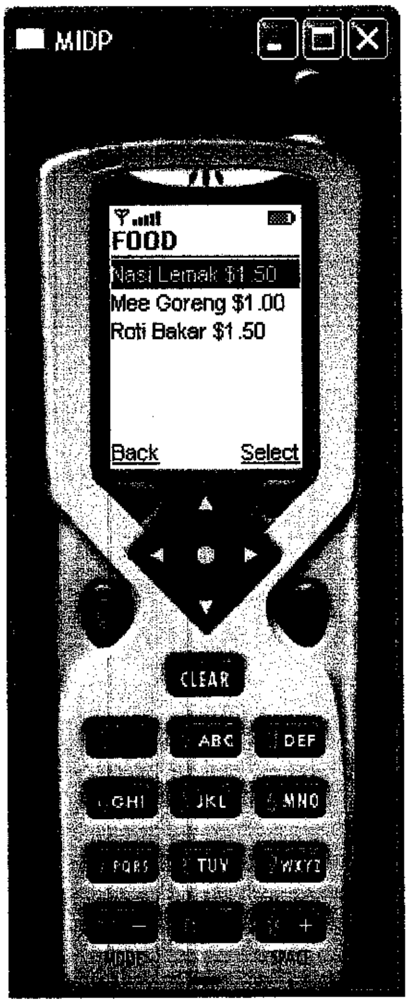


Figure 4.4 : F&B Ordering System - MainFood Page

In Figure 4.4 shows the page when the users chooses the "Food" submenu, where below the Food title there are several list of order that are offered within the vicinity. The users can select the "Back" or "Select" button according to their own preferences.

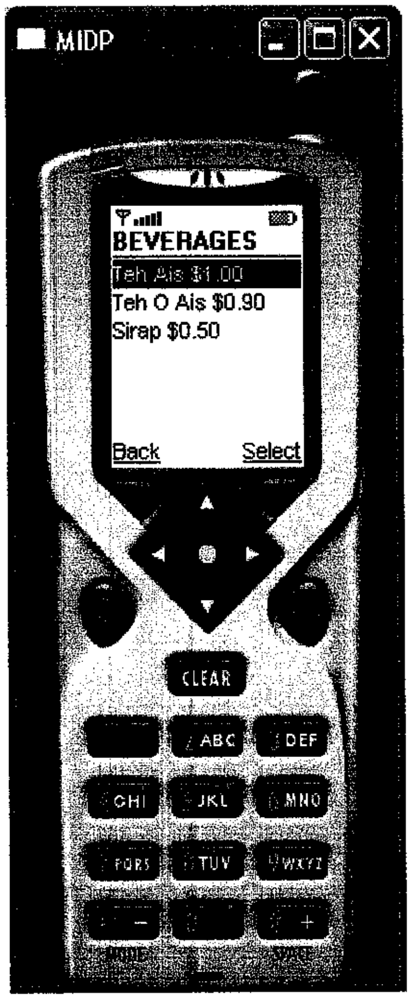


Figure 4.5 : F&B Ordering System - MainBeverages Page

In the next figure, it shows the page when the users chooses the "Beverage" submenu, where below the Beverage title takes place and shown all the beverages to choose from. The users can select the "Back" or "Select" button according to their own preferences.

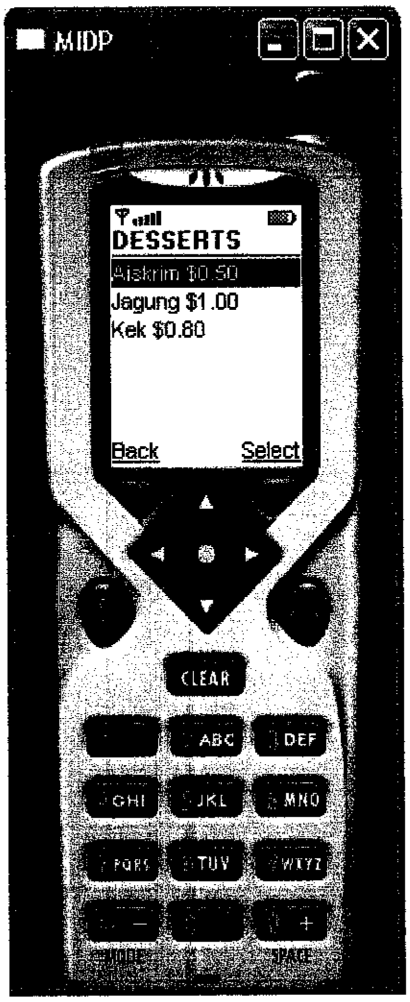


Figure 4.6 : F&B Ordering System - MainDesserts Page

In Figure 4.6, the similar action is done but the user is shown the "Dessert" menu and listed are all the dessert that are available in the vicinity. The users can select the "Back" or "Select" button according to their own preferences.

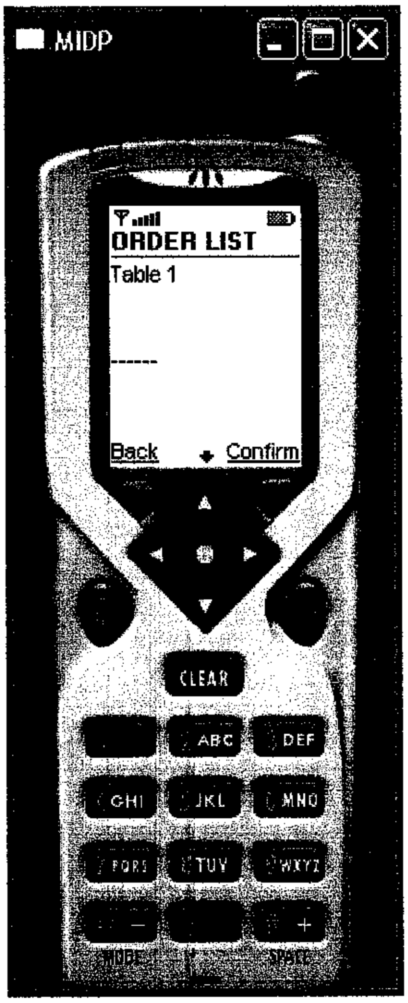
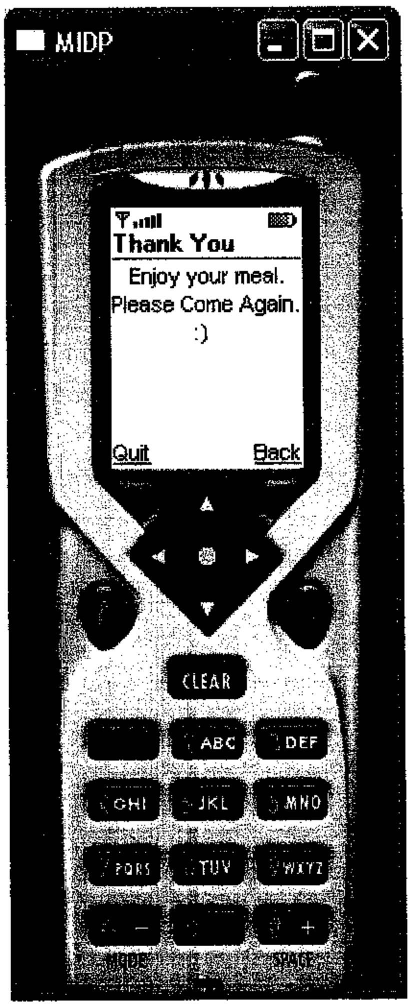


Figure 4.7 : F&B Ordering System - OrderList Page

When the users choose from the MainMenu page, to show the orderlist that they have chosen, then they will be directed to the OrderList page and all the orders that have been selected by the user will be listed in an according manner. The user will have the capability to choose "Confirm" if they are satisfied with their orders, or choose "Back" to edit their orders.



Thank You

Enjoy your meal.
Please Come Again.
:)

Quit Back

Figure 4.8 : F&B Ordering System - Thank Page

Finally, when the user select "Confirm" in the OrderList page, they will be directed to a thank page to thank the user for using the system and hope that they will have a good meal. This process is shown in Figure 4.8.

4.2 DISCUSSION

There are several constraints that had occurred during the development of the system. Below are some of the discussion of the problems and recommendation of the solution that can help solve the entire problem.

4.2.1 Limited Screen Size

The limited screen size of the PDA has caused the author has to amount the items that can be shown in every single menu. The system has been programmed to show several menu items and there are also limitations of every order taken from the customer, whereby the customer can not order more than the maximum order that has been set up.

4.2.2 Connecting via Bluetooth

The connection that has been set up for the system is via Bluetooth connection. This type of connection gives better wireless network because the devices do not have to be parallel with each other rather than using infra red. During the integration of the system, I have encountered several difficulties in setting up the connection, which I have consulted with my supervisor. These problems are caused by the complexity of the programming language to integrate with the Bluetooth system.

CHAPTER 5: CONCLUSION AND RECOMMENDATION

5.1 CONCLUSION

This project needs a complete understanding in Java 2 Micro Edition (J2ME) in order to create a running application in a device. Basically this project consists of both study and research only; and implementing the outcome from the project to create a working prototype. The expected results are a fully functional application that can be fully utilized with any today's standard of Java enabled devices. Upon completing the project, the student will fully understand the power of J2ME and the future power that it possesses. Hopefully the knowledge obtained from this project will be useful in the future in technological environment.

5.2 RECOMMENDATION

The recommendation for this current topic is to upgrade is usage by adding a database interface whereby the manager can update the menu by just adding it in the screen and updating it to the server located in the kitchen. This is very useful whenever there is replacement of the chef, so the cooking specialties will also be changed.

Apart from upgrading it with the database, the proposed recommendation is to let the user view the current queue of all the orders for the current time and estimation of how long it will take for their food to be prepared. But to do this will need the synchronization of the kitchen (mainly the chef as the human labor) with the system, which will be hard to manage, but still possible especially for fast food service or food court outlet.

CHAPTER 6: REFERENCES

http://java.sun.com/j2me

http://developers.sun.com/

James Keogh, "The Complete Reference J2ME" (McGraw Hill and Osborne)

Martin J Wells, "J2ME^{TM} GAME PROGRAMMING" (Premier Press)

Cary A. Jardin,"Java^{TM} Electronic Commerce" (Wiley Computer Publishing)

Bluetooth The Basics (Tech Publications PTE Ltd)

Bluetooth Demystified (McGraw Hill)

C Bala Kumar, Paul J. Kline, Timothy J. Thompson ," Bluetooth Application Programming with the JAVA APIs" (Morgan Kauffman)

Appendix

Compilation Logs with Screen Shot.
In order to compile the coding that have been created, there are several step that must be followed in order to show the end result of the program. There are several tools that are required to get started. The table below shows the basic tools needed to develop a fully running J2ME application.

Tool	Version	URL
JDK (Java Development Kit)	Version 1.4.2	http://java.sun.com.products/j2se
MIDP(mobile information Devices Profile)	Version 1.0.3	http://java.sun.com/product/midp
Bluetooth Dongle		

Installing the software

All the software listed above can be downloaded from the url given. After you've obtain the software, you can start by installing the software required. The steps in order to build the program will be shown in a manner to ensure that the user will understand the development system of the application.

The first step is to create a directory which will contain all the files that you will create for the project. This folder will be used through out the development of the project. The directory that can be used, for example is :

C:\j2me

(You can choose another drive or path it suits you). From now on, I'll refer to this as the "j2me home".
Next you need to set up the Java Development environment using JDK before you move on to the MIDP.

Setting up the JDK

Setting the MIDP

Next step you need to install the MIDP. To do this, use your favorite compression tool to unpack the files into subdirectories of the j2me home. You should end up directory something like c:\j2me\midp1.0.3fcs.

After you have unpacked the files, you will have a number of directories containing all the components of MIDP package. The folder root will look similar to the diagram below:

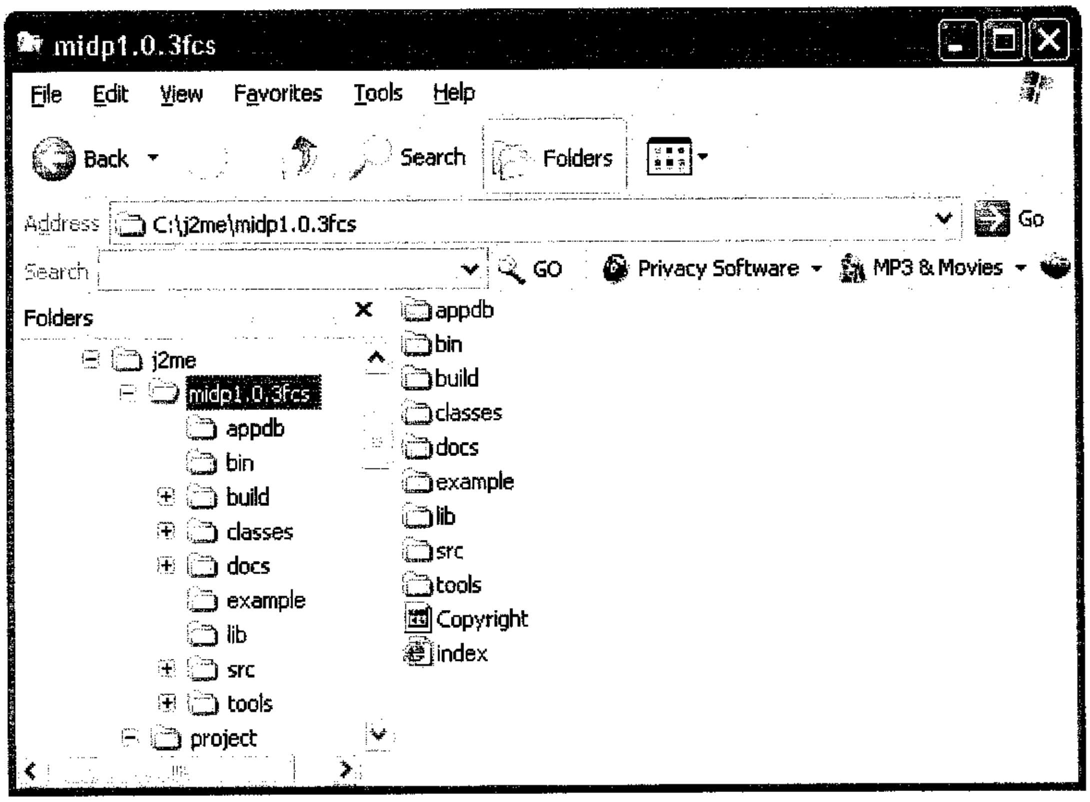


Figure 7.1

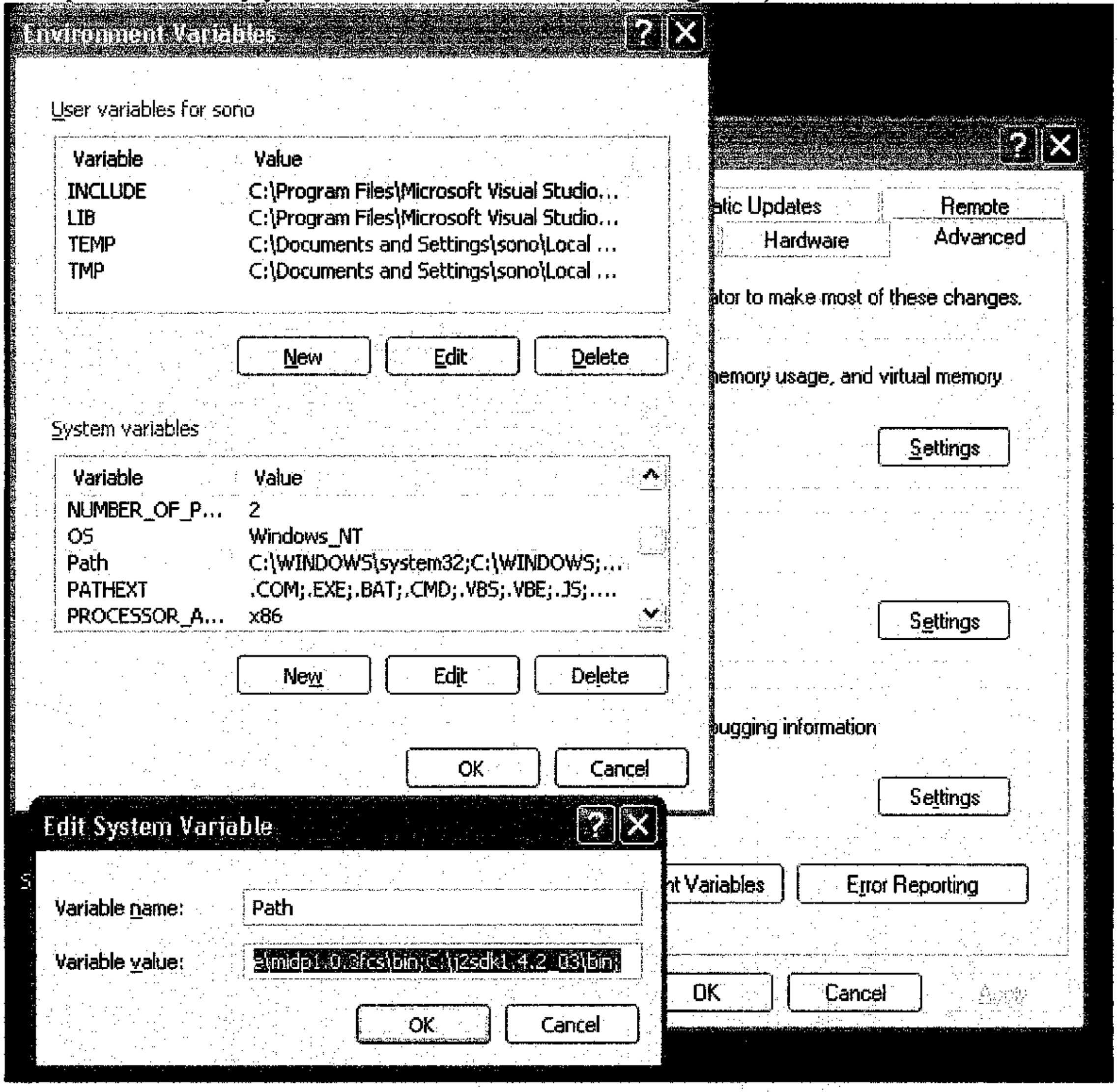
After unpacking the MIDP files you should have a directory structure matching this.

MIDP Directories	
Directory	Description
\bin	Contains the command line tools, preverify.exe, and the midp.exe emulator
\classes	Contains MIDP classes; you will compile using these as a base.
\docs	Contains comprehensive documentation of the MIDP. Includes guides, reference materials and release notes.
\example	Contains example JARs and JaDs for demonstration purposes.
\lib	Contains configuration files
\src	Contains example source code.

Making path changes

The purpose of using this PATH environment variable is allowing the program to locate the java environment variables when executing the programs or libraries. A common PATH is to a specify directories that contains programs you want to execute, without having to navigate to other directory that contains it.

How to edit the path depends on your operating system version. Here I give example on WINDOWS XP. Firstly, use the control panel to open the system panel, and then select the option to modify your environment variables (see figure 4)


Environment Variables

User variables for sono

Variable	Value
INCLUDE	C:\Program Files\Microsoft Visual Studio...
LIB	C:\Program Files\Microsoft Visual Studio...
TEMP	C:\Documents and Settings\sono\Local ...
TMP	C:\Documents and Settings\sono\Local ...

System variables

Variable	Value
NUMBER_OF_P...	2
OS	Windows_NT
Path	C:\WINDOWS\system32;C:\WINDOWS;...
PATHEXT	.COM;.EXE;.BAT;.CMD;.VBS;.VBE;.JS;....
PROCESSOR_A...	x86

Edit System Variable

Variable name: Path

Variable value: 3\midp1.0.3fcs\bin;C:\j2sdk1.4.2_03\bin;

Figure 7.2

Edit the PATH system variable to add the MIDP executable path.

Select Path variable and hit Edit then type in all of your executable directories on to the end of your path, proceeding by a semicolon:

;c\:j2me\midp1.0.3fcs\bin;

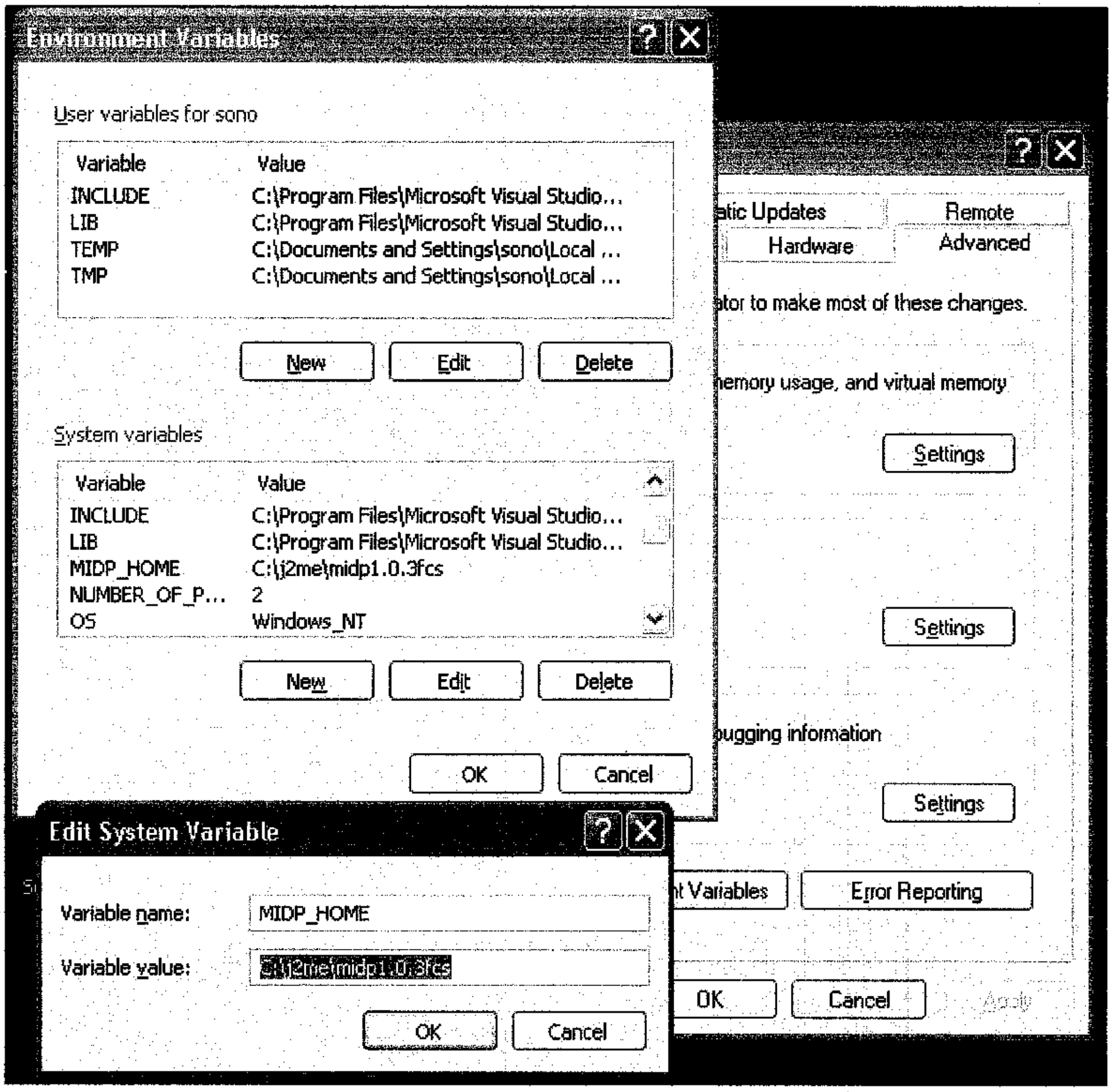
Finally, you should also check and add the JDK binary directory to your path if it isn't already there. For example:

;c:\j2sdk1.4.1_02\bin;

When you've finished editing the PATH and saving the new settings, you should open a new shell. This will cause the new path take effect.

Setting MIDP_HOME

The MIDP executable requires you to set an environment variable named MIDP_HOME for the executable to function correctly. Setting the variable is quite similar to adjusting the PATH. However, instead of editing the existing variable, you should create a new one. To do this, open the Environment Variables panel again, and then hit the NEW button and fill the fields as shown in figure 5



Environment Variables

User variables for sono

Variable	Value
INCLUDE	C:\Program Files\Microsoft Visual Studio...
LIB	C:\Program Files\Microsoft Visual Studio...
TEMP	C:\Documents and Settings\sono\Local ...
TMP	C:\Documents and Settings\sono\Local ...

System variables

Variable	Value
INCLUDE	C:\Program Files\Microsoft Visual Studio...
LIB	C:\Program Files\Microsoft Visual Studio...
MIDP_HOME	C:\j2me\midp1.0.3fcs
NUMBER_OF_P...	2
OS	Windows_NT

Edit System Variable

Variable name: MIDP_HOME

Variable value: C:\j2me\midp1.0.3fcs

Figure 7.3

Add a new environment named MIDP_HOME with the value corresponding to where you installed the MIDP RI.

Updating CLASSPATH

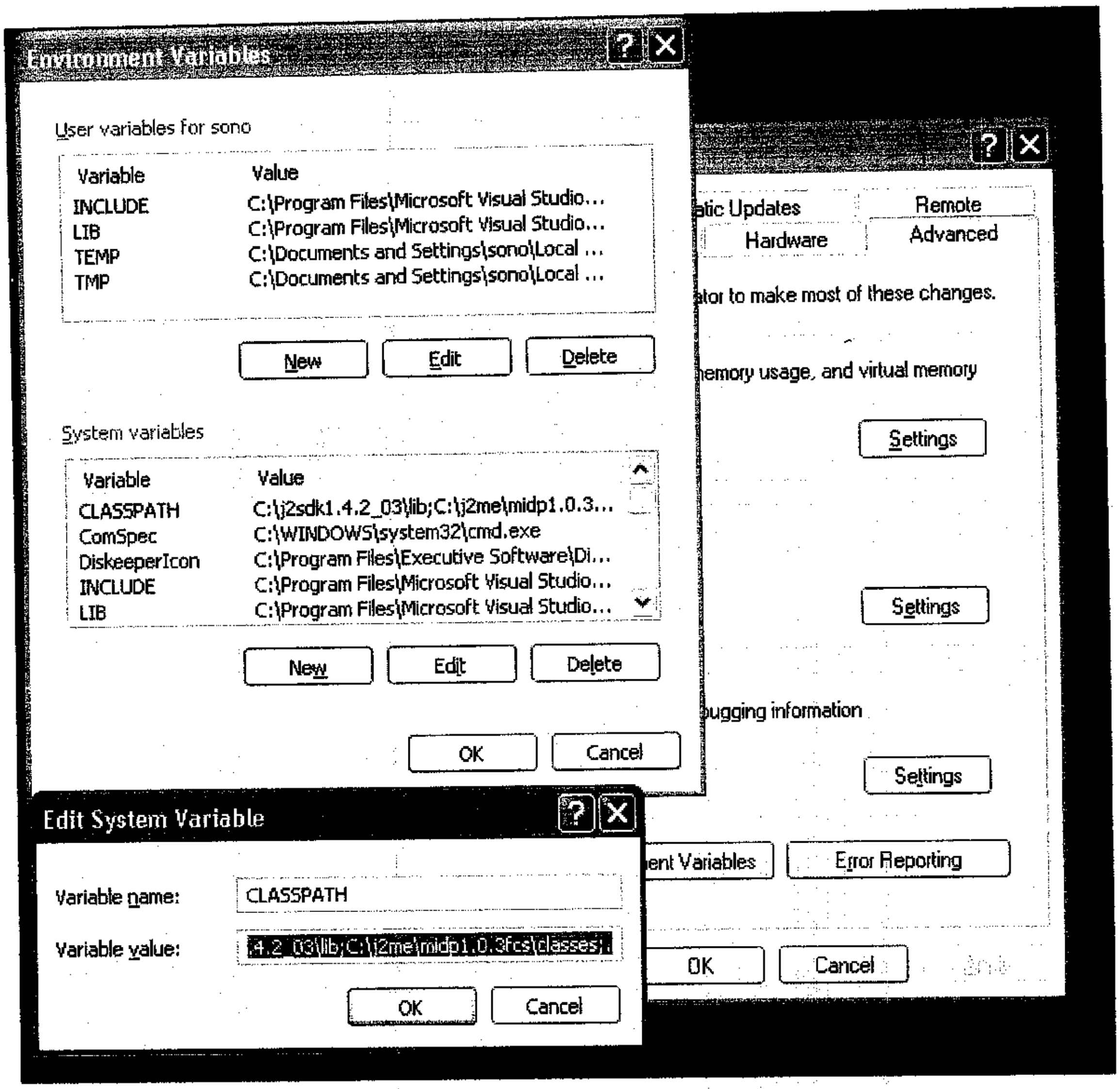
Like the operating system, java has a similar system for locating classes when compiling and executing programs in the virtual machine. The CLASSPATH environment variable is just a list of directories which contains your installed class files and directories. To compile and run your applications, you should need to adjust your Java CLASSPATH – the place where the Java compiler and run time (JVM) will look for classes.

At this stage, all you need to do is add the MIDP *classes* directory. You can do this by editing the CLASSPATH variable in the same way of editing PATH variable. If you don't have an existing CLASSPATH_ variable just create a new one. Update it by adding \classes path of your MIDP installation. For example:

C:\j2me\midp1.0.3fcs\classes;

Including the JDK library path, a complete CLASSPATH environment variable generally will look like the following line. (Don't miss that "." At the end!).

CLASSPATH = c:\j2sdk1.4.1_02\lib;c:\j2me\midp10.0.3fcs\classes;.



User variables for sono

Variable	Value
INCLUDE	C:\Program Files\Microsoft Visual Studio...
LIB	C:\Program Files\Microsoft Visual Studio...
TEMP	C:\Documents and Settings\sono\Local ...
TMP	C:\Documents and Settings\sono\Local ...

System variables

Variable	Value
CLASSPATH	C:\j2sdk1.4.2_03\lib;C:\j2me\midp1.0.3...
ComSpec	C:\WINDOWS\system32\cmd.exe
DiskeeperIcon	C:\Program Files\Executive Software\Di...
INCLUDE	C:\Program Files\Microsoft Visual Studio...
LIB	C:\Program Files\Microsoft Visual Studio...

Variable name: CLASSPATH

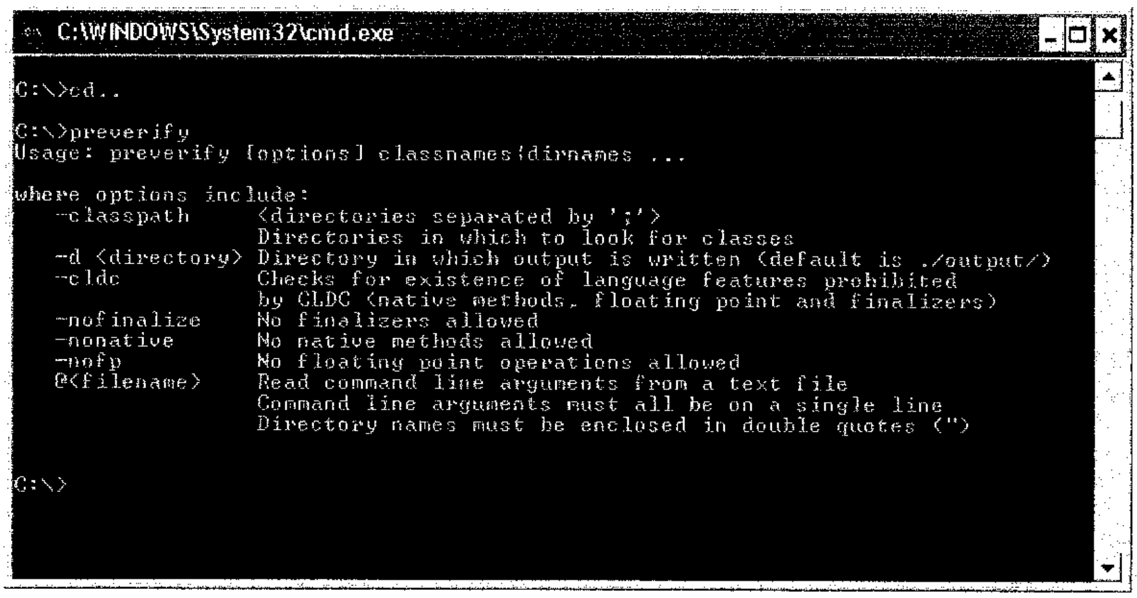
Variable value: .4.2_03\lib;C:\j2me\midp1.0.3fcs\classes;.

Figure 7.4

Testing the setup.

Now you've installed the JDK and MIDP, you should test things to make sure you're really ready to go.

Grab a shell and type *preverify*. The result should match those in figure 6

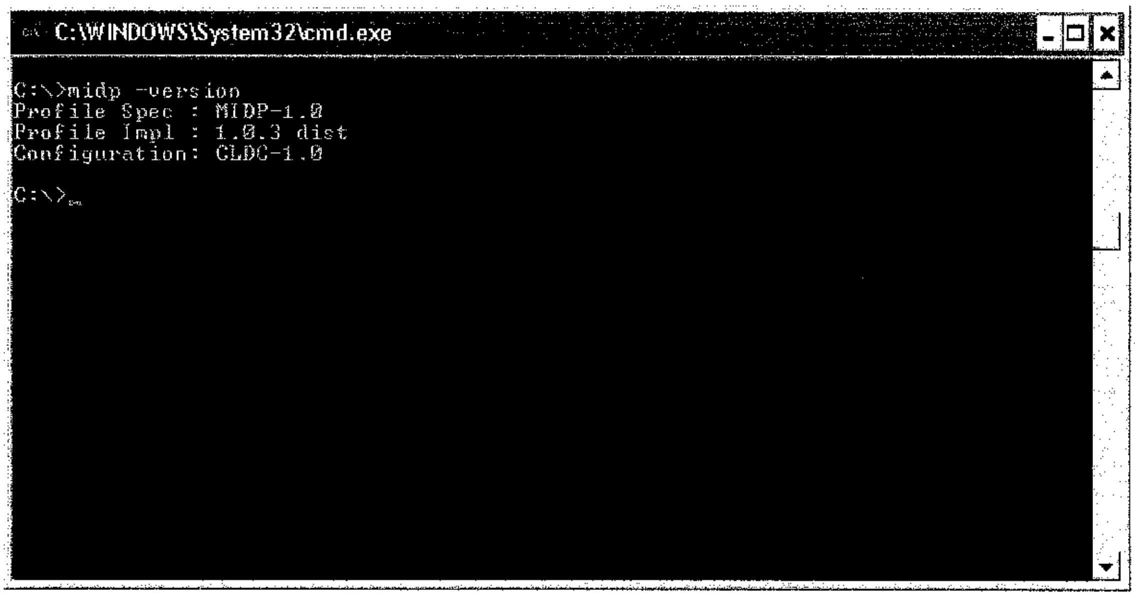


```
C:\>cd..

C:\>preverify
Usage: preverify [options] classnames|dirnames ...

where options include:
   -classpath     <directories separated by ';'>
                  Directories in which to look for classes
   -d <directory> Directory in which output is written (default is ./output/)
   -cldc          Checks for existence of language features prohibited
                  by CLDC (native methods, floating point and finalizers)
   -nofinalize    No finalizers allowed
   -nonative      No native methods allowed
   -nofp          No floating point operations allowed
   @<filename>    Read command line arguments from a text file
                  Command line arguments must all be on a single line
                  Directory names must be enclosed in double quotes (")


C:\>
```

Figure 7.5

Output from the *previrify* command.

Next you need to make sure the MIDP emulator is available. Enter *midp –version* in the shell. The results should looks like figure 7.



```
C:\>midp -version
Profile Spec : MIDP-1.0
Profile Impl : 1.0.3 dist
Configuration: CLDC-1.0

C:\>_
```

Figure 7.6

The result from entering the *midp –version* command in the command prompt.

Write codes

Compiling the Application

Next, is to compile the application. To prepare for this, open a shell and change the path to c:\j2me\project\RoadRun. To start compiling, enter the command line as

Javac –target 1.1 –bootclasspath %MIDP_HOME%\classes <filename>.java

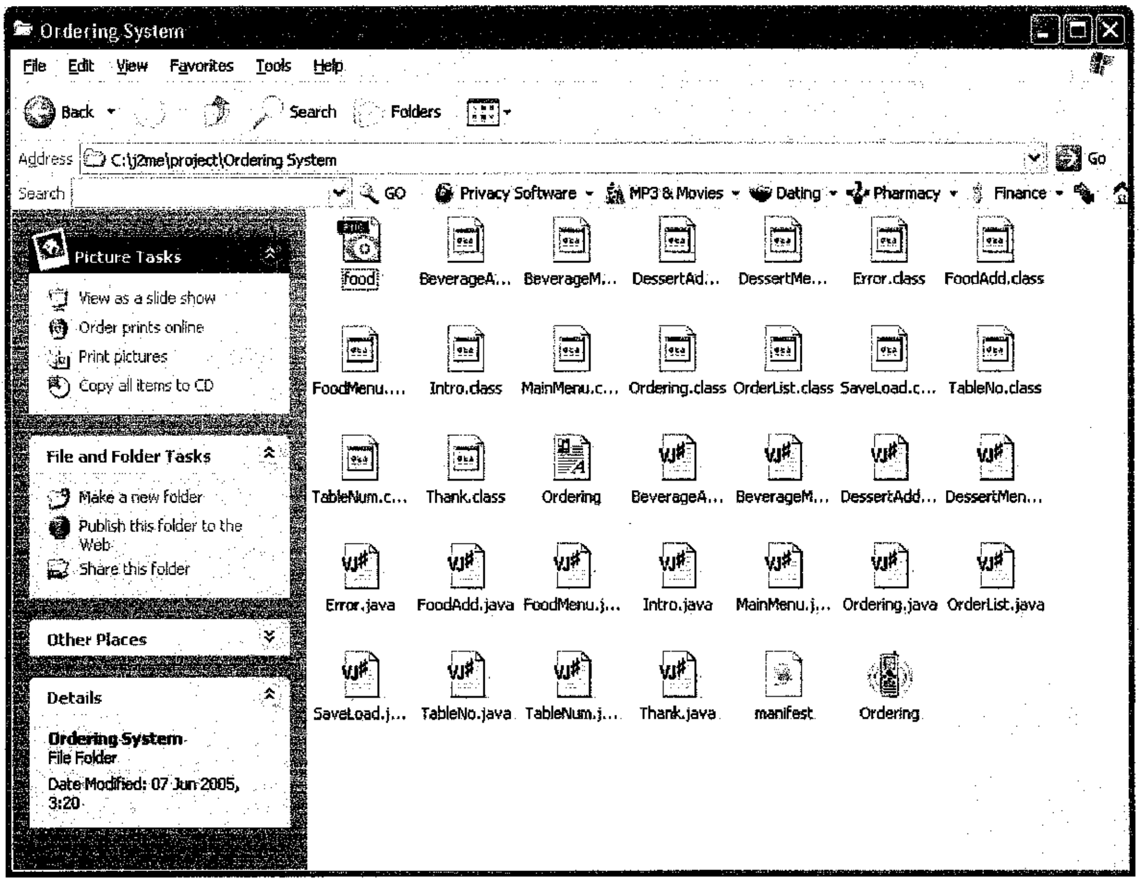
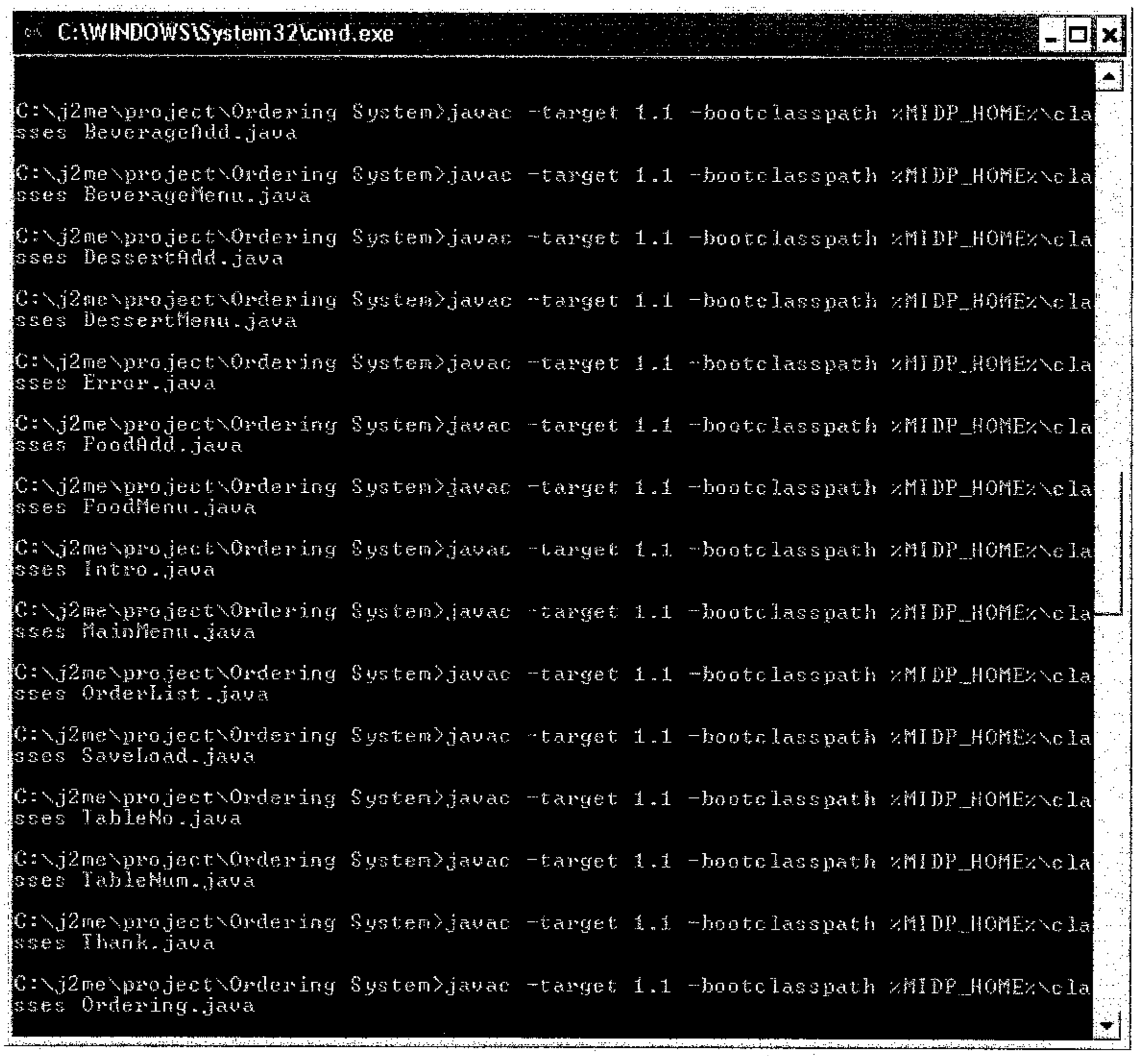

Ordering System
File Edit View Favorites Tools Help
Back Search Folders
Address C:\j2me\project\Ordering System
Go
Search GO Privacy Software MP3 & Movies Dating Pharmacy Finance
Picture Tasks
View as a slide show
Order prints online
Print pictures
Copy all items to CD
File and Folder Tasks
Make a new folder
Publish this folder to the Web
Share this folder
Other Places
Details
Ordering System
File Folder
Date Modified: 07 Jun 2005, 3:20
food
BeverageA...
BeverageM...
DessertAd...
DessertMe...
Error.class
FoodAdd.class
FoodMenu....
Intro.class
MainMenu.c...
Ordering.class
OrderList.class
SaveLoad.c...
TableNo.class
TableNum.c...
Thank.class
Ordering
BeverageA...
BeverageM...
DessertAdd...
DessertMen...
Error.java
FoodAdd.java
FoodMenu.j...
Intro.java
MainMenu.j...
Ordering.java
OrderList.java
SaveLoad.j...
TableNo.java
TableNum.j...
Thank.java
manifest
Ordering

Figure 7.7



C:\WINDOWS\System32\cmd.exe

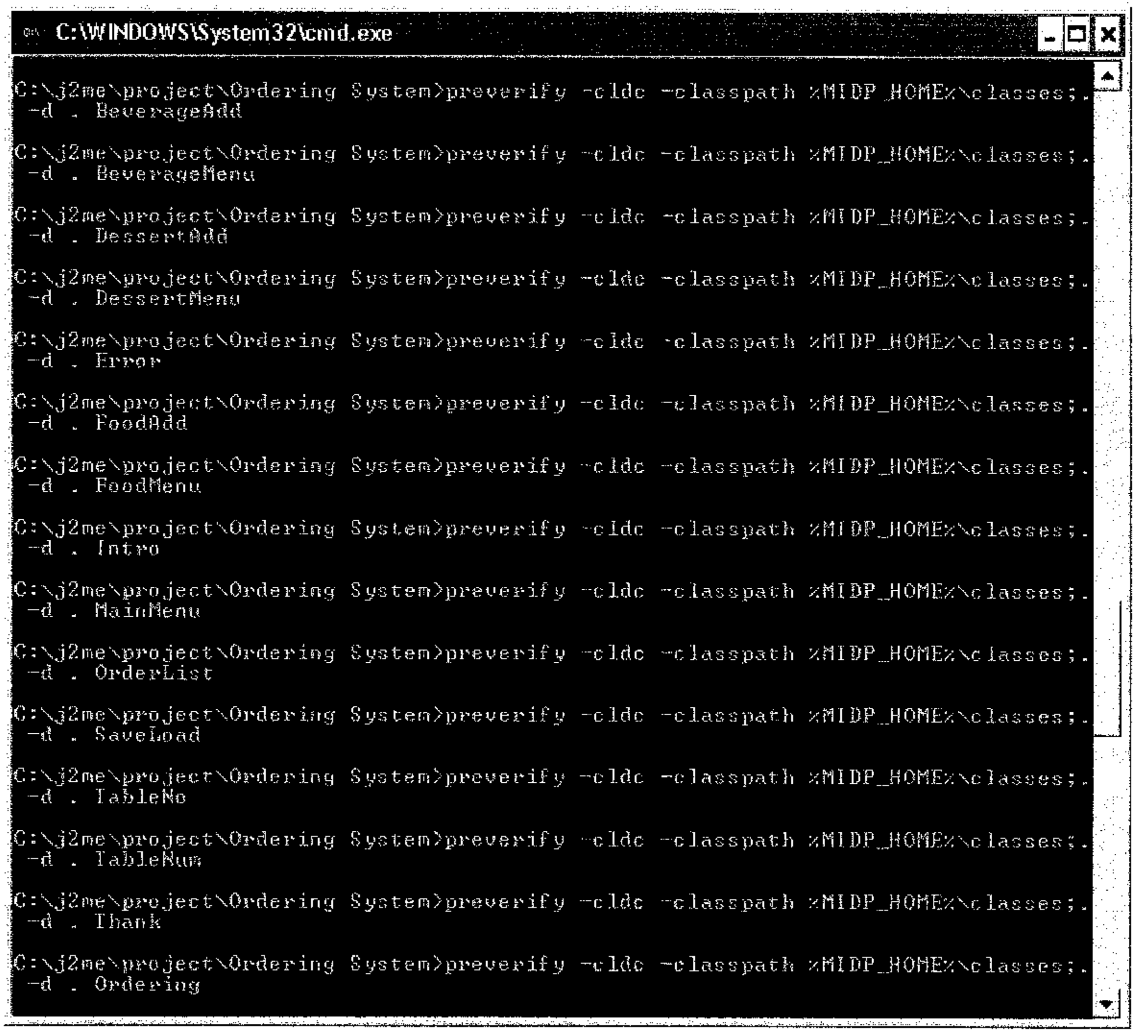
```
C:\j2me\project\Ordering System>javac -target 1.1 -bootclasspath %MIDP_HOME%\cla
sses BeverageAdd.java

C:\j2me\project\Ordering System>javac -target 1.1 -bootclasspath %MIDP_HOME%\cla
sses BeverageMenu.java

C:\j2me\project\Ordering System>javac -target 1.1 -bootclasspath %MIDP_HOME%\cla
sses DessertAdd.java

C:\j2me\project\Ordering System>javac -target 1.1 -bootclasspath %MIDP_HOME%\cla
sses DessertMenu.java

C:\j2me\project\Ordering System>javac -target 1.1 -bootclasspath %MIDP_HOME%\cla
sses Error.java

C:\j2me\project\Ordering System>javac -target 1.1 -bootclasspath %MIDP_HOME%\cla
sses FoodAdd.java

C:\j2me\project\Ordering System>javac -target 1.1 -bootclasspath %MIDP_HOME%\cla
sses FoodMenu.java

C:\j2me\project\Ordering System>javac -target 1.1 -bootclasspath %MIDP_HOME%\cla
sses Intro.java

C:\j2me\project\Ordering System>javac -target 1.1 -bootclasspath %MIDP_HOME%\cla
sses MainMenu.java

C:\j2me\project\Ordering System>javac -target 1.1 -bootclasspath %MIDP_HOME%\cla
sses OrderList.java

C:\j2me\project\Ordering System>javac -target 1.1 -bootclasspath %MIDP_HOME%\cla
sses SaveLoad.java

C:\j2me\project\Ordering System>javac -target 1.1 -bootclasspath %MIDP_HOME%\cla
sses TableNo.java

C:\j2me\project\Ordering System>javac -target 1.1 -bootclasspath %MIDP_HOME%\cla
sses TableNum.java

C:\j2me\project\Ordering System>javac -target 1.1 -bootclasspath %MIDP_HOME%\cla
sses Thank.java

C:\j2me\project\Ordering System>javac -target 1.1 -bootclasspath %MIDP_HOME%\cla
sses Ordering.java
```

Figure 7.8

Preverify your Class File.

Next step is to preverify the class file. Enter the command

Preverify –cldc –classpath %MIDP_HOME%\classes;. –d . <filename>



```
C:\WINDOWS\System32\cmd.exe

C:\j2me\project\Ordering System>preverify -cldc -classpath %MIDP_HOME%\classes;.
 -d . BeverageAdd

C:\j2me\project\Ordering System>preverify -cldc -classpath %MIDP_HOME%\classes;.
 -d . BeverageMenu

C:\j2me\project\Ordering System>preverify -cldc -classpath %MIDP_HOME%\classes;.
 -d . DessertAdd

C:\j2me\project\Ordering System>preverify -cldc -classpath %MIDP_HOME%\classes;.
 -d . DessertMenu

C:\j2me\project\Ordering System>preverify -cldc -classpath %MIDP_HOME%\classes;.
 -d . Error

C:\j2me\project\Ordering System>preverify -cldc -classpath %MIDP_HOME%\classes;.
 -d . FoodAdd

C:\j2me\project\Ordering System>preverify -cldc -classpath %MIDP_HOME%\classes;.
 -d . FoodMenu

C:\j2me\project\Ordering System>preverify -cldc -classpath %MIDP_HOME%\classes;.
 -d . Intro

C:\j2me\project\Ordering System>preverify -cldc -classpath %MIDP_HOME%\classes;.
 -d . MainMenu

C:\j2me\project\Ordering System>preverify -cldc -classpath %MIDP_HOME%\classes;.
 -d . OrderList

C:\j2me\project\Ordering System>preverify -cldc -classpath %MIDP_HOME%\classes;.
 -d . SaveLoad

C:\j2me\project\Ordering System>preverify -cldc -classpath %MIDP_HOME%\classes;.
 -d . TableNo

C:\j2me\project\Ordering System>preverify -cldc -classpath %MIDP_HOME%\classes;.
 -d . TableNum

C:\j2me\project\Ordering System>preverify -cldc -classpath %MIDP_HOME%\classes;.
 -d . Thank

C:\j2me\project\Ordering System>preverify -cldc -classpath %MIDP_HOME%\classes;.
 -d . Ordering
```

Figure 7.9

The output form building demo MIDlet.

Running the MIDlet

After compiled and verified the classes, now time to see the result in action. The simplest way to view a MIDlet is to use MIDP RI emulator. Command used

Midp –classpath . RoadRun

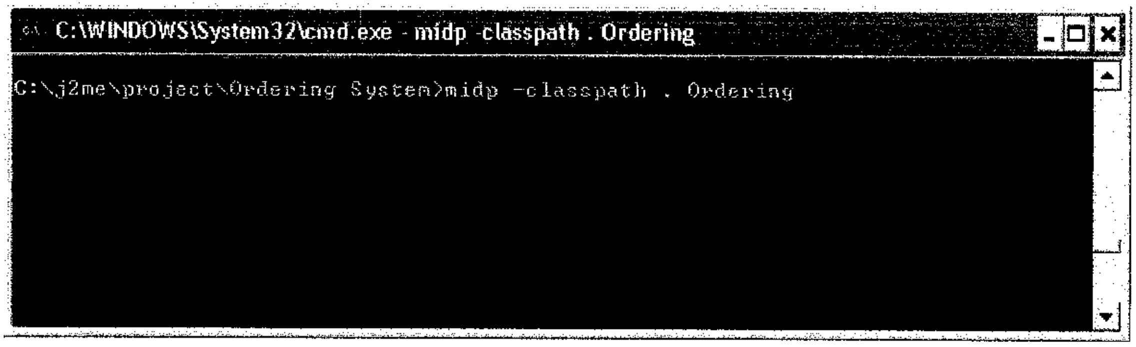

```
C:\WINDOWS\System32\cmd.exe - midp -classpath . Ordering
C:\j2me\project\Ordering System>midp -classpath . Ordering
```

Figure 7.10

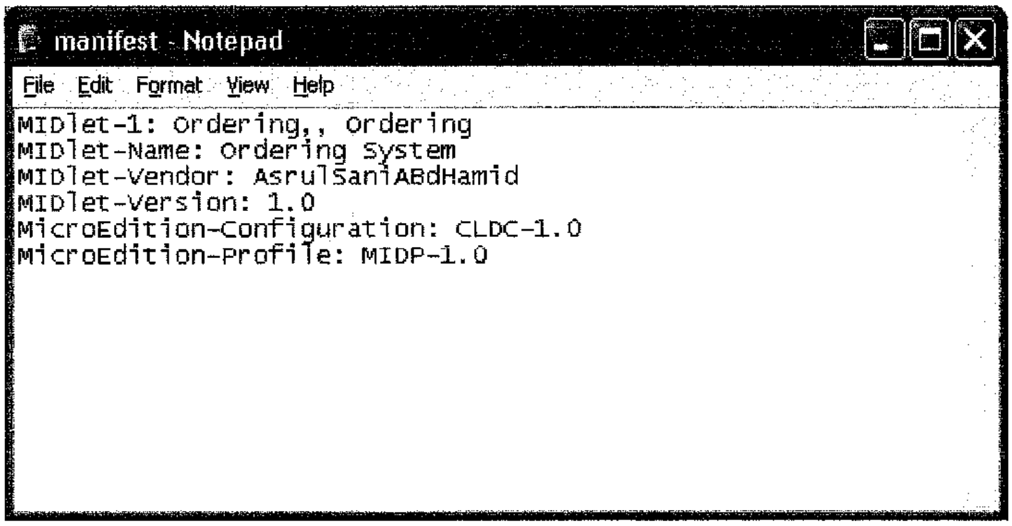
Our RoadRun MIDlet running on the sun emulator.

Figure 7.11

Creating Full Package.

Creating a JAR
Now when the classes ready, we create JAR (Java archive) file.

First using text editor, make a new file named *manifest.mf* in the project directory. This new file contain the as figure shows.

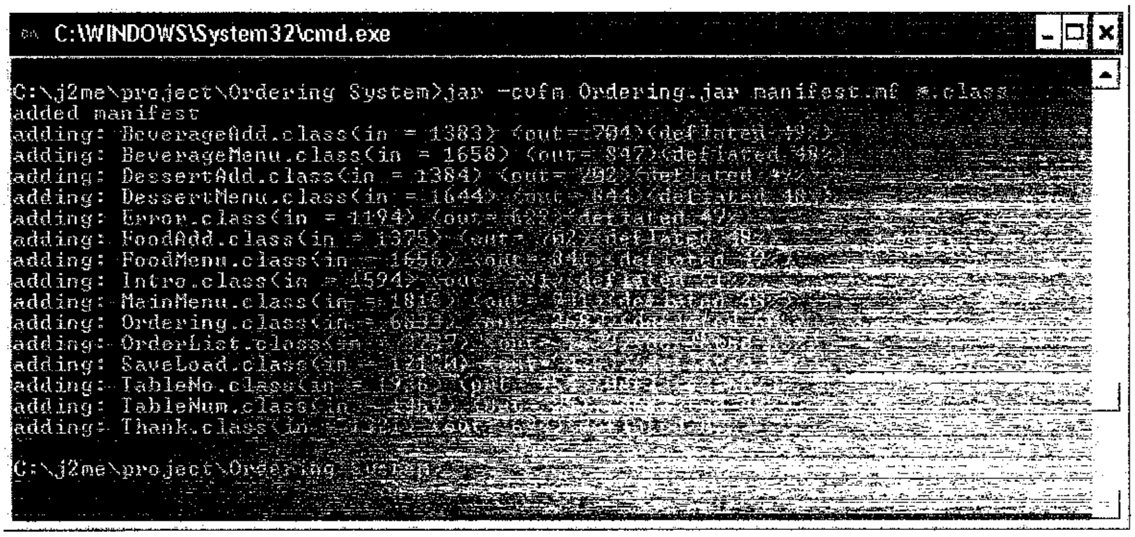


```
MIDlet-1: Ordering,, Ordering
MIDlet-Name: Ordering System
MIDlet-Vendor: AsrulSaniABdHamid
MIDlet-Version: 1.0
MicroEdition-Configuration: CLDC-1.0
MicroEdition-Profile: MIDP-1.0
```

Figure 7.12

Then create a JAR file using command :

*Jar –cvfm RoadRun.jar manifest.mf *.class*

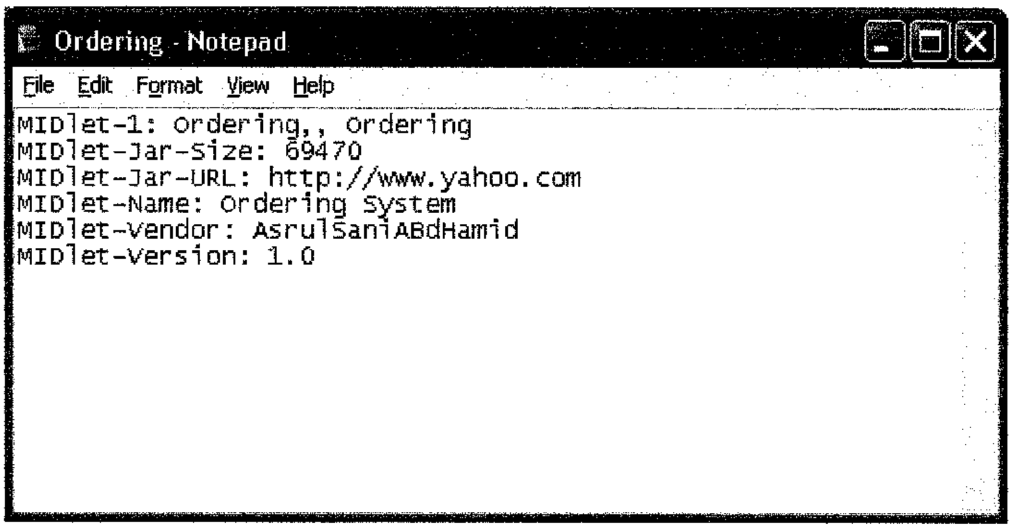


```
C:\j2me\project\Ordering System>jar -cvfm Ordering.jar manifest.mf *.class
added manifest
adding: BeverageAdd.class(in = 1383) (out= 704)(deflated 49%)
adding: BeverageMenu.class(in = 1658) (out= 847)(deflated 48%)
adding: DessertAdd.class(in = 1384) (out= 702)(deflated 49%)
adding: DessertMenu.class(in = 1644) (out= 844)(deflated 48%)
adding: Error.class(in = 1194) (out= 622)(deflated 47%)
adding: FoodAdd.class(in = 1375) (out= 702)(deflated 48%)
adding: FoodMenu.class(in = 1656) (out= 84[illegible]
adding: Intro.class(in = 1594) [illegible]
adding: MainMenu.class(in = 1816) [illegible]
adding: Ordering.class(in = 6833) [illegible]
adding: OrderList.class(in = [illegible]
adding: SaveLoad.class(in = [illegible]
adding: TableNo.class(in = [illegible]
adding: TableNum.class(in = [illegible]
adding: Thank.class(in [illegible]

C:\j2me\project\Ordering System>
```

Figure 7.13

Figure shows that JAR filed created.

Creating a JAD file.
Then create the corresponding JAD file to represent your suite. Using a text editor, make a new file name RoadRun.jad in the project directory. This file should contain the contents as shown in figure ?



```
MIDlet-1: Ordering,, Ordering
MIDlet-Jar-Size: 69470
MIDlet-Jar-URL: http://www.yahoo.com
MIDlet-Name: Ordering System
MIDlet-Vendor: AsrulSaniABdHamid
MIDlet-Version: 1.0
```

Figure 7.15

Running the package.
Now to run the package in the emulator using following command;

Midp –classpath . –Xdescriptor RoadRun.jad

Transfer into Device.

Use the Bluetooth Dongle to transfer only the RoadRun.Jar not all of the files.

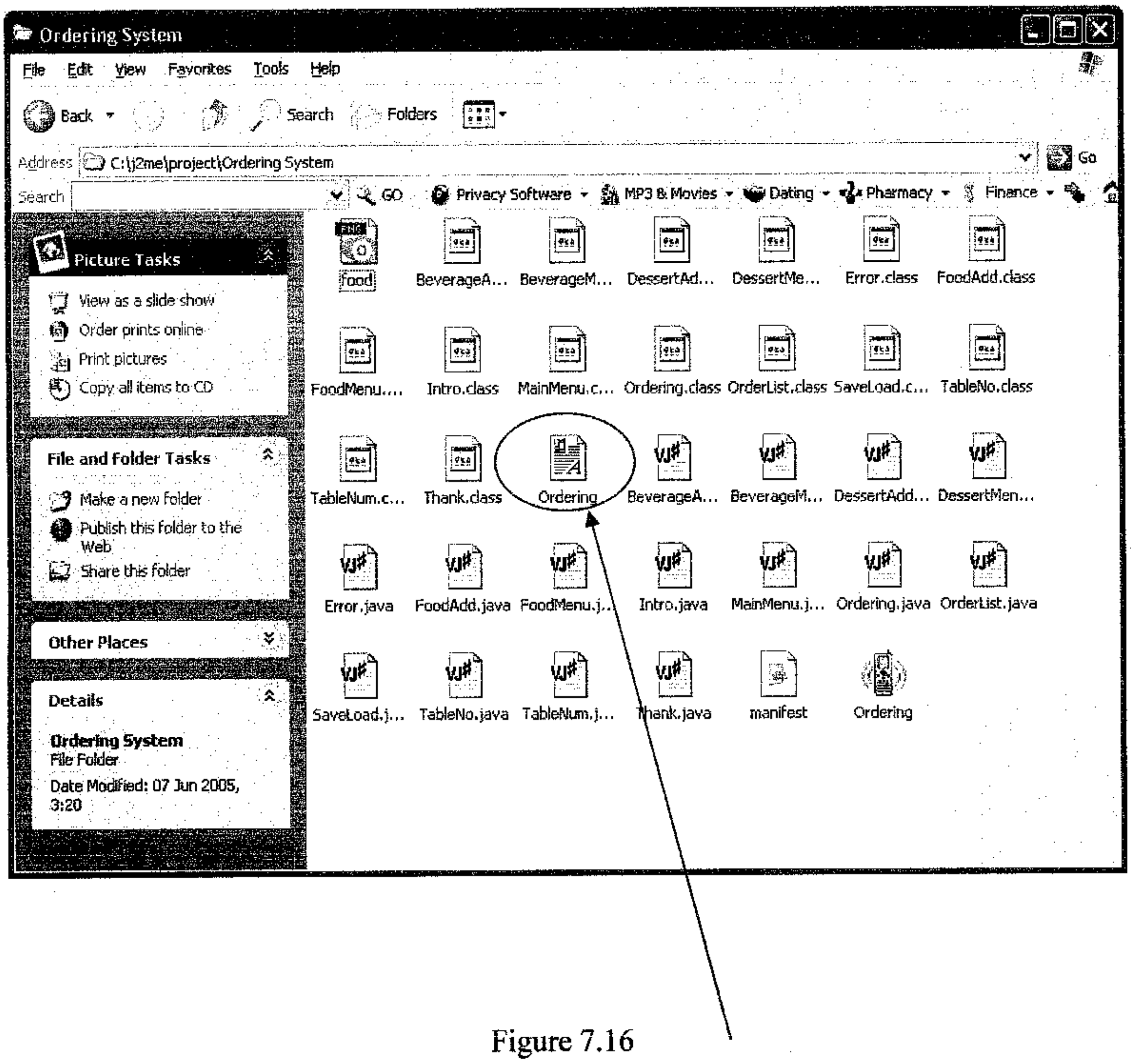


Figure 7.16

The Ordering.jar that will be transferred to the mobile device.