ANALYSIS, DESIGN, AND DEVELOPMENT OF MOBILE APPLICATION BY USING JAVA TECHNOLOGY

By:

Naema Yahya Ahmmed

Supervisor:

Dr. Mohammed A. Khlaif

This thesis is submitted to the Department of Computer Science in partial fulfillment of the requirements for the degree of Master of Science in Computer Science

Al_Tahadi University, Faculty of Science

Department of Computer Science

Sirt, G. S. P. L. A. J.

Academic year 2007-2008

Faculty of Science

Department of Computer Science

Title of Thesis

Analysis, Design, and Development of Mobile Application by Using Java A technology

By

Naema Yahya Ahmed

Approved by:

Dr. Mohammed A. Khlaif ....[illegible]............
(Supervisor)

Dr. Majdi Ali Muhammed Attoumi ....[illegible]............
(External examiner)

Dr. Omar M. Sallabi ....[illegible]............
(Internal examiner)

Countersigned by:

Dr. Ahmed Farag Migoub

(Dean of faculty of science)

[illegible]

بِسْمِ اللهِ الرَّحْمَنِ الرَّحِيمِ

﴿قَالُوا سُبْحَانَكَ لَا عِلْمَ لَنَا إِلَّا مَا عَلَّمْتَنَا إِنَّكَ أَنتَ الْعَلِيمُ الْحَكِيمُ﴾

صـــدق الله العظيم

سورة البقرة الآية (32)

Acknowledgements

I would like to thank my Supervisor Dr. Mohammed A. Khlaif for giving me an opportunity to complete this thesis. It is really my pleasure to work under his guidance. He taught me not only how to be a successful MSc student but also how to do a good academic work.

My special thanks are extended to Dr.Zakaria.S.Zubi whose information, comments, corrections and criticisms really enriched my work.

I would like to thank Dr.Jaras Raju for correcting the Language of this thesis.

Thanks for my family and friends for their support and understanding.

Above all I thank my God for his grace and mercy that gave me confidence and Knowledge.

Naema Yahya Ahmed

Abstract

In this research, we will discuss the main scope for building application on mobile phone. The application is to build a directory which will contain Personal *Mobile number, Name and E-mail* with some functions for which the possibility to browse all records stored in the directory. The possibility for querying a current e-mails and also some processes such as adding new record, delete an existing record. The Analyzing and Designing will be carried out using a common Unified Modeling Language (UML) because of it's clear definition of the system stages and it's object oriented properties with the use of different UML types models such as use case diagram, class diagram, sequence diagram and many others diagrams as well which will be used in case of no methodology used and also implemented by the use of J2ME Technology as a programming language. J2ME is the most suitable technology for mobile application today. It contains many ready components for mobile application which is going to be the programming environment for my research work. This environment requires Wireless tool Kit called (WKTool) and a Software Development Kit named as (SDK) environment which are readily available for the use with any operating system today. The developed application will be tested on different operating system such as Windows XP with a certain mobile devices that are very limited in memory , Small screen size and data entry capabilities.

The development of the system will be tested and carried out on a suitable mobile device

July 2007

Table of Contents

List Of Figures

Glossary:

Concept	Brief Explanation
- API	Application Programming Interface. A standard Program Library that comes with the Java Software.
- AWT	Abstract Windowing Toolkit, the Library graphics and GUI Programming from IDK1.0.
- CLDC	The Configuration that uses a limited JVM called KVM for suitable mobile phones.
- J2ME	Java 2MicroEdition, the version of Java used for limited power and storage devices such as mobile phone.
-JAD	Java Application Descriptor file, contains information used by the devise.
-JAR	Java archive file, which contains all class files and resource files used by the application.
- JVM	Java Virtual Machine. The theoretical model for a computer, which the Java interpreter realizes.
- MIDP	Mobile Information Device Profile, a set of J2ME APIs defines how Software application interface with mobile phones.
- MIDlet	A small Java application that is intended for mobile devices.
- OMG	Object Management Group, a standard set for the modeling all phases of software development system.
- PDA	Personal Digital Assistance, the simple type of PCs provides a way

	for interact with computers and mobile phone.
- RMS	Record Management System. Set of Java classes for storing and retrieving simple set of data.
- SDK	Software Development Kit. A standard development tool for Java.
- UML	Unified Modeling Language, an object-oriented modeling language for modeling software system.
- WTK	Wireless Tool Kit, application development kit to run MIDlets.

Chapter One

Introduction

The miniaturization of PCs in the form of Personal Digital Assistance (PDAs), Tablet PCs, and other mobile devices provided new ways for users to interact with computers with growing digital systems such as Cellular standards that are providing easier and quicker ways for use by a user. This is due to the incomplete presentation of the knowledge needed to develop an mobile applications. Much of the existing literature tends to explain how to develop mobile applications technically without describing other factors influence mobile application development and integration. In this thesis we introduce the means of the mobility and describe what mobile is. Our description includes some of the broad considerations must be taken into account during the mobile application design and development [15, 3]. Mobile phone will play the main role in the way we communicate but it will be in tandem with other devices such as laptop, palmtop and others. Convergence is the current communication industry buzz word - providing the solutions to meet people's needs as cost effectively as possible, by combine the best of fixed and the best of mobile.

1.1 Application Development

The application is to build a directory which will contain Personal Mobile number, Name and E-mail Address. The analysis and design of this directory will be carried out using Unified Modeling Language (UML) and the data stored as records in a Record Store as a particular database, while programming will be considered using Java2 Micro Edition (J2ME) Technology. In this work, we try to build an application on mobile system by using different UML models such as UseCase diagram, class diagram, sequence diagram and other diagrams.

Since J2ME is the most suitable technology for mobile application, contains many ready components for mobile application it is going to be the programming environment for this research, it also requires Wireless Kit (WKTool) and Software Development Kit (SDK) environment which are available under any operating system platform.

The programming language we used is J2ME, in addition to this work; we provide some features and services or functionalities to be used with this application which include the following:

- The possibility to display all records stored in RecordStore.
- Query for certain record.
- Adding a new record to the RecordStore.
- Delete Existing record.
- Option Order to Browse records.
- Dialing phone numbers.

1.2 Thesis Goals

The goal of application of this thesis will provide information on Records stored in a mobile directory named Personal Email directory. The application should be designed to enable users to access this information quickly and efficiently, without having to navigate through multiple screens. All records stored in the Record Store named as E_mail Addr.

1.3 Related work

- **Rolf Henniker[23]**, discussed in mobile application, some components which plays an essential role to develop mobile application such as display text (Hello World) as a simple application programmed using J2ME, this study based on design of web applications, which is built with classes window, client page and server page which specify contents displayed in mobile screen. From this study, it can be concluded that the use of J2ME to build any application of web application to show how can develop it [23].

- **Wisdom Assen BV**, studied mobile application for use multi_tier technologies where in the last years wireless devices such as cell phones, PDA's have grown enormously in popularity[36].

  Wireless devices has kept their owners connected to the out side world at any time from anywhere and are thus much more flexible than the wire connection on PCs [3].

- **Valentino Lee Heather Schneider[35]**, explained how to create mobile applications, and identify the main technologies used in the mobile application development. He attempts to address the advanced principles of mobile application and also illustrating the specific details of mobile technologies with several examples with some case studies, such as (mobile zoo) this application allows zoo visitors to use Pocket PCs to view web pages that contain animal and zoo information while they are touring the zoo [35].

1.4 Mobility Definition

Mobility can be defined as the capability of being able to move or to be moved easily.

Mobility pertains to people's use of portable and functionally powerful mobile devices that offer the ability to perform a set of application provided to the user [31].

Mobile can be defined as "any service or facility that supplies a user with general information of knowledge regardless of location and time" [15]. There are some of discussions concerned with this part, Vavoula and sharples (2002) they suggested three ways which mobile can be considered mobile as "mobile in term of space; it is mobile in different areas of life; it is mobile with respect to time ", These definitions suggest that mobile systems should be capable of delivering educational content any time any where the users needs [15].

1.4.1 Mobile Devices

The advancements being made in mobile devices are truly incredible, and the selection is astounding. A few years ago the choice was between a mobile phone and a simple PDA. Now there is a long list of options ranging from high PDAs to small phones with mobiles capabilities [15].Even the simplest devices provide enough computing power to run small application such as play games. No matter which type of mobile application you are looking to deploy application, a device will be available to meet your needs [35].

1.4.2 A Brief History of Mobile phone

The basic concept of mobile phones began in 1947 when researchers looked at crude mobile (car) phones and realized that by using small cells (range of service area) with frequency reuse could increase the traffic capacity of mobile phones substantially, however, the technology to do it was nonexistent[16].

1947 - A Public mobile system using frequencies in the (35 to 44 MHz) band began operations along the highway between New York and Boston. These frequencies were thought to carry greater distances however a problem with skip-distance propagation carried interfering conversations for long distances. These early mobile telephone systems used push-to-talk operation.

1955 - Number of wireless line channels available at (150 MHz) was expanded from (5 to 11) by the creation of new channels between the old ones (channel spacing of 30 kHz).

1956 - 12 wireless line channels were added near (450 MHz). All systems operated in a manual mode, with each call to or from a mobile unit being handled by a special mobile telephone operator.

1964 - A new system (150 MHz) was developed providing automatic channel selection for each call, eliminated the need to push-to-talk operation, and allowed customers to do their own dialing.

1.4.3 Mobile phone features

Mobile phones often have features beyond sending text messages called(SMS) and making voice calls- including Internet browsing, music (MP3) playback, personal organizers, e-mail, built-in cameras and camcorders, games, infrared and Bluetooth connectivity, call registers, ability to watch streaming video or download video for later viewing, and serving as a Wireless Modem for a PC[35].

In most countries, the person receiving a cellular phone call pays nothing. However, in China (including Hong Kong), and Canada, one can be charged per minute [35].

1.5 Mobile Application Programs

Application Programs that uses the Java Micro Edition (Java ME) platform is a set of technologies and specifications developed for small devices such as mobile phones. Java2ME uses smaller subsets of Java SE components, such as smaller virtual machines and APIs that are specifically targeted at consumer [35].

Creating successful mobile applications requires a profound knowledge of various technologies such as application design and development of mobile applications using mobile software infrastructure [35].

1.5.1 Mobile Application infrastructure

In this section, we consider the mobile infrastructure from a mobile application developer's perspective.

We start by describing the characteristics of several mobile device types, including cellular telephones, pagers, and PDAs which our application must run on it ,this characteristics, the capabilities, portability, and cost considerations of each of these mobile devices that is an important to determine what the primary function of the user's mobile device will be used [35]. As we have seen, not every mobile device does the same thing, nor is it necessarily capable of the same functionality.

In the other word, the capabilities of mobile device are constrained both by its size and cost [31, 35].

1.5.2 Mobile Application architectures

Application architectures are often modeled to highlight or illustrate the overall layout of the software (e.g. application code) and hardware (e.g. mobile device). While there are many possible combinations of software and hardware, application architectures often fall into a series of recognizable patterns.

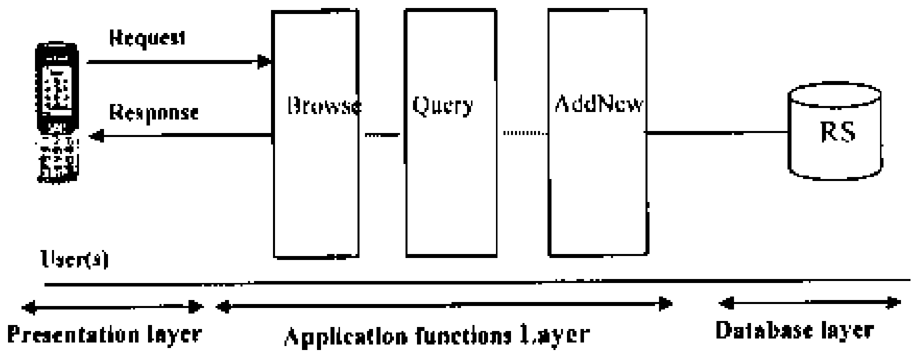


Figure 1.1: Mobile application Architecture

By breaking up application functionality into sub layers to make the architecture scalable. The code that interacts with the user to run a mobile application is often placed in the presentation tier. A second tier, which holds the application function (AddNew-delete- Browse Record) is often referred to as the application tier. The third tier often referred to as the Data Base RecordStore (RS) or data source tier which contains all data required for an application as shown in figure 1.1.

In addition, many factors contribute to the success (or failure) of mobile application, these include the mobile device and wireless connectivity. Many people not realize several application models are available for mobile development, a different set of characteristics that make it appropriate for some applications and inappropriate for other.

1.6 Architecture Requirements

Architectural design must address system functionality, and user requirements [35].

1.6.1 User System Requirements

The architecture should typically handle the widest range and number of user different requirements[31]. In our application, personal E-mail Directory contains several elements that the user need for displaying e-mails on the screen , create new e-mail, delete an exist record in the directory.

1.6.2 System Requirements

For all mobile applications we use J2ME technology to download all needs of environment for J2ME. The application applied on real devices to transfer applications to other devices via using Bluetooth technology for instance.

1.7 Mobile Technology

The exponential growth of mobile technology in recent years, increasing availability of high –bandwidth in performance and popularity of handheld devices for any type of mobile device [16]. From a technology perspective, handheld devices such as handheld computers and digital Assistance (PDAs) are more affordable today than before [23].

"The term 'mobile technology' is commonly applied to a cluster of different techniques whereby a device can conduct communications without a physical cable to connect it for other devices or networks. During the act of communication, the device may need to be relatively still, or confined to a small area. Alternatively, it may be able to move, and even to move at considerable speed, during the act of communication. In some cases, it is imperative that the device moves at speed, because its operation depends on current induced by the transit of a coil that it contains through a magnetic field created by some nearby static device"[23].

The technologies are extremely interesting for educated people due to their low cost relative to desktop computers and the spontaneous and highly personalised access they give to the vast educational resources of the Internet [23].

The impacts of the new mobile technologies need to be appraised, evaluated and made more widely known.

1.7.1 Technology Independence

In an ideal world, you should develop mobile applications that are used with hardware as device and platform independent as possible [37]. This is not easy but good applications tend to be written so that they can run on many devices and platforms (any time any where).

1.7.2 High performance and availability

The architecture of mobile application must typically have excellent performance during analysis and design resource. "If people are expected to use the application for any device at any given time, the architecture must also have high technology available" [35].

1.8 Thesis organization

The thesis ordered as follows:

Chapter one including the introduction about the research with all requirements that we needed to complete it.

Chapter two presents the Life Cycle of the application by using different models of UML diagrams.

Chapter three describes the analysis of our application that specialized in another diagrams.

Chapter four describes the design as a complete phase of analysis with using another UML diagrams.

Chapter five describes the implementation phase as a result of analysis and design phases to programming the application.

Finally chapter six summarizes the conclusion and recommendations in the same scope of another mobile application.

Chapter Two

System Development Life Cycle (SDLC)

Any Information system, projects move through four phases, planning, analysis, design and implementation; all systems require analysts to gather information required to create a new system. Today most of the existing systems analysis and design moving to object- oriented techniques, which view a system as a collection of classes or objects that have both data and processes [1].

These phases refined and illustrated as follows:

- **Planning**:-Planning phase is a fundamental process to perform required operations (AddNew, Delete, Browse and Query) by listing the tasks needed to complete the user requirements to meet the system's objectives and information about those tasks, this phase includes all of the system management deliverables [33].

- **Analysis**:-The analysis phase includes how a user uses the application for each operation with all tasks that explained in the planning phase.

- **Design**:-The design phase decides how the system will operate, in terms of the software, the user interface, forms, the specific programs, Record store as database, and classes that will be needed. In most cases, the system will design the interface specifies how the user will move through the system (e.g; navigation methods such as menus and on-screen buttons and forms that the system will use it).Next the RecordStore as database that define exactly what data will be stored any where are developed [1].

Finally, the application will be divided into classes each class will reflect the needs and the functionality of using each class of the program [34].

- **Implementation**:-The final phase in SDLC which is the implementation phase, during which the system is actually built (in the case of a Packaged software design) [7].

2.1 Unified Modeling Language (UML)

- UML is an object-oriented modeling language for specifying, constructing and documenting the software systems, as well as for business modeling and other non software systems [34].
- UML is a standard set by the Object Management Group (OMG) for the modeling of all phases of software development from requirements, analysis, and design.
- UML is one of the most welcome events in recent years in the field of Object-oriented analysis and design. It was implemented by three gentlemen, Booch, Jacobson, and Rumbaugh; they joined forces and created a single common modeling language called Unified Modeling Language (UML).

The word 'unified' in UML's in the title refers to the fact that when the UML was introduced in the late 1990 it brought together the notations of analysis and design methods [34].

This language is a recognized standard developed and maintained by the Object Management Group (OMG) that is based on object oriented methodology [33], the OMG publishes guide lines and specifications that provide a common frame work for application development will gradually introduce the elements of the language as we proceed. We will start with class diagram that illustrates a single class and gradually add more classes using associations and generalizations. We will also use an activity diagram to illustrate control structures, threads, and other parallel processes. Another element called Sequence diagrams can be used as well for showing how objects send messages to each other.

The focus on the system analysis and design of an applications resulted from the need for comprehensive approach to develop the system with object-oriented technology.

Current object-oriented development methods use the UML to define object-oriented constructs and models.

The object-oriented approach used potentially to reduce errors, costs, and increase flexibility based on its inherent features [4]

2.2 The Importance of UML

The UML is applicable to object-oriented problem solving. It gives anyone from business analyst the ability to analyze, design and then program any common applications in terms of Software design [34]. This ability demands the ability to address the model. Where this model is an abstraction of the underlying problem domain that is in the actual world from which the problem comes.

Any models consist of objects (instance of class) that interact by sending messages to other hosts. All objects consists of elements so called (attributes), these elements had some events or actions called as Behavior or operation in our abstraction model [10].

2.3 Software Modeling

The most common types of software models are analysis, design, implementation, and deployment models. These models represent the system at different levels of abstraction and from different perspectives during the software life cycle.

- **Analysis models:** It captures the abstractions in the problem domain. Software engineers must understand the problem before they can solve it. The most common types of analysis models are use case and class diagram models.

- **Design models:** These models capture the abstractions in the solution domain. They allow engineers to experiment with potential solutions and judge their quality without actually building the system. The most common types of design models are data flow diagrams, class diagrams and interaction diagram [1].

- **Implementation models** capture the logical design of an implementation that would otherwise be lost in the detail of the implementation. The most common types of implementation models are class diagrams and interaction diagram .Some modeling tools may even create implementation models directly from source code.

-**Deployment models** are used to plan and document the physical deployment of software and hardware components [1]. A deployment diagram shows hardware devices and software components that run on them.

2.3.1 The Use of Models

Models are abstractions used to understand complex problems and their potential solution better.

Architectures, for example could not build even simple dwellings if the maps have not been prepared prior to construction in order to achieve success and quality.

Models are a simplification of the real world and help us to understand what we are building [6].

- The purposes of models are to describe requirements during analysis and design phases.
- In the analysis phase, the purpose of the model is to capture or use of all requirements of the system and to model the basic classes and collaboration between them [34].

• In the design phase, the purpose of the model is to expand the analysis model into working technical solution with consideration for the implementation environment [34].

2.3.2 Static and Dynamic Software

Software has both static and dynamic aspects. The static aspects are those aspects that don't change over time. This includes classes, interfaces, attributes and relations between classes in the used system. The dynamic aspects of software are those aspects that can change as the software executes [33].

Interaction diagrams complement class diagrams. Class diagrams model the static structure of software, and interaction diagrams model its dynamic behavior. Interaction diagram show how objects interact at runtime to complete a task at different level of the system. Interaction diagrams are used to model the behavior of:

1. **The system as whole:** - use cases describe the high-level behavior of the system as whole. An interaction diagram at this level would model the interactions between high-level analysis classes required to realize a use case or scenario from a use case [34].
2. **An operation:** - At the implementation level, an interaction diagram shows the sequence of methods calls that take place over a given period of time. Most common are interaction diagrams that model a single operation on a class [32].

2.3.2.1 Structural diagrams

UML defines a set of structural diagrams that are used to show relationships between classes in a system as the following:-

• **Class Diagram**:-the class diagram shows the relationships between a set of classes. This is the most common diagram in UML and is a structural view of how classes are related [15].

• **Sequence Diagram**:-the sequence diagram shows the internal interaction between classes in a specific time of cycle line [34].

• **Collaboration diagram**:-the collaboration diagram shows the structural organization of objects that communicate with each other [34, 14].

• **Component diagram**:-the component diagram shows the relationship between components. Components are built and combined to form applications.

2.3.2.2 Behavioral (dynamic) diagrams

• **Use Case diagram**: -the use case diagram shows the relationships between actors and use case where the actors are external entities to the system such as users and other systems [14].

• **State diagram**: - the state diagram contains a finite state machine that describes the behavior of a class. State machines are an excellent way to describe the event of a system.

• **Activity diagram**: - shows the flow of control through activities in a system.

2.4 Analysis phase

- **Class diagram**:-In this section we will take an application named (Personal E-Mail Directory) to illustrate all the use of UML tools as a description of diagrams that are used in our system including the classes such as: PersonalEmailDirectory, SimpleRecord, SimpleComparator, SimpleFilter and NetworkQuery.

- **State diagram:** - A state diagram shows the sequences of states when an object passes through it during its life cycle with responses and actions.

2.5 Design phase

- **Sequence diagram (one type of interaction diagram):** Sequence diagrams are used to display the interaction between objects and users within the system. They provide a sequential map of message passing between objects over a certain time [10].
- **Collaboration diagram (two type of interaction diagram):** Collaboration Diagrams describe the set of interactions and links between classes and show the relationships among objects [5]. The sequence and collaboration diagrams both show interaction, but the first diagram focus on the use of time for sending and receive messages; the second diagram focuses on the execution of an operation.

2.6 Development tools and Emulators

Choosing a development tool for mobile applications is not an easy task [9]. There are many development tools available, all with various strengths and weaknesses. Most development tools for mobile Application are Software Development Kit (SDK). This software is used to add support for emulation environments to their offerings. In addition to the development environments, the tools provide emulators to simulate how the application will look and feel without having to deploy it to a physical device [11]. Simulation is often necessary, because we will not have every device combination available for our testing [11]. This development tools are geared to develop applications for multiple devices.

2.6.1 J2ME Technology

J2ME stands for Java 2, Micro Edition, it is a version of Java at devices which have limited processing power and storage capabilities; these includes mobile phones, wireless devices and others [20].The J2MEtool can be described as the following:

- J2ME can run on devices with as little as 16 KB.
- J2ME is divided into configuration, profiles and optional APIs (Application Programming Interface) demonstrated as following:-
    - **Configuration:-**

      A configuration defines the minimum Java technology that an application developer can expect on a broad range of implementing devices and also it designed for a specific type of device based on memory capacity and processor power.

      It specifies a Java Virtual Machine (JVM) which can be used on these devices [27].

      Most of configuration which is suitable for mobile phones is Connected Limited Devices Configuration (CLDC), it uses a limited JVM called Kilo Virtual Machine (KVM) is provided as part of J2ME and running on devices with limited memory size and Network connected devices [27].
    - **Profile:-**

      This is the application environment for wireless devices based on the CLDC. A profile is layered on top of a configuration and adds the APIs required developing applications for a family of devices [11].

      Mobile Information Device Profile (MIDP) is the profile suitable for mobile phone; MIDP includes the idea of a MIDlet, a small Java application similar to an applet but one that is intended for mobile devices [11].

MIDP is a set of J2ME APIs that define how software applications interface with mobile phones; it is a part of Java 2 platform Micro Edition (J2ME) provides a persistent storage mechanism called the Record Management System (RMS). The RMS is a set of Java classes for storing and retrieving simple sets of data [9, 11].

- **Optional APIs:-**

  This define specific additional functionality which may be included in a particular configuration, as an Example including the Java API for Bluetooth as emulation technologies which will be discussed later in chapter five, section (5.10)[11].

  Another J2ME configuration is the Connected Devices Configuration (CDC) targets the advanced consumer electronic and embedded devices such as smart communicators and smart personal digital assistants (PDAs), these devices runs on 32-bit microprocessor /controller and has also 2MB of total memory for the storage of Virtual Machine (VM) [11]

2.6.2 J2ME tools

With the use of Java 2ME, we need to install Software Development Kits SDK. This is a basic tool for installing the Application development kit by Java wireless toolkit (WKTool) to run MIDlets [37, 10], these tools will be described in chapter Five section (5.1).

Chapter Three

Analysis phase

The goal of analysis phase is to understand the system requirements to decide whether the new system is or not needed [4]. In UML, the common notation will be used with every Object-Oriented Analysis (OOA) technique to represent both analysis and design phases together where object-oriented analysis consists of three steps to analyze OOA requirements these steps are shown as follows:

1-**UseCase modeling**:-Determines how various results are computed by the mobile application system [10].

2-**Class modeling**:-Determine the classes, their attributes, and inter-relationships between the classes [34].

3-**Dynamic modeling**:-Determine the actions performed by or to each class or subclass. The aim of dynamic modeling is to produce a state diagram [10].

A detail description of these diagrams will be given in this phase later.

3.1 Analysis Requirements

In this section, we describe requirement models through diagrams that include UseCase, Class and activity diagrams. The process that will be followed to create the requirement model is the object-oriented analysis. The object-oriented analysis process of the system involves four main activities. These activities are: create mobile system definition, decide the system's main functionality through UseCases, build the class diagram, and develop scenarios and corresponding sequence diagrams [32].

The system definition is created first, and UseCase model are identified. It provides a natural way of dividing the system into manageable units. The objective is to identify what the system must do for the user to complete the required work tasks.

3.2 Definition of Personal E_Mail Directory

Our application is based on the use of Object-oriented approach to use classes to define elements of the system, such as Forms, Commands, and Alerts also there might be many more events that are important to be identified in the mobile system. The events translate into a list of UseCases as following:

1. Open Directory.
2. Query for certain record.
3. Add New Record.
4. Delete Record.
5. Brows Directory.

The approach involves creating models that define the requirements, the design, and then the implementation of E-Mail Directory application. This approach is determining how to interpret the behavior of objects in an application, through written scenarios with the use of UseCases, Class, and Sequence Diagram. In our application it requires Directory Items that will store information about a Person such as (personal Name, E_Mail, and phone number). The requirements for this item include a class named Personal E-Mail Directory. The class allows us to store records about any number of people based on the list stored in database (Record Store) where we can add new record, browse information about certain record by using query function. We can also delete any record. These are potential uses for requirements definitions.

Requirements are needed in order to help us formulate and crystallize what the application must be able to do in certain sense [14].

3.3 Conceptual Design

The objective of the conceptual design is to build a conceptual model of the application domain taking into account the functional requirements captured with Use Case. Object- oriented techniques are used to construct the conceptual model, such as finding classes and association and defining inheritance structures. The conceptual model is represented by an ordinary UML class diagram [1].

3.4 Structural Modeling

A structural or conceptual model describes the data in a logical perspective without indicating how the data are stored [1]. During Design phase, the structural model reflects exactly how the data will be stored in Record Store. We will use the UseCase, Class diagram, and object diagram to create the structural model [1].

3.4.1 Use Case Description Level (Abstraction)

- **Essential**: It describes the general description of the system process without including technical information [34].
- **Real** : a real use cases are undesirable during analysis and should only be used during analysis for specific reasons. Real use cases are handy for requirements gathering.

Normally high level essential UseCases are done during the analysis phase of the system. A high level real UseCase is done during the design phase only if necessary [18].

UseCase is behaviorally related sequence of steps (a scenario) with both automated and manual, for the purpose of completing a single business task described from the perspective of external users that needs to interact with the system, and results of decomposing the scope of system functionality into many smaller statements.

3.4.1.1 Scenarios, Use cases, Relationships and the use case Diagram

When we analysis any system, we try to identify the main functionality that the system will have and the main ways it will be used. In this application of E-Mail directory there are some ways used to indicate processes such as: add new record, Browse all records, delete exist record, we call each of these ways in the application as a UseCase. Each use case must have scenario from another technique to meet the requirements analysis to be identified and described to accomplish some objective by describes what happens when the user select a certain function. The product then display out sample menus for that selection [33]. One approach for doing scenarios is to write them out like scripts that highlight how the objects behave when the system operates. Each scenario documents the way the objects or a small set of the objects behave in a certain situation. This helps us to modulate and analyst the system [1].

A use case is a sequence of actions or processes that an application will perform to represent how a system interacts with its environment. An actor is a person (User) who interacts with the application where each UseCase involves one or more actors that uses the system [1].

Use Cases can be described using the Unified Modeling Language (UML) and illustrated in UseCase diagram [21].

A UseCase captures the main functionality (one and only one function) of the system from a user or an actor for dividing the system into smaller parts which can be implemented separately. As in the event "open directory" is UseCase of UseCase diagram, each main event usually corresponds to the UseCase [34].

UseCase relationships explain how the use case is related to other use case and users. The four basic types of relationships are association, extend, include, and generalization, these basic types are considered as follow:

- An association relationship is the communication that takes place between the UseCase and the actors that use the UseCase [33, 34].
- An extend relationship represents the extension of the functionality of the UseCase to incorporate optional behavior.
- An include relationship represents the mandatory inclusion of another UseCase and the breaking up of a complex UseCase into several simpler ones.
- The generalization relationship allows UseCases to support inheritance. Inheritance represents a specialized UseCase to a more generalized one (from the specialized UseCase to the base UseCase) [34].

3.4.1.2 System Boundary in UseCase

The UseCases are enclosed within a system boundary, which is a box that represents the system and clearly delineates what parts of the diagram are external or internal to it [1]. It represents all actions in the system.

3.4.2 Main Use Case Diagram

A UseCase Diagram (UCD) describes what a system does rather than how? The UseCase diagrams are connected to scenarios that happen when someone interacts with the system. Each UseCase are represented as ellipses, and actor is depicted as icon connected with solid line to the UseCase that is connected and interacts with him [26]. UML UseCase Diagram can be used to describe the functionality of a system in a horizontal way. That is, rather than merely representing the details of individual features of the system. UCDs can be used to show all of its available functionality .UCDs has only 4 major elements such as: The actors that interact with a system, the use cases or the services, that the system knows how to perform, and the lines that represent relationships between these elements [34].

Use Case using relation is called a Uses relation. It is represented by directed line between classes (user and directory) as shown in the figure 3.1.

The Use Case Model serves to defining what functions will or will not be performed by the system [1]. The necessary good UseCase conditions are stated as the following: Easy to understand and that means the following:

- The user enters a number of items into the system.
- The system displays the item details to the user.
- The user enters the quantity required into the system.

- Mapped to prototype screen or system interface prototype.
- Complete including all alternate flows.
- Greed with the users of the system.

3.4.2.1 Creation of Main Use Case Diagram

First, use case diagram starts with the system boundary.This forms the border of the system (e.g; the system's functionality) from a user. Second, use cases are drawn on the diagram taken directly from use case descriptions. Third, the actors are placed taken from who will use a system. The forth step is to draw lines connecting actors to the UseCase interacts with it.

The system was developed with main Use Case diagram shown in the figure 3.1.

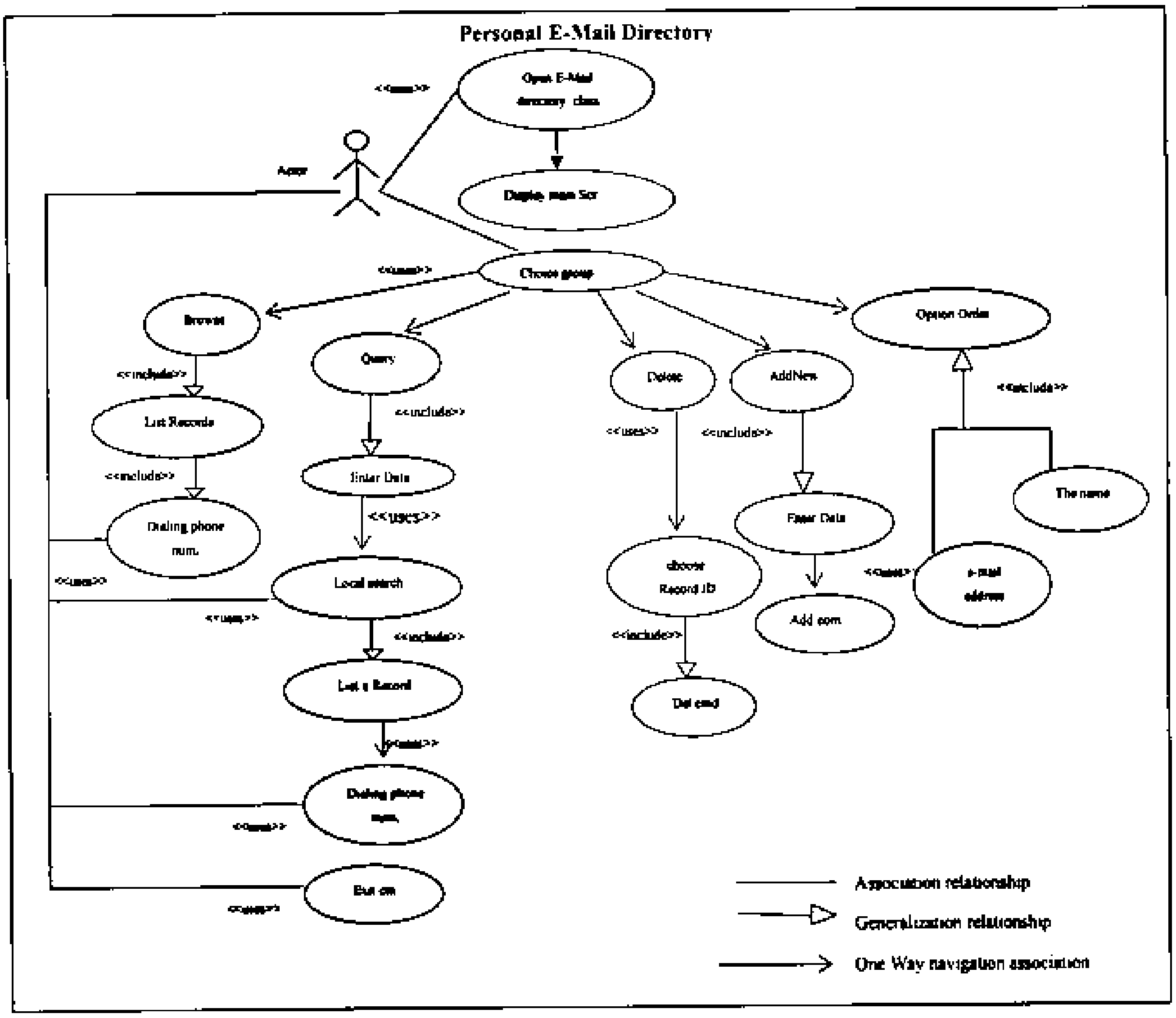


Figure 3.1: Main Use Case Diagram (UsecaseD)

This Figure shows the user of the E_Mail directory system and their use cases. Actor uses the system to perform specific selection of a grouping of choices. Thus, the UseCases Browse records associated with add new record process that is created to list all recorded stored by user and search for a certain record by the name or E_Mail address. All functions can be selected while user's select the main class to show the main screen with one of these operations before the user can check its status [32].

Some UseCases have similar features in their behavior. For instance, Browse all record and Query for a certain record are both UseCases where a record can be displayed.

Scenarios can be used as a textual describtion of the UseCase goals.UseCases that are sub-goals of another UseCase can be related using the <<uses>> dependency. Thuse, the use of <<uses>>dependencies create a hierarchy of UseCase [32].

Recall that each UseCase might have more than one scenario .Each scenario might be documented as follows:

UseCase: Open Personal E_Mail Directory (PED), are represented in the main Scenario.

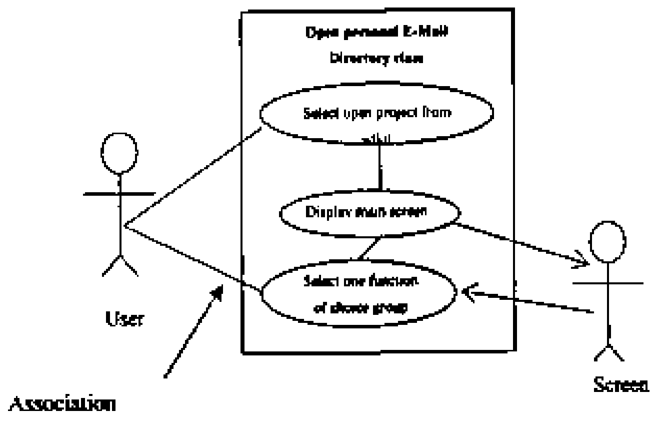


Figure 3.2: use case for use an application

This use case illustrated in figure 3.2: used when we want to use an application, the user selects a certain function to be performed. All use case can identify an interaction activity since these actions may be classified as inputting, Displaying, Deleting, changing or searching. Each scenario of use case extended from the scenario previously to complete this function.

- UseCase: Add new Record, are represented in the main Scenario.

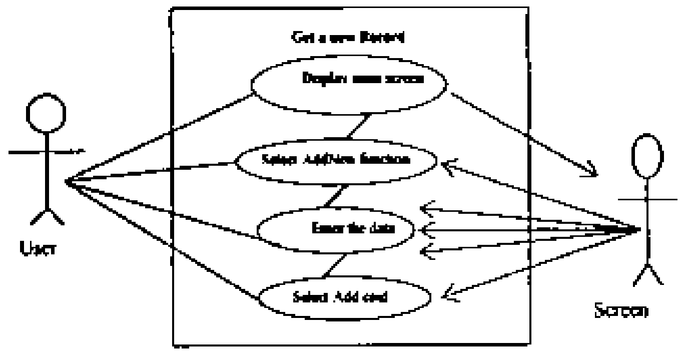


Figure 3.3: use case for Adding new Record

This UseCase used when we want to add new Record with the other information to record store, the user enter the user name, phone number, and E_Mail address and then select Add command to add this record to the other records stored in the directory. The system response with this selection by shown message illustrates this record has been added safely. All these adding of records any time can be request for any function selected to be applied. In figure 3.3:the class name screen represent the interfaces between all the other screens used in this UseCase such as the screen for entering the data or the screen for displaying alert information to complete any tasks.

- UseCase: Indicates information about a certain Record where placed in the main scenario.

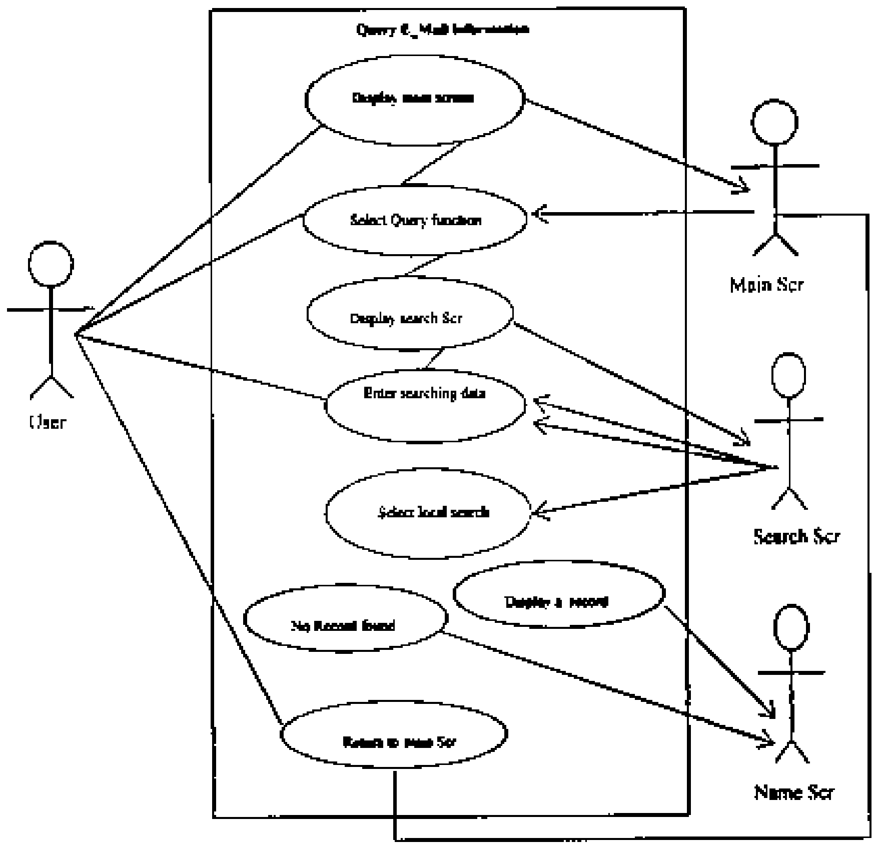


Figure 3.4: use case for Query Screen

When user selects this function, the directory response to this action that a user need to display information about record exist in the directory and determined it to a user, or show information this record is not found. This function can be divided into two interactions which can be identified in the scenario; the receiving personal information; and displays that presents in the results of the query.

- Use Case: Representing a select option order shown also from the main scenario.

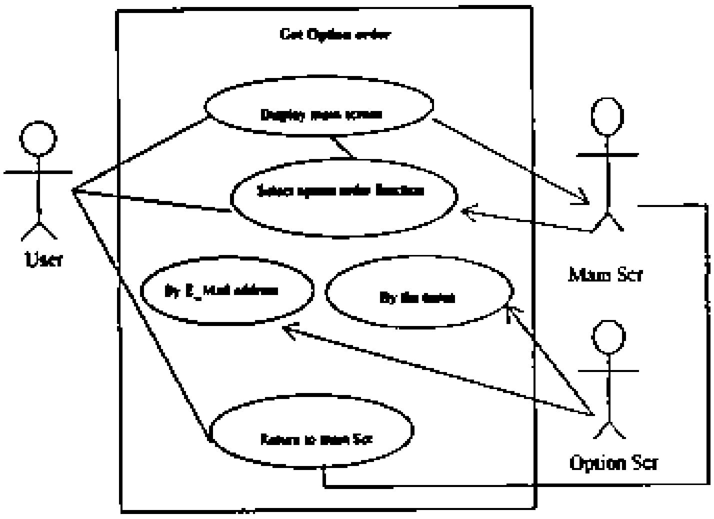


Figure 3.5: use case for option order Screen

This function used when a user Browse all records in certain order either by the name or by E_Mail address and then Browse all Records as a result to run this function. This order is also used with query function, as shown in the figure 3.4. This function performed with another classes with a simple Comparator that contains the order data of the name, E_Mail and a simple filter that contains the method which allow us to access to the data stored in the class simple Comparator. As a result we will show this order by displaying all the records in the order that it has been selected.

- UseCase: Display directory. Shown also from main scenario.

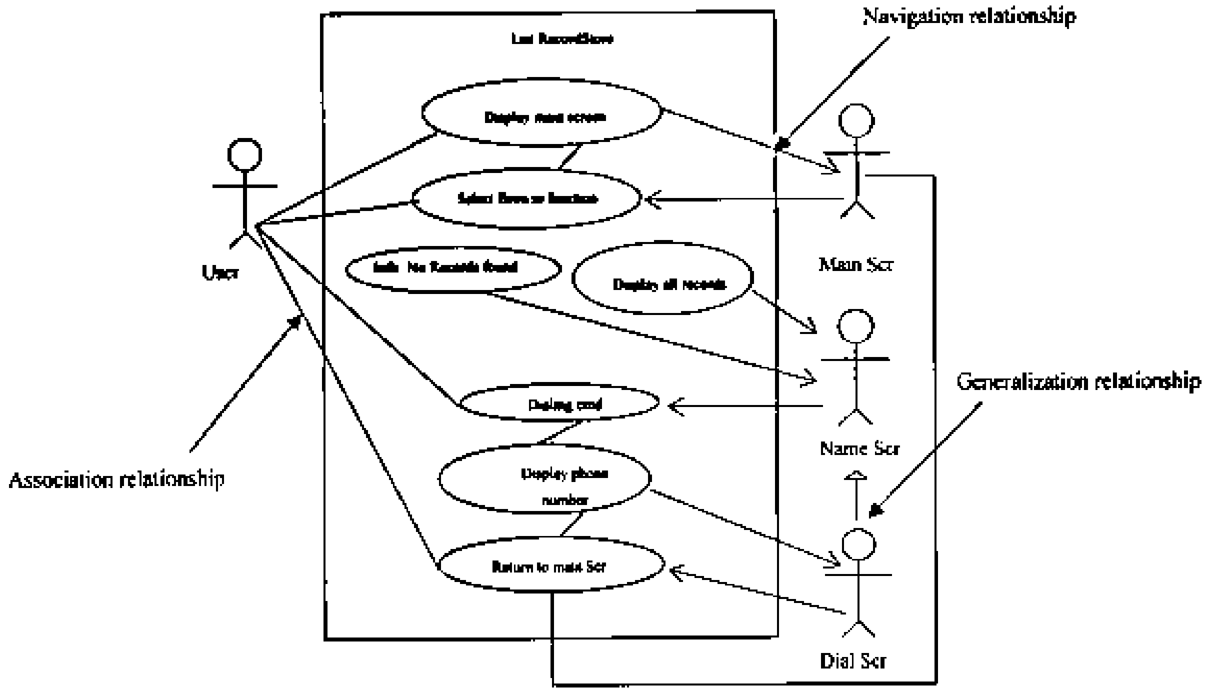


Figure 3.6: use case for Browse all Records

This function used to see a list of all records stored in the directory such as record store. Directory class list that on the screen all the stored records or shown the information alert (**records not found**) when the records are not stored. When we use such a functions a class SimpleComparator, SimpleFilter, and RecordEnumeration are used to retrieve all record with the option order to display record, as shown in the figure3.6 we can also show the phone number related with any record selected from a user by using Dial command in the Name Screen.

- UseCase: Delete a certain record. Were shown in the main scenario.

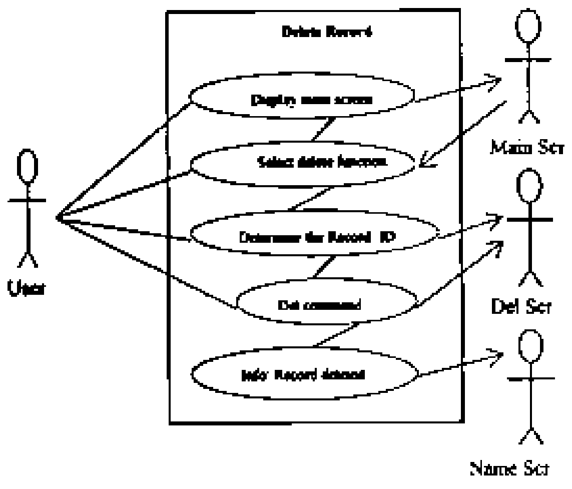


Figure 3.7: use case for Delete a Record

This function is used when a user delete an exist record from a directory based on the record ID, where the system request for this task with sure deleting if record found, or show alert message (**record not found**)if a record not found as shown in figure 3.7 .

3.4.2.2 Use Case Relationships

There are several standard relationships among UseCases or between actors and UseCase; we list these relationships as: Association, Generalization, and Extend relationship. These relationships described follows:

- **Association**:-The participation of the use case and actor communicate with each other are an association relationship, this is only relationship between actors and UseCases.
- **Generalization**:- is a generalization from use case A (one use case) to use case B (another use case) which indicates that A is a specialization of B, this communication is between UseCases or actors from the same kind which one is instance of another[34].

- Extend:-An extend relationship from use case A to use case B indicates that UseCase B may be extended by the use case A[12].

3.4.2.2.1 Notation of Use Case Relationships

- An association between an actor and a use case is shown as a solid line between the actor and the use case [10].
- An extended relationship between use cases is shown by a dashed arrow with an open arrow-head from the use case providing the extension to the base use case. The arrow is labeled with the key word <<extend>>.
- A generalization between use cases is shown by a generalization arrow, represented by solid line with a closed arrow -head.

3.4.2.2.2 Example of Use Case Relationships

To create a record we will use a create instruction from the record store in UseCase which can be shown as the following

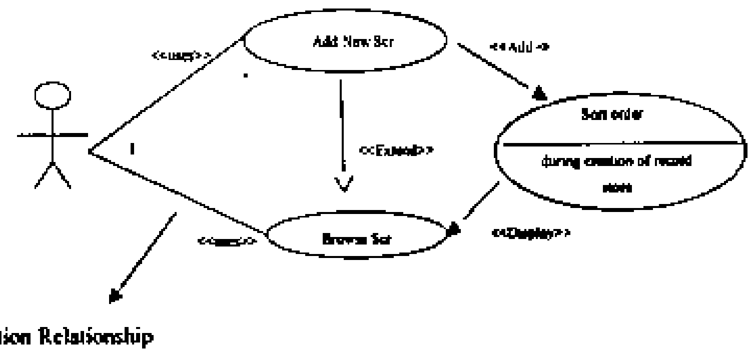


Figure 3.8: use case Relationships

In this figure, use case of Browse Scr is extended from a use case add new Scr and use case sort order based on display all records in a certain order by the name or by E_Mail address during creation of Record Store so that the relationship is Display between them as extend type relationship.

As described in section 3.4.2.2.1, and Represented in figure 3.8. Generalization relationship between two use cases can be represented in the figure 3.9:

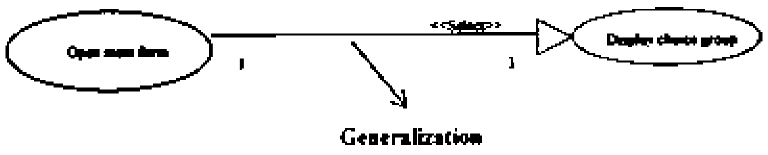


Figure 3.9: Generalization Relationships

3.5 Class Diagram

3.5.1 Analysis of UML Relationships between classes

An object might also be naturally related to other objects or classes [26]. Object relationships are much like relationships in a data model based on the message in which we view the objects. For example, in our application there is a button that is attached to a window and each window might have many buttons attached to it. A menu might contain many menu items and menu item might be contained on one menu [33].

The other aspect of an object relationship that is the same as in data modeling is the cardinality of a relationship. Cardinality refers to the number of associations that occur between objects [2,29]. The Unified Modeling Language (UML) uses the term multiplicity in place of cardinality. In our application e-mail directory is found for many mobile phones this is called one-to-many relationship .On the other hand many mobile phones are owned by one E_Mail directory. This is called one-to-one relationship [34].

There are two types of object relationships in our application: an association (or connection) relationship this describes Whole-part relationships. An association relationship means that one object is associated with other objects in some way. A whole-part relationship means that the relationship between objects is stronger. For this relationship there are two types: aggregation relationships and composition relationships[29].

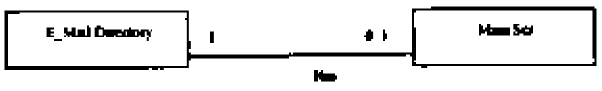


Figure 3.10: Relationships between objects

3.5.2 Developing initial scenarios and initial classes

The system is ready to start developing scenarios and building the class diagram. They will choose one or two use cases that are considered to be the most important and develop those in parallel with the class diagram [22].

3.5.3 Procedure for building the class diagram

There are several steps for developing the class diagram followed in the system as:

1. Identify main classes.
2. Identify relationships between classes.
3. Identify the important attributes.
4. Identify additional association relationships.
5. Identify methods of the classes.

3.5.4 UML Design Class Diagram (DCD)

UML Design Class Diagram (DCD) shows the classes and the relationships among classes that remain constant in the system over time. Classes are shown with their simple attributes, methods and relationship to other classes, these relationships include association, inheritance, and aggregation for a proposed system [21].

The class Diagrams identifies the static aspects of a system. It identifies all the classes for a system that are used where class diagram is the central diagram of a UML model for any system [33].

The static aspects mean that it does not have a time component. For example, the name of a class is static that does not change over time. The interactions between classes are dynamic aspect of the class which interactions describe the behavior or actions of class over time. In UML a class is the main building block in class diagram which represents information in the system. During analysis, class refers to actor, events, and things about which the system will capture information to create an application. During design and implementation, classes can refer to implementation specific artifacts like windows, forms, and other objects used to build the application [1].

Each class has Attributes and operation. Attributes are properties of the class and operations are actions or functions that a class can perform. In our application, the class name is PersonalE_MailDirectory, and the attributes of the class are:

- The name_Mail.
- E_Mail address.
- Phone number.
- Alert message.

And the methods of this class are:

- Start App ().
- #Pause App ().
- - Display alert ().
- -midletExit ().
- +command Action(c, d), as shown in figure3.11.

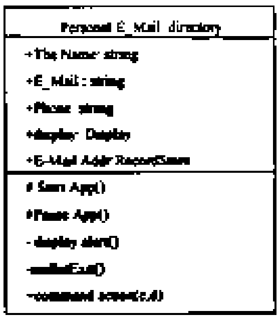


Figure 3.11: Personal E-Mail Directory class

The class Diagram produced during object-oriented analysis is a logical model. In object-oriented approach, each object represents an entity in a real-world application scenario, such as a user. Each object may have a set of properties and set of methods. Properties represent the characteristic of an object, methods are the functions and procedures that operate on an object for example, Query operation makes information about the state of object available to other object (Browse), If Query function asks for information from attributes in the class (e.g; name, E_Mail Address, or phone) then it will show that on the new class (screen) as a result of an active operation [33].

The main class diagram shown in the figure 3.12 of OOA phase depicts the classes and their attributes but not their actions (methods). Where the methods are inserted into the detailed class diagram of the design phase in chapter five. Determination of all the actions of the product is performed by examining the interaction diagrams of every scenario. The attributes of a class should be declared private (+) that means (accessible only within an object of that class), or a protected (#) which is (accessible only within an object of that class or subclass of that class, or public (-) which is (accessible for all an object of that class and other classes).

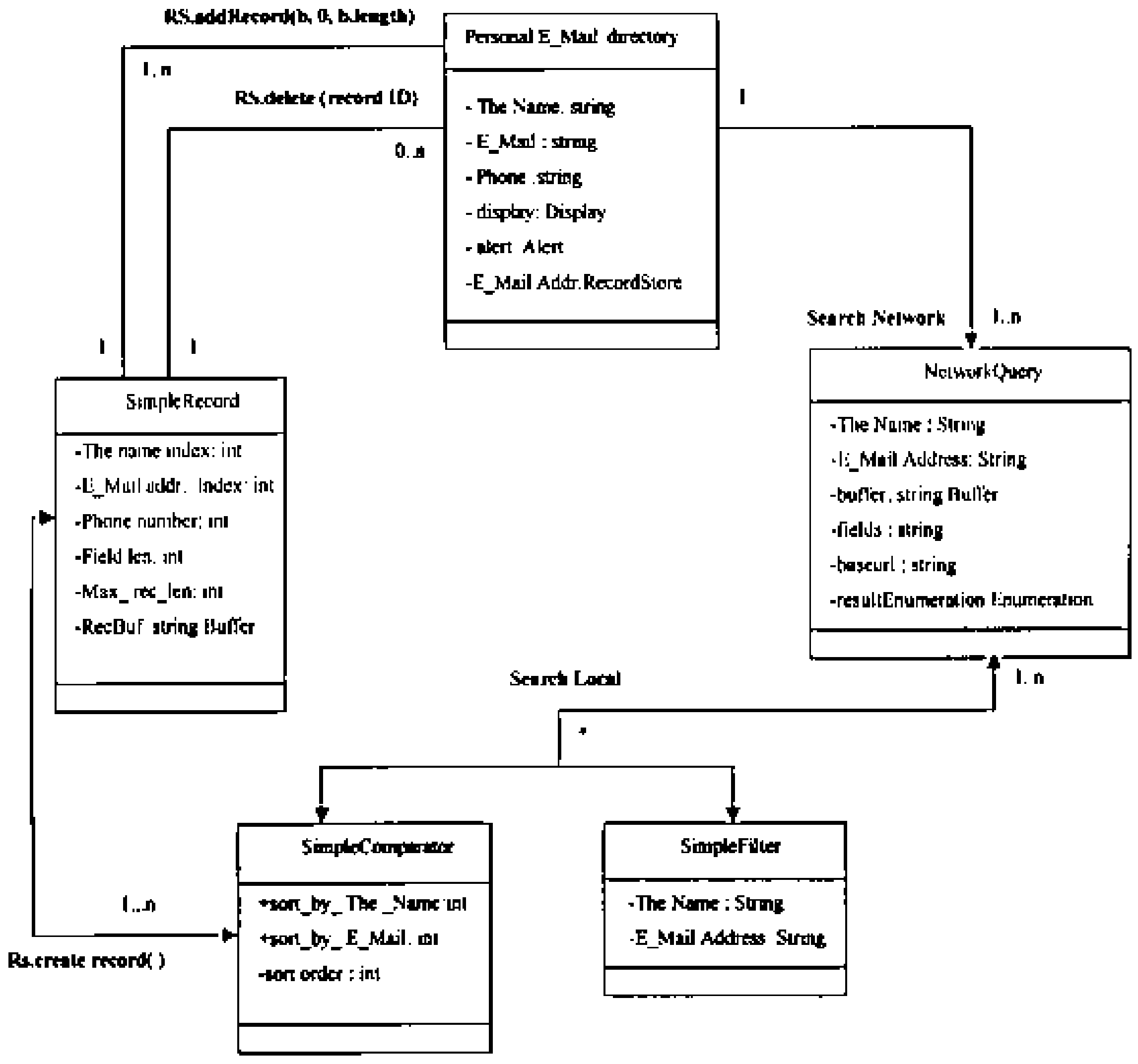


Figure 3.12: Main class diagram

3.6 Object Diagram

The second type of static structure diagram called an object diagram [1]. An object diagram is essentially an instantiation of all or part of a class diagram, this means to create an instance of the class [1]. In our application, the objects are Browse Scr, Entry Scr, AddNew Scr, and Option Order Scr that are instance of Screen class of Personal Email Directory.

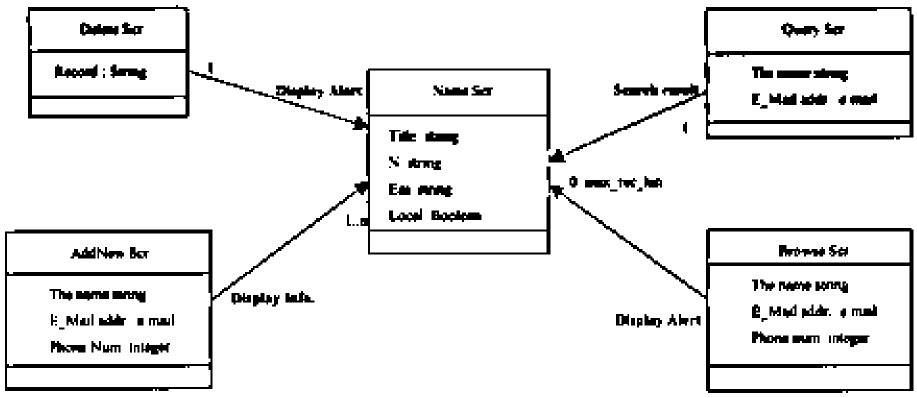


Figure 3.13: Object diagram from Personal E_Mail directory class

In the figure 3.13 as shown, the name Scr object represents the screen that used to display any result for any functions performed by the class as in the Query function we need to display result of searching, in the Browse function must list all records in the screen if there are records or to display alert information if there are no record stored, as in delete and change function we need to display information about delete or add or change record.

3.7 Dynamic Models

Behavioral models describe the internal dynamic aspects of an information system that supports in the system[6]. This Behavioral represented by some of diagrams that describes this behavior of a given system as well as messages between classes. From the group of this diagram in our system we have used as:

3.7.1 UML State chart Diagram

The class diagram captures much of the important system capabilities. Another graphical model, called the state chart. This model is often used to document more complex behavior of a class. A state chart shows the states a class might be in a system and the action or conditions that cause a class to make a transition from one state to another. UML state charts are not normally needed. They are needed when the class has a different reaction dependent on its state [1, 33].

State chart can be viewed as extensions of the class diagram. And you could conceivably create one state chart for every class in our class diagram; the states are depicted by boxes. The transitions are shown as arrows between boxes. Showing that a class in one state can change to another state there is a start and an end state depicted by the solid dots [12].

State chart or state transition diagrams are also used for modeling real-time system requirements, where shows the sequence of states a class goes through during it's life cycle in response to stimuli, together with its responses and actions where the class have behaviors and state [14]. This state depends on its current activity or condition, where the state diagram shows the possible state of the class and the transition that cause a change in state [14]. The representation of a UML state diagram with the three aspects of a state, event, and predicate are distributed over the UML diagram. For example, in our application, the state Query function is entered if the present state is select function and the predicate {select] is true .when the state query has been entered, action query Ser is to be carried out.

In our system : the E_Mail directory system move through a sequence of states based on the input from the user (external events) and internal system actions such as Record Store. The key elements of an UML state chart diagram are: states, transitions and optionally an initial state and final state. States are connected via transitions that control the transfer of the system from one state to another [18], we have shown that by:

1-state chart for run application:

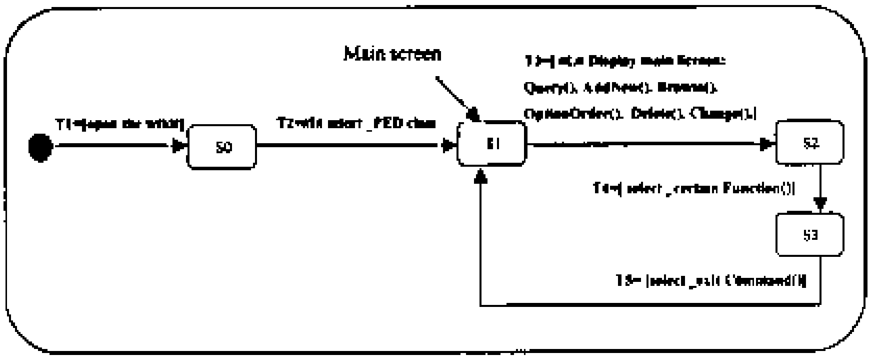

Main screen
T1=[open the white]
S0
T2=[it select _FED class
S1
S2
S3
T4=[select _certain Function()]
T5= [select _exit Command()]
Query(), AddNew(), Browse(),
OptionOrder(), Delete(), Change()]

Figure 3.14: State chart for run application

this state is used during the run of the application starting with opening the main class to display the screen that contain the selection of functions that will be applied. This choices are Browse, Query, Addnew, Delete, And Change as selected list, To exit from any screen select cancel command to return to the main screen.

State Sequence:

S0=project class="open" and screen="close"

S1=project class="opened" and screen=" Personal E_Mail Directory class"

S2= project class="opened" and screen="main Scr"

S3= project class="opened" and screen="certain selection" and command="exit"

2- State chart for Adding New Record:

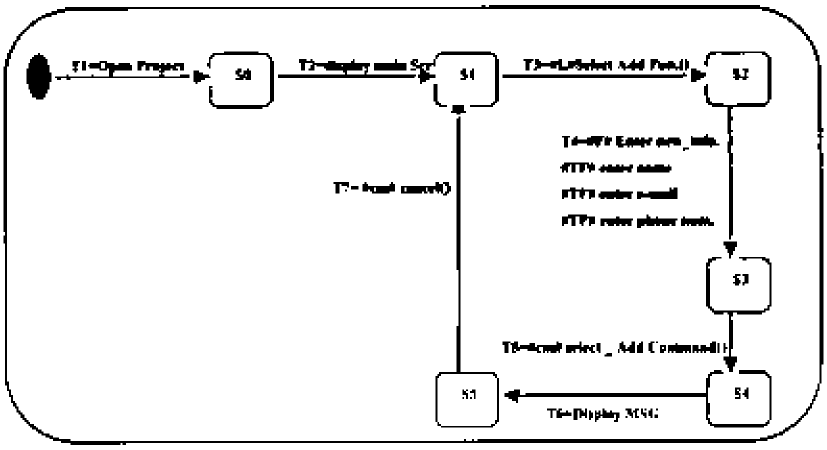


Figure 3.15: State chart for adding New Record

In this figure, the transition between the starting state to the next by selecting Add New subclass from main screen where will present three text field to inter the data, The name, E_Mail Address, and phone number and then select add command for adding a Record to Record Store, When this task is completed the system will show message illustrating that record added.

State Sequence:

S0=project class="opened" and screen=" main Scr"

S1=project class="opened" and selection=" AddNew function"

S2= project class="opened" and(screen=#TF# "Enter the name","Enter E_Mail address", "Enter phone number").

S3= project class="opened" and screen=" AddNew Scr." and command="add"

S4= project class="opened" and screen="alert info."

S5= project class="opened" and screen="main Scr." and command="cancel"

3- State chart for Display E_Mail Directory:

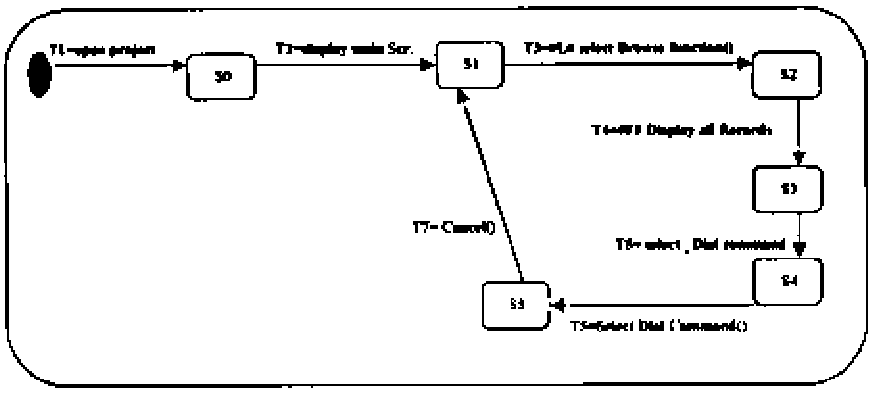

T1=open project
S0
T2=display main Scr.
S1
T3=[illegible] select Browse function()
S2
T4=[illegible] Display all Records
S3
T6= select , Dial command
S4
T5=Select Dial Command()
S5
T7= Cancel()

Figure 3.16: State chart for Display Records

The figure 3.16 illustrate how to create Display all Records stored in Record Store by select Browse function from main screen as a starting state and then follow the another activity for this task by the transition between task, a Response for this selection, browse will be activated for all information concerned with each record. A message (**no names found**) will be shown if there are no names stored.

State Sequence:

S0=project class="opened" and screen=" main screen"

S1=project class="opened" and selection=" Browse function"

S2= project class="opened" and screen=" Name scr."

S3= project class="opened" and screen=" Name scr." and command="Dial"

S4= project class="opened" and screen=" Dialing"

S5= project class="opened" and command="cancel" and screen="main screen"

4- State chart for Query:

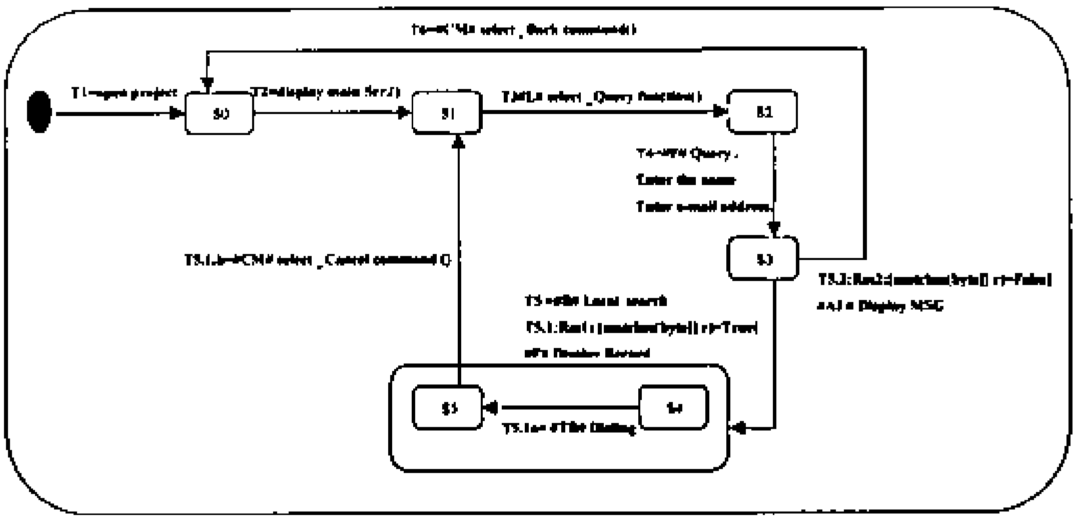


Figure 3.17: State chart for Query function

To create a search for a certain Record : via select Query function from the main screen as a starting state in the state chart , Responding for this state , the system display two (code) text box , one for the name we want to Query about it, and one for E_Mail address, as a result from these transition the system will show that Record based on condition either this record is found or not found in the record store, If the result of searching is true based on a method matches byte of the class simple filter, it will display that record while it can be doing dialing for a phone number, else a result is false a system show messaging that record not found . For return to main screen for another chosen, select cancel command from the dialing screen and back of the searching screen opened now.

State Sequence:

S0=project class="opened" and screen="main screen"

S1= project class="opened" and selection="Query function"

S2= project class="opened" and (screen=#TF# " The name"," E_Mail address")and search ="local" and [matches(byte[] r) =True] .

S3= search ="local" and [matches(byte[] r) =False]and command="Back" and screen="main Scr.".

S4= project class="opened" and screen="Result" and command="Dial"

S5= project class="opened" and screen="Dialing" and command="Cancel"

5- State chart for Delete:

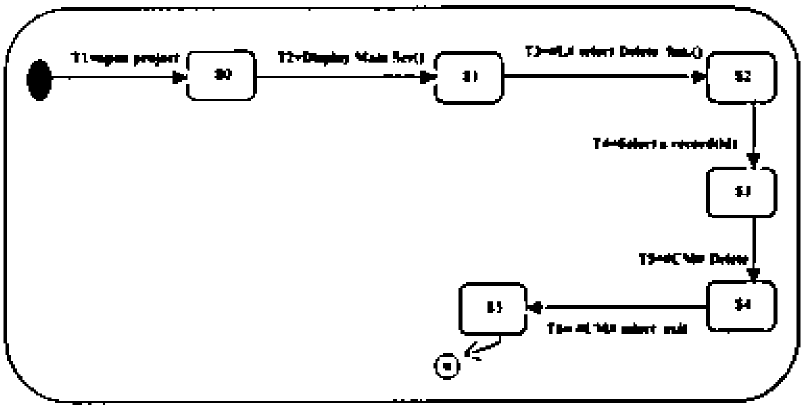


Figure 3.18: State chart for Delete function

These states illustrate how to create Deleted for a certain record existing in record store for E_Mail address directory. In this case we will select delete function from a main screen and then determine a record that we want to delete. And sure that by ok command and display message that illustrate that record deleted.

State Sequence:

S0=project class="open" and screen="close".

S1= project class="opened" and screen="Main Scr".

S2= project class="opened" and screen="Delete function".

S3= project class="opened" and screen="name Scr".

S4= project class="opened" and command="Delete" and screen= "Alert Info".

S5= project class="opened" and command="exit".

3.7.2 Activity Diagram

Activity diagrams are another type of UML diagram for modeling the dynamic aspects of software. (The other types are sequence diagrams, collaboration diagrams, state chart diagrams, and use case diagrams.) Activity diagrams model the activities or procedural steps in an algorithmic process. Activity diagrams provide support for specifying concurrent control flow of activities of use case. It depict activities that occur in parallel , because of this they are very useful for modeling actions that will be performed when an operation is executing as well as the results of those actions- such as modeling the events that cause windows to be displayed or closed[8,1].

Activity diagrams model control flow from activity to activity. This diagram is used to model activities before objects are identified and when it's not important to assign activities to objects [14, 12]. Applying activity diagrams used to describe user task models. As illustrated in the figure 3.19.

In activity diagram, a solid dot represents the start of the process; a solid dot inside a hollow circle represents the termination of the process. The decision activity used when there are condition for a certain activity to be selected, and the synchronization bar allows us to depict activities that can occur in parallel and can be performed in any order, but it have to be performed before the next activity [1].

In the figure3.19 each activity contains another sub-activity that must performed to complete the activity that are selected, the activities: Query, AddNew, Browse, and delete can be executed only when we open the main class to display main screen that content these activity. First the activity Browse is used to display all records stored in the directory based on the option order determined from a user with this activity we have to use dialing activity to phone number for a record selected, if there are no recorded stored before, the system will show alert information for a user about that not record or names found in the directory and return for the main screen to use another choosing for complete the other functions. Second query activity that allow the user to search in the directory about a record either exist or not exist by enter the name of person and E_Mail address and then the searching is local ,the system request for this activity by display this record if exist and display alert information if this record does not exist. The third activity is add New record for a record store by enter the name ,E_Mail address, and phone number and then select add command from this screen to adding this record for the directory, the system response to this task by display alert information for this record added. The fourth activity option order that allow a user to select the order to display records in the Browse function or in query function, this option is select the order by the name field or by E_Mail address field. Fifth activity delete for a certain record by determine this record and then select delete command to complete the task, The transitions in activity diagrams are inter-class transitions; such as those transitions between classes describe interaction behaviors [10]. Activity diagrams describe the workflow behavior of a system. Activity diagrams are similar to state diagrams because activities are the state of doing something. The diagrams describe the state of activities by showing the sequence of activities performed. Activity diagrams can show activities that are conditional or parallel.

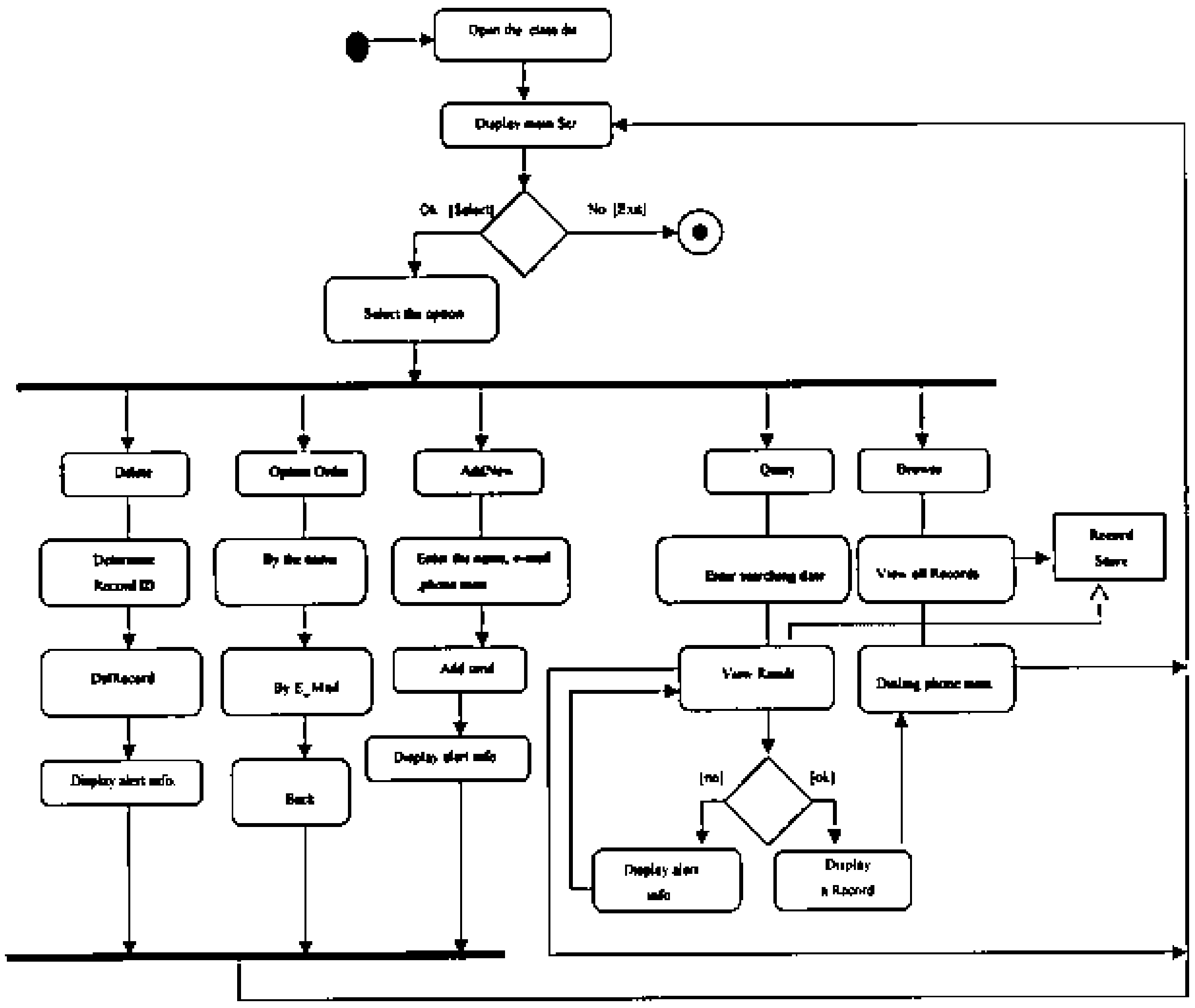


Figure 3.19 Activity Diagram for PersonalE_Mail Directory Application

3.8 Interaction Object Behavior

Objects are related to activities using object flows. Object flow is basically used for indicating which objects are related to each activity, and if the objects are generated or used by the related activities. Activities that are action states and that have object flows connected to them can describe the behavior of related objects since they can describe how methods may be invoked on these objects [21]. In UML there are some of specialized object flows for interaction class can have within a related activity as follows:

1. An<<interacts>> object flow relates a primitive interaction object to an action state, which is a primitive activity[22]. Further, the object flow indicates that the action state involved in the object flow is responsible for an interaction between a user and the application. This can be an interaction where the user is visualizing the result of an object operation. The action state in the view result activity in the figure 3.20 is example of display the result of local searching.
2. A <<presents>> object flow relates a RecordStore to an activity. It specifies that RecordStore should be visible while the activity is active. In the figure 3.20 the Record Store and its contents are visible while the local searching activity is active.
3. <<Confirms>> object flow relates an action confirm to specify search domain. In the figure 3.20 the local searching activity is responsible for finishing the search operation if this domain is matching with the domain defined in the RecordStore.

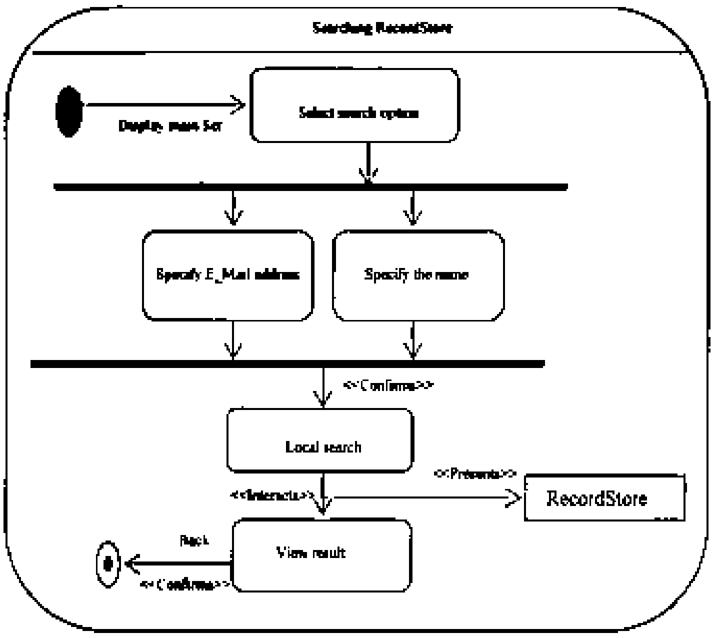


Figure 3.20 Object flow of local search activity

Chapter Four

Design phase

Previous chapter described the tasks that must be performed to complete the analysis phase by using number of UML diagrams. This chapter is to decide how to build the application, the major activity that takes place during design phase is evolving the set of analysis representation into design representation [1].

The design includes activities such as designing system inputs, system outputs and user interface such as form, selection screen that will help the system design meets the needs of its users.

In this phase, we will discuss other types of UML diagrams that are used to complete analysis phase in chapter three. From these diagrams interaction diagrams (sequence and collaboration diagrams) and package diagram to be transferred to component diagram as coding for the applications.

4.1 Interaction Diagram

The primary differences between the class diagram and the interaction diagram is that the class diagram describes structure of the system and the interaction diagram describes the behavior of the system [23].

4.1.1 Sequence diagrams

A pattern of interaction among objects is shown on an interaction diagram. Interaction diagrams come in two forms based on the system, this are: (Sequence diagrams and Collaboration diagrams). The sequence diagram show the explicit sequence of stimuli and based on the real-time specifications, In other word, A sequence diagram shows an interaction arranged in time sequence by their "life lines " and the stimuli that they exchange arranged in time sequence (as a set of messages between objects) within a collaboration to effect a desired operation or result. A collaboration diagrams show the relationships or participants among classes or objects when interacting with each other and are better for understanding all of the effects on a given class and for procedural design. [12].

4.1.1.1 Sequence diagram notation

A sequence diagram has two dimensions [1]:

1. The vertical dimension represents the time.
2. The horizontal dimension represents different objects.

4.1.1.2 Life line

In sequence diagram an object life line denotes the object playing a specific role. Arrows between the life lines denote communication between the objects playing those roles. Within a sequence diagram the relationships among the objects are not shown. The role describes the properties of the objects playing in a diagram [10].

4.1.1.3 Messages

Each object can send and receive messages that define information sent to other object to tell that object to execute one of its behaviors where the message is a function or procedure called from one object to another object[1]. In the application, If a user wants to add new Record, the system will send an adding message to the application, and then the directory class will receive the addition and gives what is need to do the addition that new record add into the record store (i.e.; the behavior) and send the message (record added).

4.1.1.4 The Personal E_Mail Directory Sequence Diagram

The PED Sequence diagram shows how messages are sent from one object to another. A sequence diagrams for all interactions that appears in the application can be created by the following:-

- **Create the first screen MIDlet**, Where this screen is a list of choices that is used in the application. Figure 4.1 illustrated below show the main screen that contains the objects (**Query Scr – Browse Scr – AddNew Scr – Sort order Scr – Delete Scr**) contained in the Screen class defined in J2ME, each of them perform a sequence of certain tasks specified for all records stored in E_Mail directory. All messages between objects interact from one or more object to another object to complete this function where these events sequentially input from an external source (user) to the system. Events are related by time with the top events occurring, these events are the cause of the system response by sending messages.

each message sent from first object are ending after the task corresponding with that object finished to transfer the active to the next object where the message select function (case 0) will be ending when a user select the Query function for a record in the specific time ending when the new message is sending the new object. Back message returned to the main screen object to select another function, all these messages are similar to the other object but they are different in the case of selecting the function.

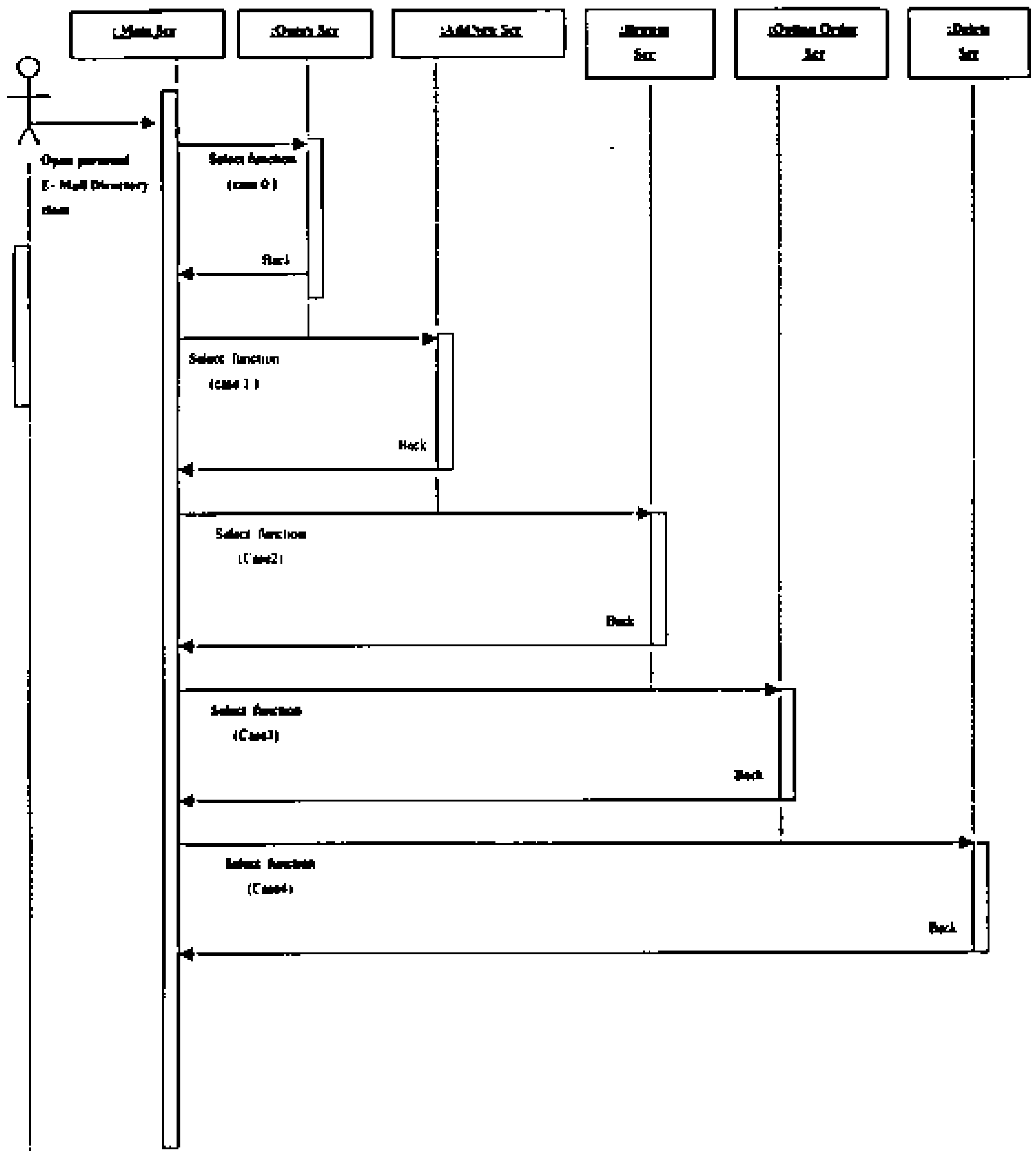


Figure 4.1: sequence diagram for main Screen object

In the Sequence diagrams the vertical dimension of the diagram represents the time. Time unfolds top to bottom and the dotted vertical lines represent the object's life span [18].

The narrow vertical rectangles that partially overlap the life lines show the period of the life when the object is active.

- Create the sequence Diagram for Query Screen object; this case displays two Text Field one for the name, and one for E_Mail. These are used for querying or searching about a certain record specified from a user where this search is local as shown in the figure below.

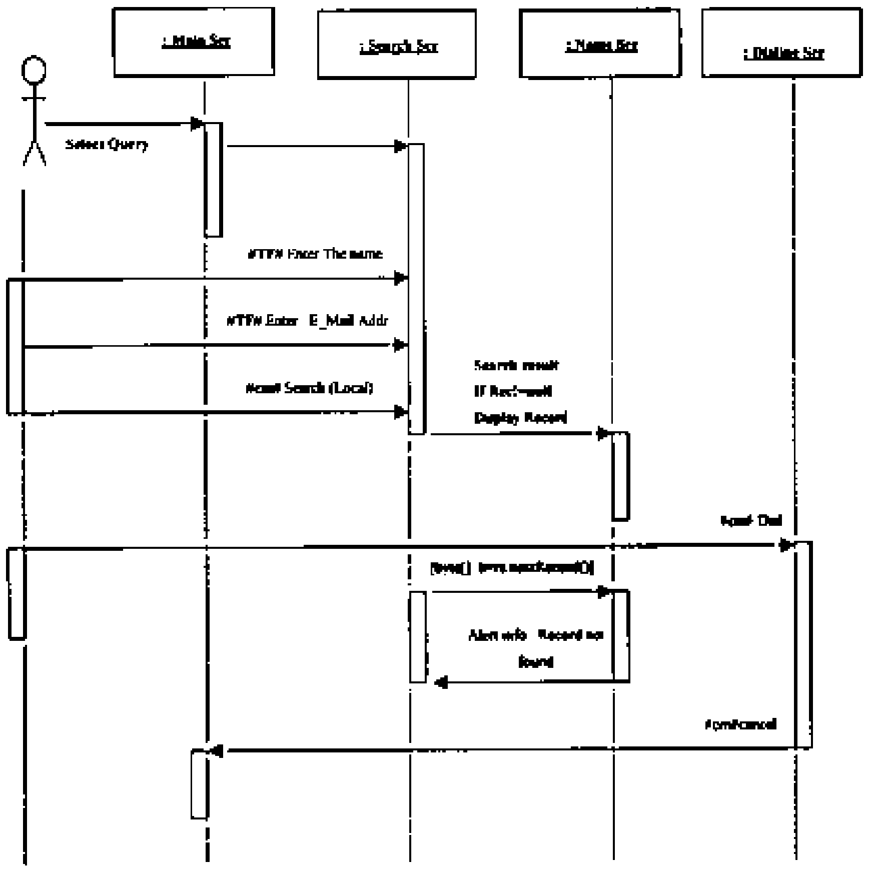


Figure 4.2: sequence diagram for Query Screen object

In the figure above, the select query message is active when a user determines this function from main screen object then the next messages determines the data record to be displayed in the name screen object by using local search, as the result for this task will be displaying a record or displaying alert message to complete this function.

- create a sequence Diagram for sort order screen. This Screen allows us to select sort order to either sorting by the name or by E_Mail Address to browse or query the records. This option used when we want to enter the records to a RecordStore in the certain order.

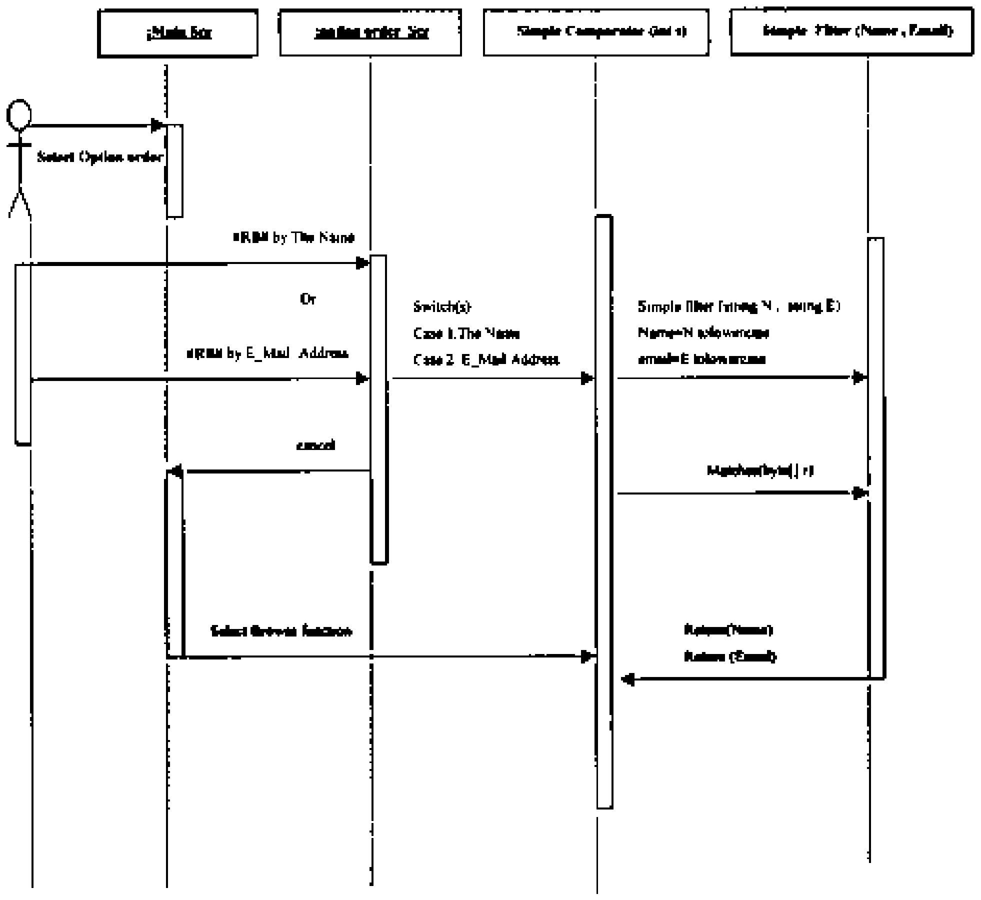


Figure 4.3: sequence diagram for Option Order Screen object

In this case, there are two classes related with Browse screen object and option order screen object:

- Simple Filter: a simple filter is an implementation to the Record Store to filter a private strings one for the name and other for Email. The Simple Filter is a public method Filter allows us to access to the class Simple Filter with string N and string E.

Matches method [byte { } r) takes a record (r) and checks to see if it matches the name and Email set in our constructor of function (method) Simple Filter.

- **Class Simple Comparator**: this class implements the Record Comparator, which can sort the values of the names and the E_Mail addresses using a compare method that takes two records depending on the sort order that extracts and compares the subfields as a two strings, r1 set as the first record which compare r2 as a second record to returns one of the following values:

  Record Comparator. **PRECEDES** if r1 is less than r2, Record Comparator. **FOLLOWS** if r1 is greater than r2 and Record Comparator.**EQUIVALENT** if r1 and r2 are equivalent.

When the user selects the option order to specify the sorting of records stored in record store, then we can select Browse screen as a result or the previous function.

- **create a sequence diagram for Name Scr**, this generates a list of Records that can be called as a result of a browse option on the main screen.

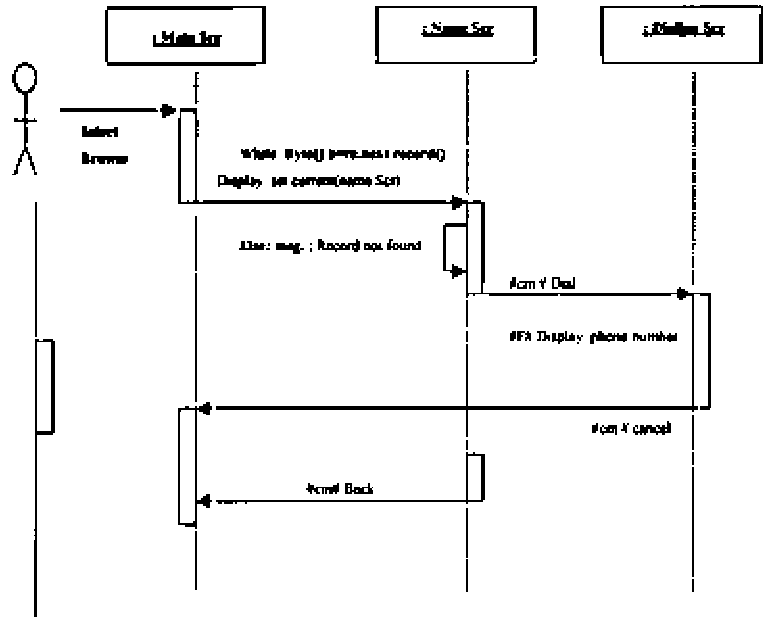


Figure 4.4: sequence diagram for Browse Screen object

When a user select this function, a system will response for that by display all records stored in record store on the E_Mail directory, In the case of no record has been added, a system will show message information which alert that (no records found), other wise a system will Browse all data in each record .

- create a sequence diagram for Add New screen, it displays three Text Field one for The name , one for E_Mail Address , and one for phone number. These are used to add new records to the Record Store.

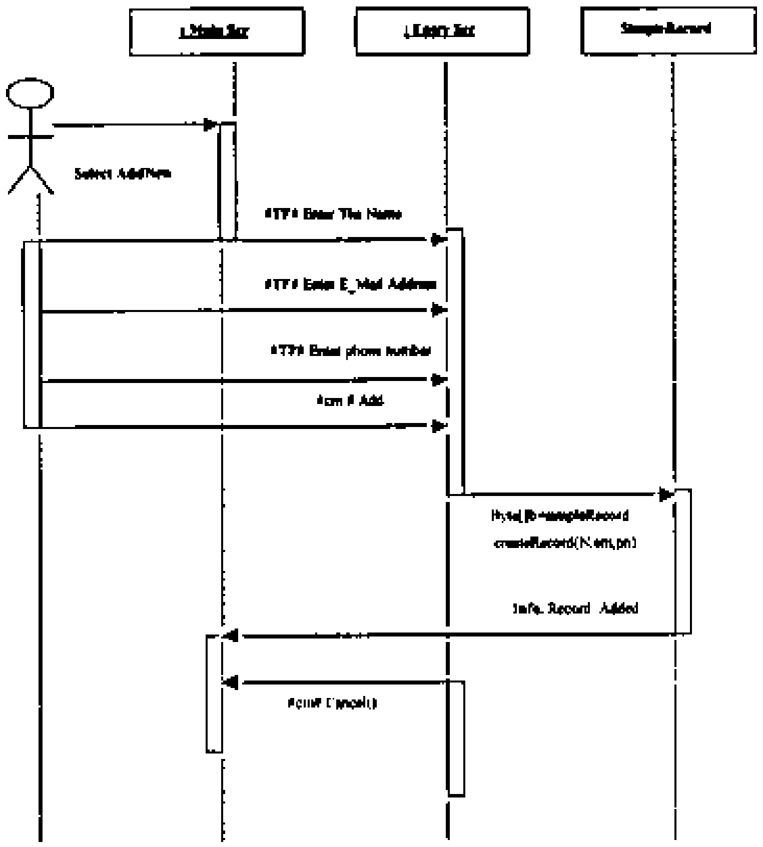


Figure 4.5: sequence diagram for Add New Screen object

In the figure 4.5 the add command represent object of command listener class to be added into the record entered by a user to record store performing that uses the class simple record which provides static methods that allow us to hide the format of a record. Creating a record by determine the index of each field set as (byte []) in record and then it will insert the field in the record buffer (the maximum length of record).

- create a sequence diagram for Delete Record, this will display one Text Field for the ID record in the record that the user wants to deleted, this function will be using the method of rs.deleteRecord(rec ID) to delete the current record from the record store.

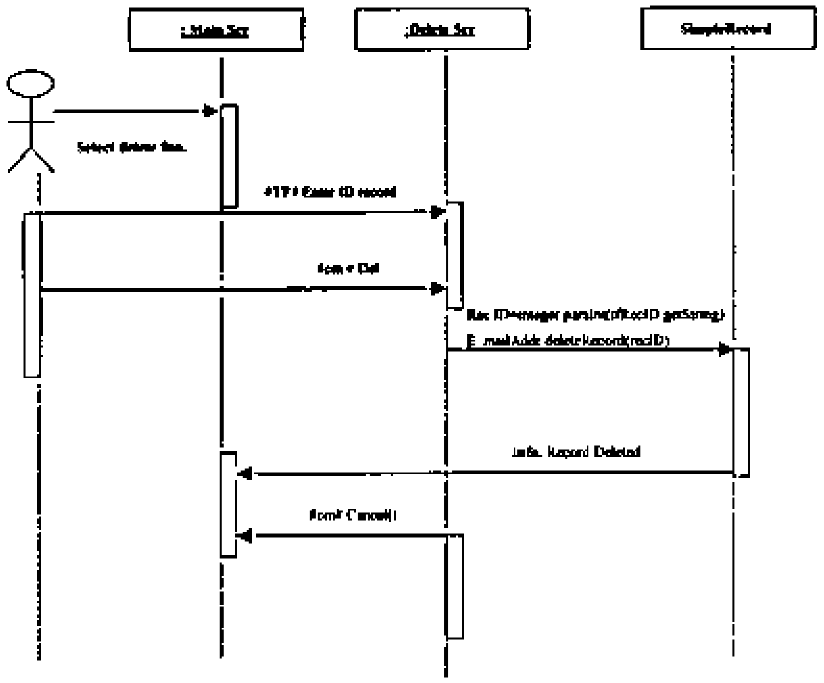


Figure 4.6: sequence diagram for Delete record Screen object

In the figure 4.6 the Del command represent object of command listener class to delete the record determined by the user. When the RecordStore is created the RecordEnumeration enumerated all record stored with ID's, so we can use the method of rs.delete (recID) in this function.

4.1.2 Collaboration diagrams

The order of the interaction in the collaboration diagram is described with a sequence of numbers starting with the number 1, and for subsequence interaction the sequence number can be a nested number. All nested sequence numbers are at the same level of the main numbers in the interaction [10].

4.1.2.1 Patterns

A pattern is a parameterized collaboration; it deals together with the guidelines of using it. A parameter can be replaced by different values to produce different collaborations. The parameters usually designate slots for classes. When a pattern is instantiated, its parameters are bound to actual classes in a class diagram or to roles within a larger collaboration.

A use of collaboration (pattern) is shown as a dashed ellipse containing the name of the collaboration. A dashed line is drawn from the collaboration symbol to each of the symbols denoting class, each line is labeled by the role of the participant between classes in the collaboration, In our application, there are collaborations between two class (Simple Filter and Simple Comparator) when we want to display a list of records as a result of a Browse or Query selection, these classes participant to select the order of display from the simple Comparator class and takes the data from Simple Filter class as shown in the below figure:

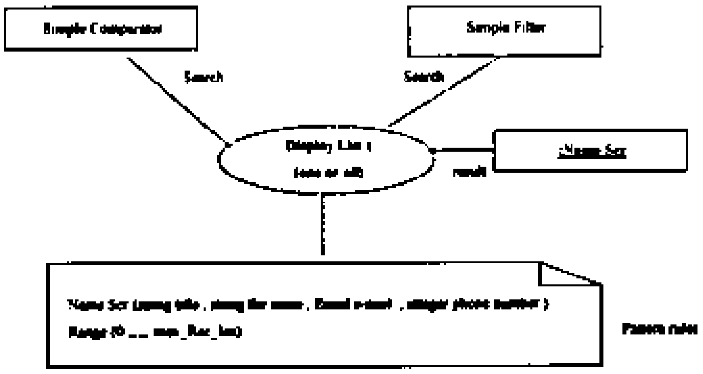


Figure 4.7: pattern usage

4.1.2.2 System Collaboration Diagram for Personal E_Mail Directory Application

We can create a collaboration diagrams for all events with interactions that appears in the application as the following:-

- Event1: Use an application.

UseCase: Open Personal E_Mail Directory from the main scenario:

1- The user selects Open Button in the WTKit window.

2- The user selects Personal E_Mail Directory class from the list.

3- The system displays the main screen to select the application for using.

As shown in the figure 4.8 which illustrates how each interaction between objects performed with a sequence number on this performance, starting with select the function and ending that with sub interaction between them.

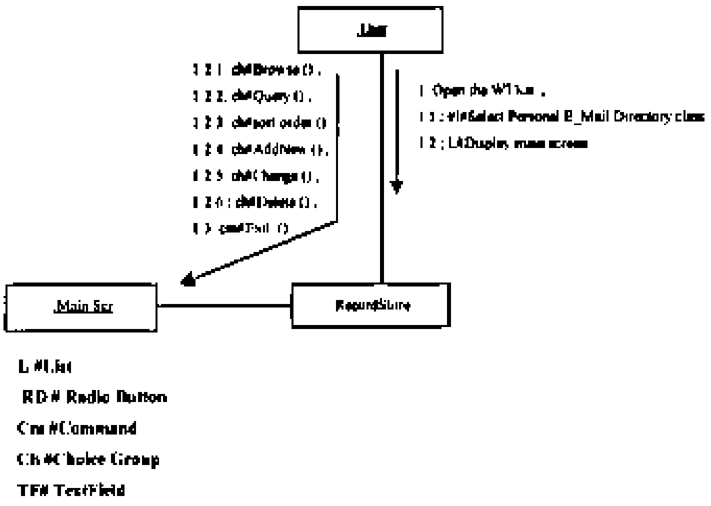


Figure 4.8: Collaboration diagram for scenario application executing

- Event2: get new Record.

UseCase: Add new E_Mail. Main selection.

1- The user selects AddNew choice from the main screen.

2- Dir. Knows that user need to add new E_Mail for RecordStore so that it will displays three text filed to inter the data, one for the name, one for E_Mail address, and one for phone number then it will adds the new record were entered by the user into the record store and inform the user that the task was completed via displaying message (record added).

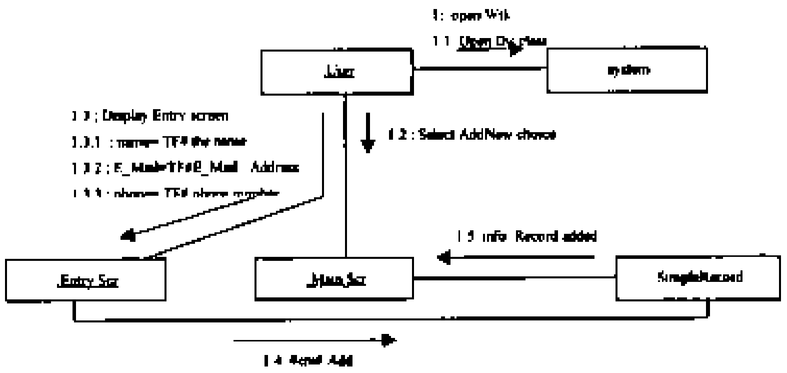

1: open Wik
1.1 Open Dir class
User
system
1.3 ; Display Entry screen
1.3.1 : name= TF# the name
1.3.2 ; E_Mail=TF#E_Mail Address
1.3.3 : phone= TF# phone number
1.2 : Select AddNew choice
1.5 info Record added
Entry Scr
Main Scr
SingleRecord
1.4 Send Add

Figure 4.9: Collaboration diagram for Add New Record

- Event3: we want to see information about certain Record.

UseCase: Query E_Mail information from the main selection.

1- The user selects a Query choice from the main screen.

2- Dir. Knows that user needs to display certain record which was determined by the user, then dir. Class shows it if it exists, and displays the alert information that alerts the record not found.

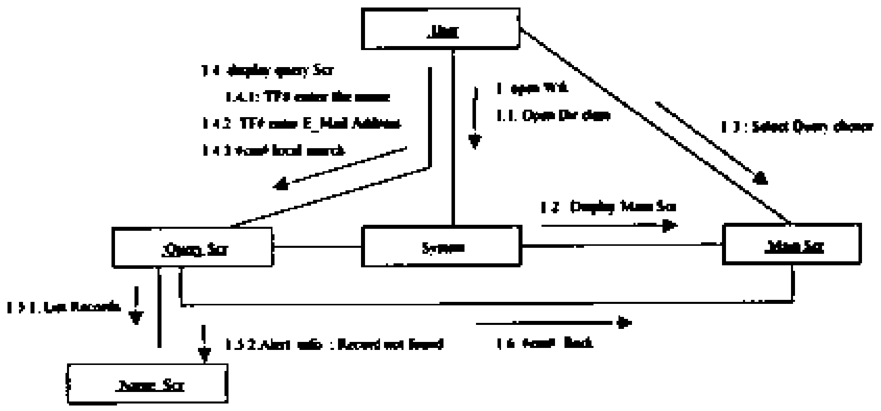


Figure 4.10: Collaboration diagram for Query Screen object

- Event4: a list of all Records stored in RecordStore.

UseCase: Browse all Records in the directory from the main selection.

1- The user selects Browse choice from the main screen.

2- Dir. Knows that the user needs to display all of its contents from the record store.

3- The dir. Class lists on the Name screen object, the phone number we can displays by the dialing command.

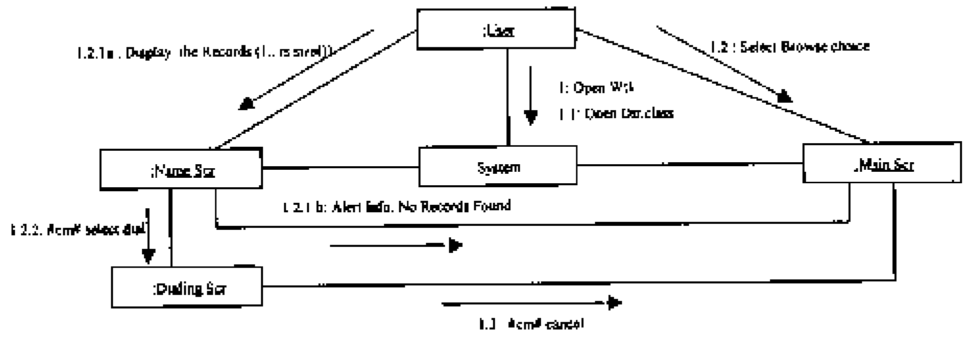


Figure 4.11: Collaboration for Browse Screen object

- Event5: We want to Delete an exist Record.

UseCase: Delete a certain Record from the main scenario.

1- The user selects the Delete choice from the main screen.

2- Dir. Knows that the user needs to delete an existing record from the directory by giving a Record ID number.

3- The dir. Class sends a message to ensure that the Record has been deleted to inform the user that the task was completed.

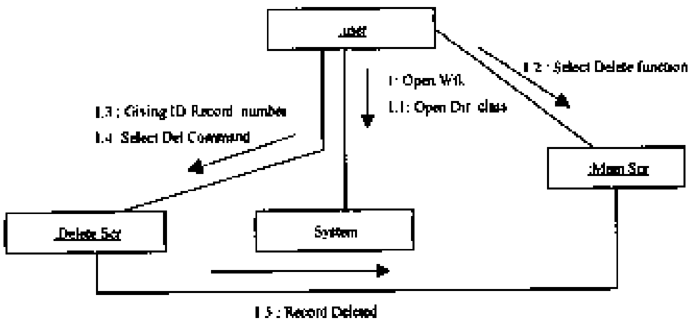


Figure 4.12: Collaboration for Delete function

- Event6: We want to select an option order from displaying all records in a certain order.

Use Case: Browses Records in certain order from main selection.

1- The user selects an option order choice from the main screen.

2- Dir. Knows that the user needs to select one of the following options sets to the use either Query or a Browse Screen.

3- The user selects the option to be sorted by names or by E_Mail addresses and select cancel command to return to main selections.

4- The user then selects Browse choice from main screen again to display all records or a Query choice to display one record at time.

5- As a result of this task the Browsed Records in that order can be dialing of phone number as well. Figure 3.13 show diagram for this task:

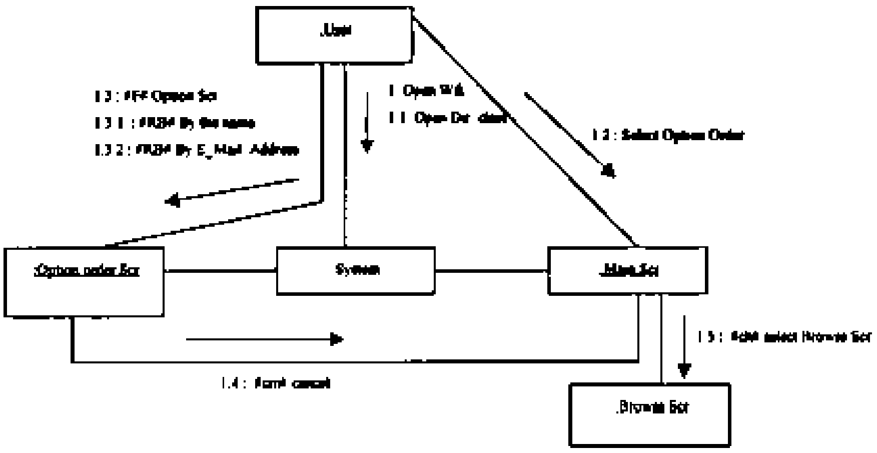


Figure 4.13: Collaboration for option Order scenario

4.2 Details class diagram

The term method means something that the object knows how to deal with in actual world, it also reflect the behavior or the responsibility of that object as well. Many object-oriented developers use this term, but UML also uses the term Operation. We will use the term "Method" in our application.

Methods or UML operations means procedures in a certain case. Methods of classes are those procedures that objects in one class knows how to apply a certain event in specify state [7].

An action can be assigned to an object that sends a message to another object which applies the method [7]. The state variables of an object should be declared private (accessible only within an object of that class) or a protected which is (accessible only within an object of that class or a subclass of that class). A detailed class diagram is obtained by adding the actions (methods) to the class diagram of chapter three, section (3.5) and shown in figure 3.12:

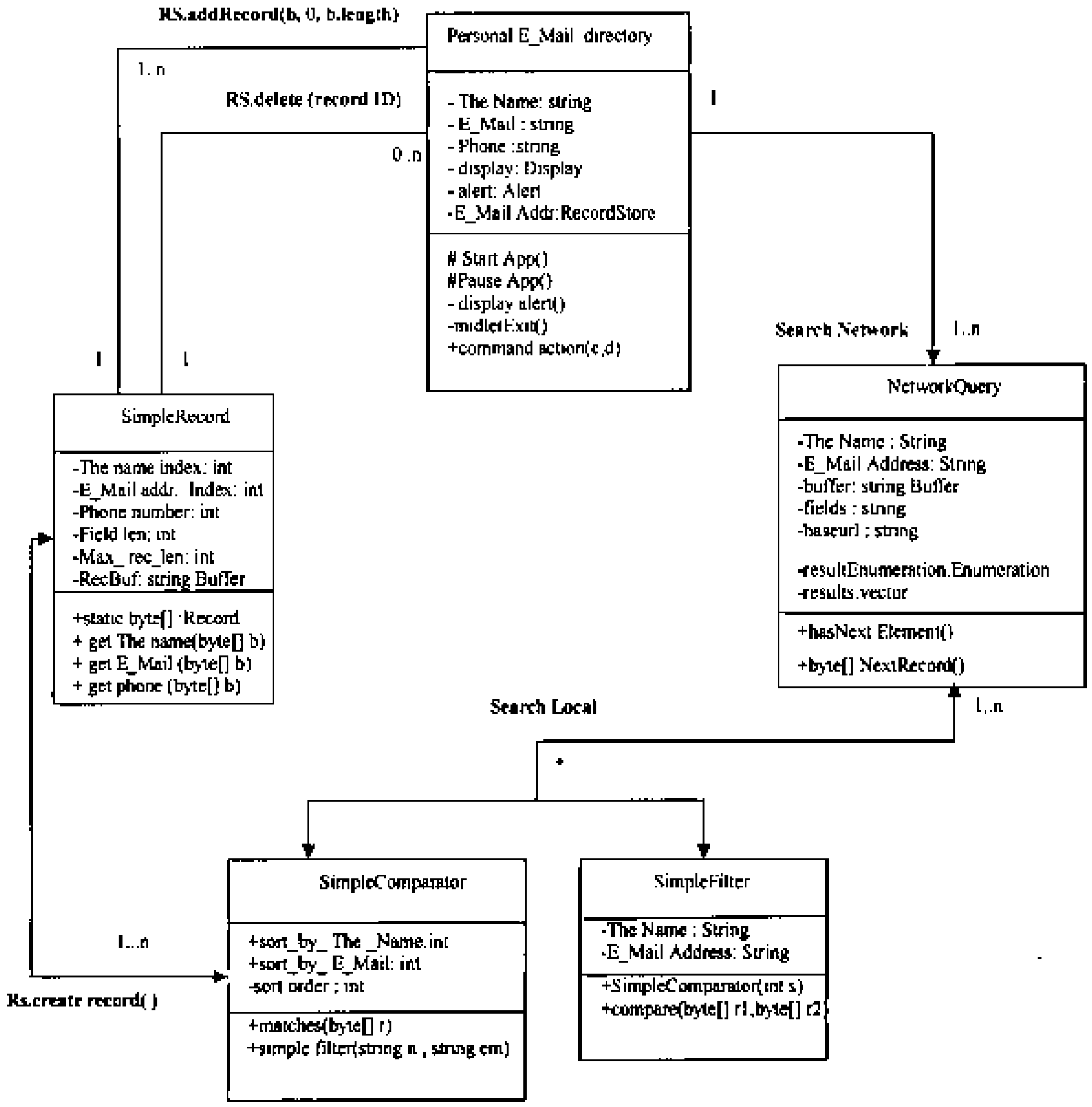


Figure 4.14: Details Class Diagram

4.3 Packages and Package Diagram

In UML simplifies complex classes diagrams via group classes into small packages. A package is a general construct that can be applied to any element in UML models were there are relation between them [1]. The symbol for a package is similar to a tabbed folder depending on where a package is used [34]. A package appears as a rectangle with small tabs at the top where the name is on the tab or inside the rectangle. The dotted arrows between packages are dependencies of the relationships [26].

In other word, package diagrams provide a mechanism for dividing and grouping model elements (e.g., classes, use cases).

- In effect, a package provides a name space such that two different elements in two different packages can have the same name.
- Packages may be nested within other packages.
- Dependencies between two packages reflect dependencies between any two classes in the packages. For example, if a class in Package A uses the services of the class in Package B that means Package A dependents on Package B [1].

4.4 A Package Diagram for Personal E_Mail Directory

In our applications a package diagram represents all sub packages used in the application design from the main class such as (BrowseScr, QueryScr, AddNewScr, Option OrderScr, DeleteScr) where these sub packages contains another internal classes such as text box, form, screen, and text field that are defined in Java Libraries which are used in our application, figure 4.15: shows a package diagram that provides an overview of the whole system, this package diagram shows dependencies between several components of the system [14].

The dependencies between sub packages such as BrowseScr, QueryScr are to for instance <<import>> with NameScr to display results, and that is for example to <<access>> between the main class and the Simple Comparator package, SimpleFilter package, SimpleRecord package, and NetworkQuery package.

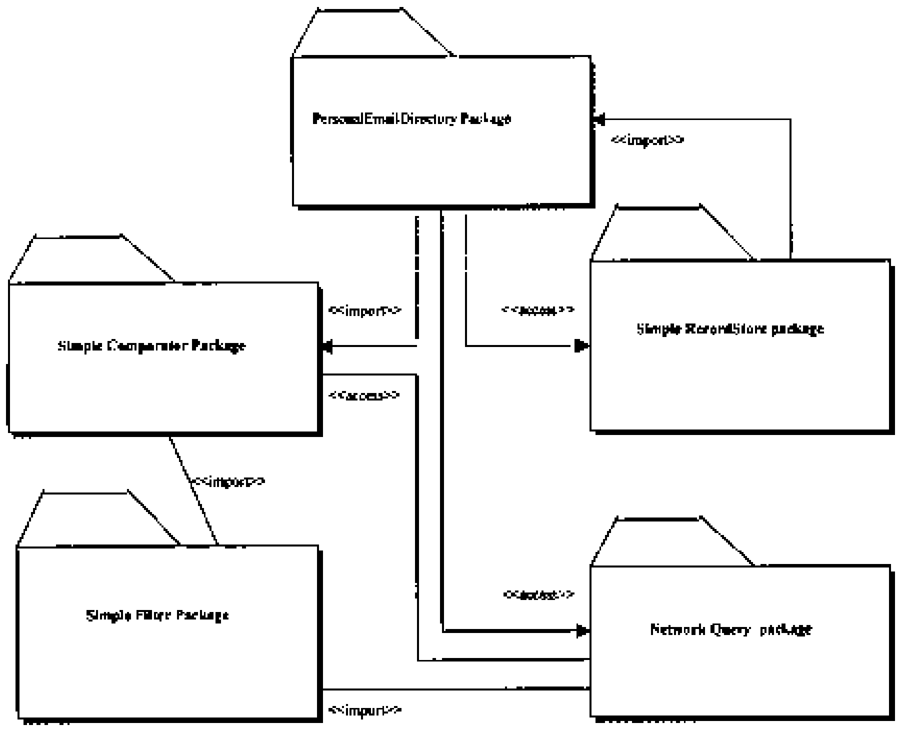


Figure 4.15: Packages and their access and import relationships

Figure 4.15: shows the dependency from package E_Mail Dir. to packages Simple Filter, Simple Comparator, Network Query, and Simple RecordStore implies the elements in sub-packages in E_Mail Dir. that depends on elements in the packages such as SimpleFilter, SimpleComparator, NetworkQuery, and Simple RecordStore. In AddNew sub-package we will use SimpleRecord to add a new record to Record Store which may displays the results in the Name Scr sub-package which is an external

element of the E_Mail Dir. package as in the Browse and Query sub-package. We will also use SimpleFilter package to return the data and SimpleComparator to determine the order, in the Query sub-package we will use NetworkQuery package to find out about the record in the RecordStore, and the same process happened when we apply for Delete sub-package as well.

4.5 Physical Diagrams

4.5.1 Component Diagram:-displays the high level packaged structure of the code itself. Dependencies among components are shown, including source code components, binary code components, and executable components[1]. Some components exist at compiling time or link time, and run times as well and it may happen more than one time [1].

We will show the component for each class used in the application and the interfaces between them in the <<select>>operation from the main class Personal E_Mail Directory, and in the Inter components inside it. It is the same in all of SimpleFilter class, NetworkQuery class, SimpleRecord class, and SimpleComparator class. Each component is related to the other component using interfaces that represent the operation that will complete these functions. Option order function uses data access interface to return the data from Simple Record from the Browse component. Match interface is used to compare the data entered in the Query component with the data that is in the Simple Filter component. Add interface is using an AddNew component to add new records to the record store as shown in figure4.16.

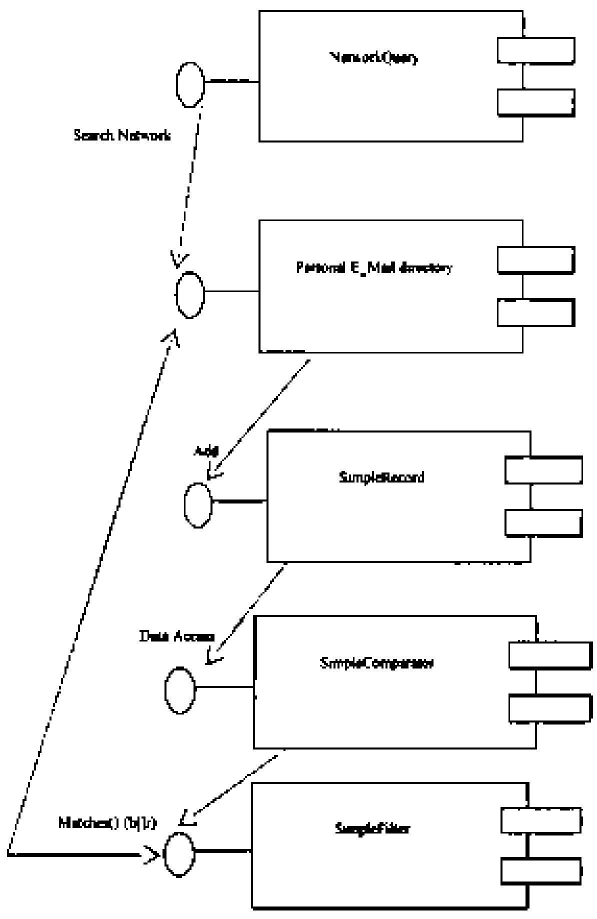


Figure 4.16: Component Diagram and interfaces

4.5.2 Deployment Diagram

This diagram displays the configuration of run-time processing elements and the software components, processes, and objects that live on them. Software component instances represent run-time manifestations of code units.

This diagram displays the components as nodes between machines communication through Bluetooth technology when we run the application on real devices[1].

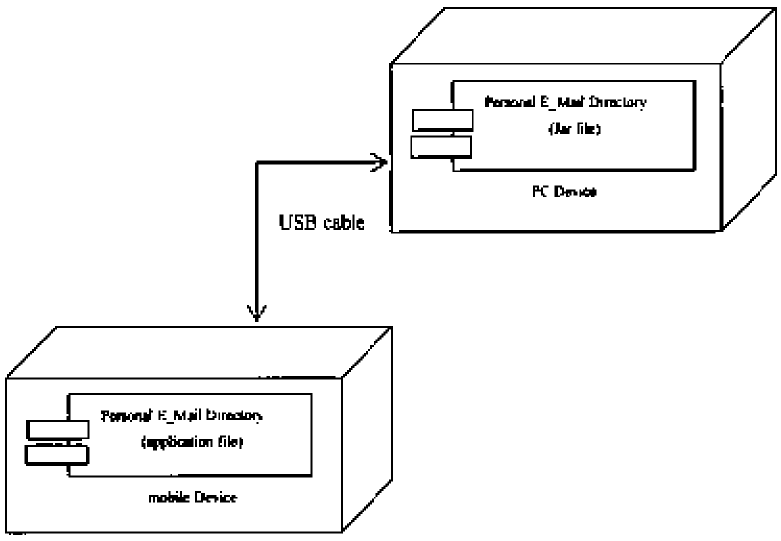


Figure 4.17: Deployment Diagram

Chapter Five

Implementation phase

The previous chapters describe how to build the application using UML diagrams and the interactions between objects, and how it is stored in my implementation. This chapter phase describe how to use J2ME Wireless Toolkit in such an application. The use J2ME Wireless Toolkit was chosen to be my programming Language tool to implement my application via mobile phone and other wireless devices as well[9]. The tool based on Mobile Information Device Profile (MIDP) 2.0, it also supports a handful of optional packages, which made a widely capable development toolkit.

5.1 The Tools in the Toolkit

The J2ME Wireless Toolkit has three main components shown as follow:

1- K Toolbar automates many of the tasks involved in the creation of MIDP applications. It also consists of an emulator that simulates a mobile phone used for testing MIDP applications [11, 9].

2- A collection of utilities provides other useful functionality, including text messaging console at most of the tools.

3- K toolbar is the center of the toolkit. This toolbar can be used to build an application, launch the emulator, and start with J2ME application where the emulator used to be in run time with any application. The only additional tool we need is a text editor for editing source codes [9].

5.2 Toolkit Features

The J2ME Wireless Toolkit supports the creation of MIDP applications with the following main features:

1- Building and Packaging: The main purpose of packaging is to write the source code to be compiled via a compiler. The compiled source will be ready to preverifies the class files, and then packaging a MIDlet application remain ready for execution [27].

2- Running: The MIDlet application will be ready for running directly from the emulator [28].

5.3 Command Listener

Every J2ME application that creates an instance of command class must also create an instance that implements the command Listener Interface. The Command Listener is notified whenever the user interacts with command with the command Action () method [9].

Classes that implement the command Listener must implement the command action method as well, that accepts two parameters. The first parameter is a reference to an instance of the command class, and the other parameter is a reference to the instance of the Displayable class.

5.4 Support Technology

The J2ME Wireless Toolkit supports many standard Application Programming Interfaces (APIs) which could be defined through the Java Community Process as following [13]

- CLDC: Connected Limited Device Configuration [13, 11].
- MIDP : Mobile Information Device Profile [30].
- Bluetooth.

5.5 Developing MIDlet applications

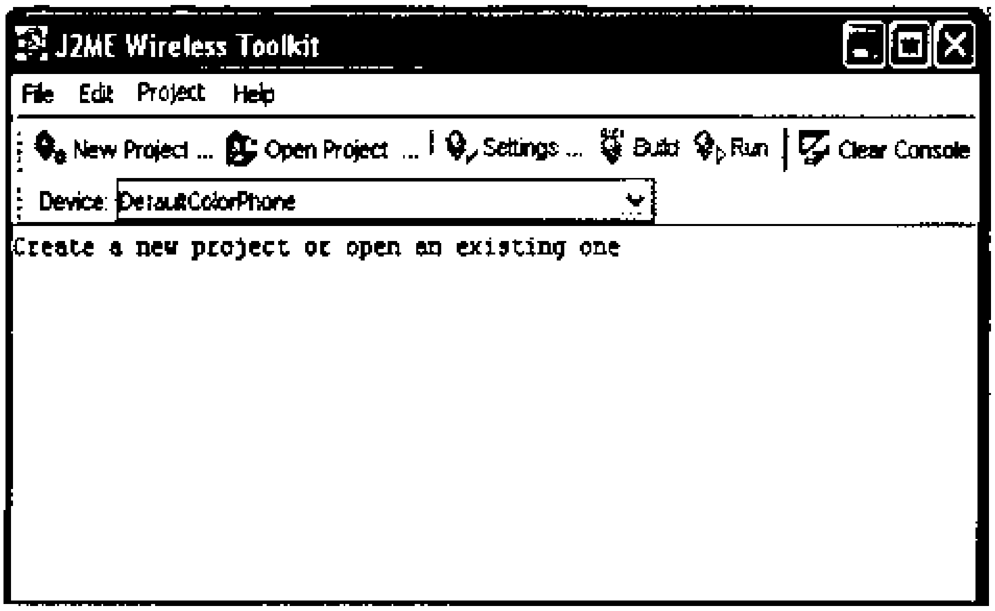
This section describes how we use J2ME Wireless Toolkit to create our application [9].

5.5.1 Personal E_Mail Directory MIDlet

In the J2ME, MIDlets are organized into many Projects. Where the end result of one project is one MIDlet which was presented in our application [9] .

- A project contains all of the files that will be used to build a MIDlet, including Java source files, resource files, and the MIDlet descriptor.
- We may via KToolbar create a new project or open an existing project.
- To open an existing project we may follow the instruction of Personal E_Mail Directory], then we should the start K Toolbar on the window and choose start > programs > J2ME Wireless Toolkit> K Toolbar.

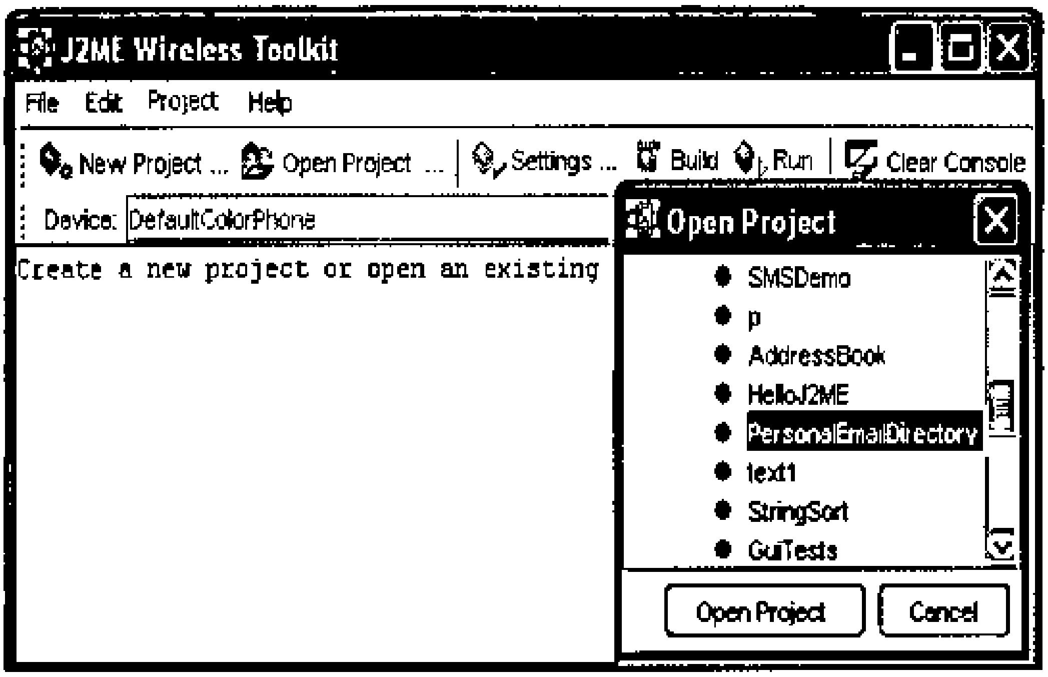
We may see the process shown in the figure as the following:


J2ME Wireless Toolkit
File Edit Project Help
New Project ... Open Project ... Settings ... Build Run Clear Console
Device: DefaultColorPhone

```
Create a new project or open an existing one
```

Figure 5.1 the Toolbar Window

From the above figure we can click on the file to open the Project, Then tool kit will response with a list that indicate all projects stored, we also may Select Personal E_Mail Directory project to display the MIDlet class from this project as the following :


J2ME Wireless Toolkit
File Edit Project Help
New Project ... Open Project ... Settings ... Build Run Clear Console
Device: DefaultColorPhone
Open Project
Open Project
Cancel

Create a new project or open an existing

SMSDemo
p
AddressBook
HelloJ2ME
PersonalEmailDirectory
text1
StringSort
GuiTests

Figure 5.2 open the project

5.5.2 Description of the project Location

In the K toolbar console, we will see some messages telling us exactly where to store the source code and resource files for this project [25].

5.5.3 The simple Development Cycle

The simple development cycle looks like Edit source code >Build>Run[25].

- Edit source code: - to create Java source file that will be used by our application (writing by using notepad editor) [9, 13].
- Build: - means the J2ME Wireless Toolkit compiles and preverifies Java source files and then create (JAD, JAR files).
- Run: - takes the compiled Java class files to run it on the emulator [13].

This step will be discussed in the following section:-

5.5.3.1 Edit Source Code

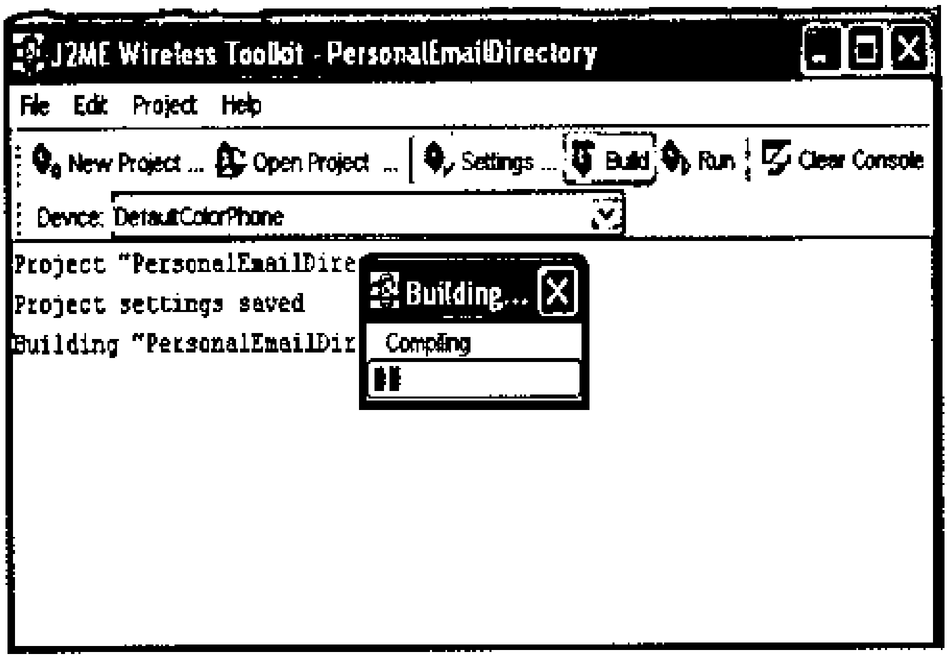
Editing source code is the only step in which the J2ME Wireless Toolkit is no help at all [10]. So that, we will need to use the text editor of our choice to create and edit source code files [20].

The source code should be saved in the source directory of the project under the name (**PersonalEmailDirectory.Java**), which will be located on (**C:\Wtk21\apps\ PersonalEmailDirectory \ src \ PersonalEmailDirectory.Java**) where {Wtk21} is the installation directory of the toolkit.

5.5.3.2 Build

The next step after Edit Source Code is to build Source Code by using K Toolbar window [9], and then click on the Build button. Assuming that, saved Source Code file happened in the right place, the toolkit will find the file and then compile it. Compilation errors will be displayed in the K Toolbar console. If there are errors we can go back and edit the source code to fix them. When eliminated an errors the K Toolbar console tells us that the project was successfully built [20, 9].

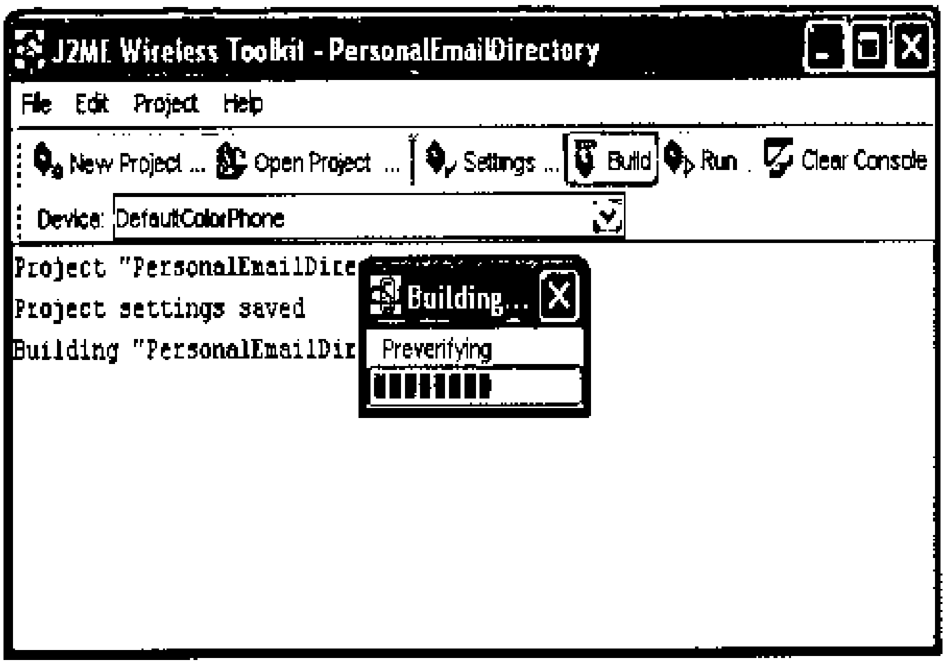
Figure 5.3 will illustrate how personal E_Mail Directory MIDlet will compiled, in this step, the T2ME compiler will be also preverifying it to complete a compilation of the application when there are no any errors.


J2ME Wireless Toolkit - PersonalEmailDirectory
File Edit Project Help
New Project ... Open Project ... Settings ... Build Run Clear Console
Device: DefaultColorPhone
Building...
Compiling

```
Project "PersonalEmailDire
Project settings saved
Building "PersonalEmailDir
```

Figure 5.3 compiling an application

Preverifying it if there are no errors in the project.


J2ME Wireless Toolkit - PersonalEmailDirectory
File Edit Project Help
New Project ... Open Project ... Settings ... Build Run Clear Console
Device: DefaultColorPhone
Building...
Preverifying

```
Project "PersonalEmailDire
Project settings saved
Building "PersonalEmailDir
```

Figure 5.4 preverifying an application

5.5.3.3 Run

After building the project successfully, we will be ready to click on the Run button. This action forces the emulator to shows the MIDlet that concerned with project Personal EmailDirectory to be used on this application.

5.5.4 Project Directory Structure

Each project had a standard directory structure. The project itself is represented by a directory in Wtk21\apps [11]. PersonalEmailDirectory project is indicated in the location of {Wtk21\apps\ PersonalEmailDirectory}, in this directory there are many sub directories with different functionality, these directories are represented as the following:

- Bin directory: -this sub directory consist of the MIDlet descriptor and the JAR file, those are placed in this directory when we package the project.
- Classes directory: - This directory is used by the Toolkit to store the compiled class files.
- Lib directory: - it places a third-party library in this directory to be included in the current project.
- Res directory: - it consists of images, sounds, and other resource files to be placed in this directory when we are packaging it into root of the MIDlet JAR [9].
- Scr directory: - it contained the source code of the currents files in the project directory.
- Tmpclasses and tmpsrc: - this directory is used by the toolkit [9].

5.6 The Abstract Windowing Toolkit (AWT)

The hierarchy structure of the most important classes in the AWT package is shown in the following figure:

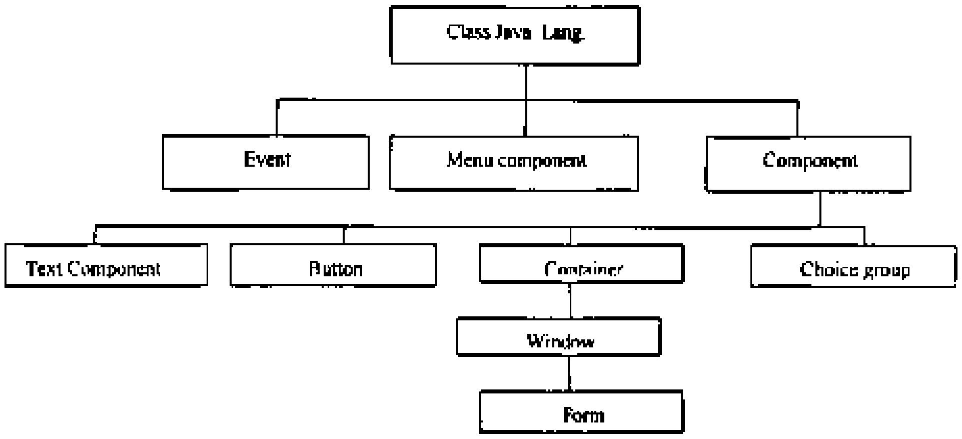


Figure 5.5 the basic components of the application

The component is an abstract class, so that we can create objects belonging to its subclasses not to the components itself, the subclasses that represent these components are as follow: Button, check box, choice group, and text Field. When the user interacts with one of these components some action is required, the component generates an event that our program can detect and react to its [2].This behavior can be demonstrated by using the following components:

- Buttons: are instances used in the {Java.awt} as Button command class, this button are created with a new button labeled with it. This button can be added to screen with the method that been set via command (this)
- Check box: is a component that has two states: checked and unchecked. The user can change the state of a check box by clicking on it. Check boxes are often used to select from a list of possible choices (in our system there are two check boxes the name, E_Mail address). Each check box has a label that should be used to tell the user what the check box represents.

- The list: a list is presented on the screen as a list of choice; each choice could be selected on the list to be an active choice [2].
- Text Field: text field allow us to enter and edit text added by the user [2].

5.7 Using Third-Party Libraries for project

Any library files placed in the project Lib directory will be included in the building and packaging of the project [2]. Libraries should be in the form of JAR file in Java classes.

The J2ME Wireless Toolkit will be installed in the location {c:\Wtk21}, in this location the Personal Email Directory will be created as well. Classes library would be located in go in {**C:\Wtk21\apps\ PersonalEmailDirectory \Lib**}.

When we build, Run, debug, and package the application, the class files in Lib directory could be used [11].

5.8 Defines Classes Used in the application

Figure 5.6 shows our application and the need of (4 Class), one the main class called **Personal E_Mail Directory** which are running the application by using a functions such as: Browse – Search or Query - Delete , option order , and AddNew, where the searching is local.

The MIDlet will be derived from a library called: **Java.microedition.Midlet**, and defined another class called **Main Form** which can displays that display the interface of our proposed system .

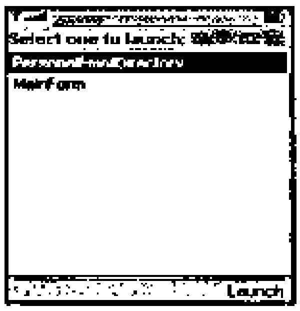


Figure 5.6 Select main class of application

The main class of our proposal system performed the following processes:

1- It can add new names, phone numbers, and E_Mail Address for some people.

2- We can search for a stored record.

3- New records can be added.

4- It displays or Browses all records stored in our RecordStore.

5- We may select a method to order the record browser.

6- It could delete a current record.

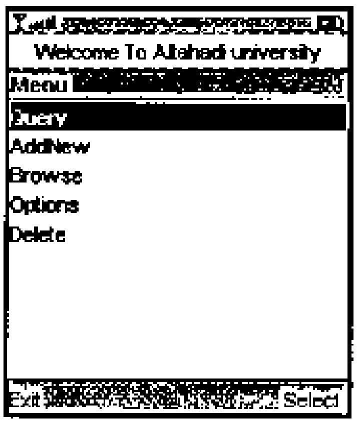


Figure 5.7 List choice group screen

The application had a class named as a Simple Filter: It contains data for both names and E_Mail addresses, and a class named as a Simple Comparator: specify the order of the data.

There is another class named as Simple Record which returns data to a text file.

There are other classes which work on the application such as: display command form, Display screen, Form, and Alert, Derived from the library: **Java.Microedition.Icdui.***.

5.8.1 Classes definition variables

We need to define a set of variables such as the following:

- Private RecordStore E_Mail address represents a text File that the data will be stored it.
- Private static Final int NA_LEN=10 determine a string range length of the name entered as a first field of the record.
- Private static Final int EM_LEN=20 determine also a string range length of the E_Mail address entered as a second field of the record.
- Private static Final int PN_LEN=15 will handle the string range length of the numbers entered as a third field of the record.

5.8.2 Giving an initial values

There are many initial values in my framework these values are set as follows:

- Final Private static int ERROR=0 when we use alert object to display errors.
- Final Private static int INFO=1 is to display info.alert to add any message.
- Private Display display: This object for displaying on any screen using in the class.
- Private Alert alert: This object specifying for an error messages.
- Private List main Scr: Define the main screen as a list of choices for {Browse, Query, AddNew, Delete, Option order}.

5.8.3 Define Command object

- Private Command CmdAdd: used to add record entered to the RecordStore.

- Private Command CmdBack: it was maintained to return for the main screen to select another function [27].
- Private Command CmdCancel: this command is used in the dialing screen for returning to the main screen [27].
- Private Command CmdExit: is to close the MIDlet.
- Private Command CmdSelect: represent the selection of the chosen group.
- Private Command CmdSearch Local: is used with the searching screen to find specified record.

5.8.4 Methods that are used in the main class

- **GetDisplay (this)**: we use this method to display the operation.
- **Open Record Store ()**: This method is used to open the RecordStore that was created to store all records.
- **Close RecordStore ()**: it was used to close a RecordStore [9, 37].
- **MainScr ()**: is used to display the main screen that contains the main functions.
- **MIDlet Exit ()**: is used to close or exit from the application.

5.8.5 Methods and variables for class simple filter

- **Private string The Name**: this method determines the name field that we want to be entered.
- **Private string E-Mail**: determine the email address field.
- **Get the Name**: this method is used to Returned with the name data.
- **Get E_Mail**: Returned with the email address data.

5.8.6 Methods and variables for class simple Comparator

- **Public final static int sort _ by _ the name = 1:** this function is used for the option order (1) for sorting order by the Name.
- **Public final static int sort _ by _ E-Mail address=2:** this function is activated when the selection of option order equal to (2) to sort an order by E_Mail address.

5.8.7 Methods and variables for class simple Record

Giving initial values:

- **Private final static int The_name_index =0:** represent the place of the first field in the record.
- **Private final static int Email_index = 20:** is used to describe the place of this field in the record based on the size of the first field.
- **Private final static int phone_index = 40:** represent the place of the field based of the first and second fields of the record.
- **Private final static int field_Len = 20:** represent the length of each field.
- **Private final static int max_Rec_len = 60:** represent the maximum length that can be giving for each record.

5.9- Packaging application

The J2ME Wireless Toolkit automates the task of packaging the MIDlet. The end result of Packaging will be representing into two files, which are the MIDlet descriptor (JAD) and the MIDlet JAR file (Java Archive) [9].

The descriptor is a small text file that contains information about the MIDlet (application). The JAR contains the class files and the resources that made up the MIDlet. To package the MIDlet, choose the tool from the following location

Project>Package > create Package for the project opened from the K Toolbar menu. The JAD and JAR will be generated and placed in the bin directory of the project created [9, 13]. Figure 5.8 show how the packaging works:

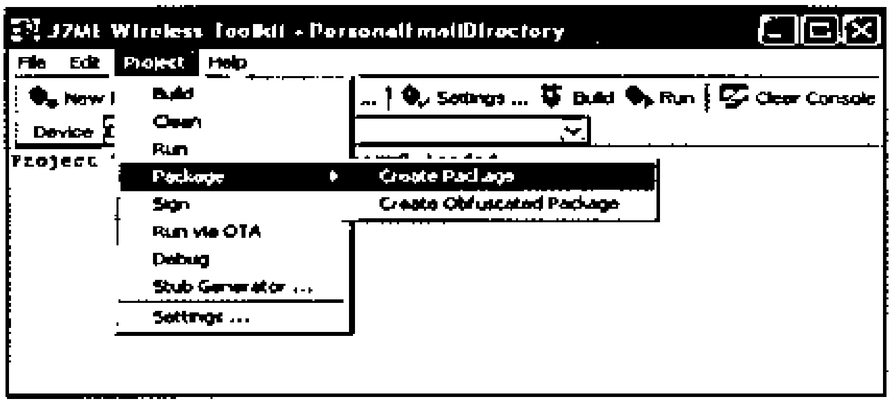


Figure 5.8 Packaging MIDlet

5.10 Using the Emulator

The emulator looks and acts like a mobile phone [9, 37].

In this section, we will describe how to control the emulator. Although the most use of the emulator is the Default Control Phone. All the emulators operate in a similar way, Where the J2ME Wireless Toolkit represents different kinds of devices as the following:-

- **DefaultColorPhone:** - the most used in run state of the MIDlet with the screen size of 240x320 and color 4096[10].
- **DefaultGrayPhone:** - works with screen size 180x208 and color 4096.
- **QwertyDevice:** - demonstrated with screen size 636x235 and color 4096.

5.11 Running the application on a real mobile phone devices

There are two possibilities: either we will transfer MIDlet to the phone from the computer via an external connection (which can be serial cable of USB, or Bluetooth - depending on the device manufacturer) [9].

Chapter Six

Conclusion and Recommendations

6.1 Conclusions

This chapter summarizes the main work in this thesis, the Design and developing of mobile applications would be a complex and sometimes a difficult task to understand for users in levels of programming in the application. These types of tasks require that an environment that determine the programming language that may take a long time for downloading it, and then to construct and configure which are required to provide a service for the device used.

The conclusion of this thesis work is organized as follows: section 6.1 presents a summary of my thesis work. In section 6.2 introduce recommendations for future research work.

6.2 Summary of the thesis work

- The analysis and design stages we used UML as a tool through the whole work of my framework to be presented in a more elegant way in several stages via a common tool design called RUP.
- In most cases, it is difficult to recognize between a long term program with one class. Based on that, my application indicates four classes (two classes and interfaces) to apply simplicity to my work, the thing that support my work to run successfully. These classes required many relationships and interactions during the analysis and design stages. Each stage needed many phases that uses many diagrams which tend a better organized via using RUP too.
- The cases that will be used can be developed to allow us to extent any other mobile applications.

6.3 Recommendations for future research work

The main focus of my future work will be to improve some ideas to complete services according to work as the other functions on the mobile device.

The practical results of the thesis work corresponding to the demonstration on the mobile are summarized as the following points:

- This work requiring an activation of the E_Mail addresses application to send and receive messaging between a numbers of users by using some support from active server address.
- The activate phone number must be tested by dialing between two or more phone numbers using some active server.
- It also could be published by using a class with a proxy server over the internet.

References

[1] Alan Dennis, Barbara H. Wixom and David Tegarden, " Systems Analysis & Design An Object-Oriented Approach with UML" , University of Virginia, 2001, pp6-54, pp94-164, pp687-690.

[2] Alauddin Alomary, P.Balakrishnan , and Mohsen Elloumi, " Programming with Java ", College of Technology , university of Qatar, 2002.

[3] Azza A. Elsahli, "Development of an SRH Clinics Requirement Mobile System ", Heidelberg University of Applied Sciences, Faculty of informatics, Germany, 2005.

[4] B. Alabio, "Transformation of Analysis Models in Object oriented approches" , pp375-377, Mar ,2000.

[5] Bill Graham, " Developing embedded and mobile Java Technology- based applications using UML", Technical Marketing Engineer at IBM , Carleton University , 24 Nov 2003.

[6] Claudia Niederee, Joachim W.Schmidt, " Software system Institute", OOA&D , TU Hamburg- Harburg , 1998/99, pp2.

[7] Geoffrey Sparks, " An Introduction to modelling Software Systems using UML", Carleton University Ottawa, Ontario- Canada , 2000.

[8] J.W. Schmidt and F.Matthes, " Object Oriented Analysis And Design Activity Diagrams ", TU Hamburg – Harburg , 2000 , pp1-3

[9] James Keogh, " The complete Reference with J2ME ", Java Application Development Istructor, columbia University, 2003.

[10] James Rumbaugh, Ivar Jacobson, and Grady Booch, " The Unified Modeling Language Reference manual ",University of Ottawa ,, First printing , September 1999.

[11] Java 2ME portal, Whait Paper, 16-10-2005.

[12] Jeffrey L.Whitten , Lonnie D. Bentley, and Kevin C.Dittman, " System analysis and Design Methods " , 5Th Edition, pp687-690.

[13] John W.Muchow, " Core J2ME TM Technology & MIDP" ,whait paper, Dec 21,2001, California .

[14] John W.Satzinger and Tore U. Qrvik, " Object-Oriented Approath Concepts, System Development, and Modeling with UML", Agder University College Norway, 2001, pp55, pp375- 377.

[15] Kinshuk, "Adaptive Mobile Learning Technologies, department of information system " , Massey University New Zealand, Paper, pp1-4.

[16] Martin Cooper, " History of Mobile Phones" , From Wikipedia, 1973,pp1-6.

[17] Martyn Mallick, " Mobile and wireless Design Essentials", published simultaneously in Canada, January 2003, pp10-285.

[18] Michael Lervik and Vegard B.Havdal, " Java The UML Way, Object-Oriented Design and Programming" , 2002, England.

[19] Michael and Soma Ghosh, " Java development tools " ,white paper, 01 Nov 2001.

[20] Michael Morrison, " Wireless Java with J2ME" , BELL College, June 2001.

[21] Paulo pinheiro da silva and Norman W.paton , "The unified Modeling Language for Interactive Application",Department of Computer Science, University of Manchester, Oxford Road , England , UK , May 2000, p10-11.

[22] Paulo pinheiro da silva and Norman W.paton, " User Interface Modelling with UML" , white paper, University of Manchester , Oxford Road, UK .

[23] Rolf Hennicker and Nora Koch, " Modeling the user requirements of mobile application with UML" , Institute of Computer Science , Ludwig-Maxiimilians - University of Munich , Oettingenstr.67, Germany, 1998.

[24] Salzburg,Austria, " Development of interactive Application for Mobile Devices" , September,19/2005, pp 1-12.

[25] Scott worley, "Mobile Application development" , Dec 21,2001, pp2-5.

[26] Sinan si Alhir, " Learning UML" , United States of America, paper, July 2003.

[27] Singli and Jonathan Knudsen, " Beginning J2ME From Novice to Professional" , Third Edition, The united States, 2005.

[28] Singli and Jonathan Knudsen, " Developing MIDLet suites for Wireless Applications" , paper, 2004, vol 2 , pp1-12.

[29] Smith and simth, " Database Abstractions: Aggregation and Generalization, relationships" , 1977, pp8.

[30] Soma Ghosh, " Think small with J2ME, senior Application Developer ", Fntigo, 01 Nov 2001.

[31] Stephen Gilmore, valentin heanel and D. Jane Hillston, " A design environment for mobile applications" , the university of Edinburgh , scotland , January 5, 2006.

[32] Stephen R.Schach, " object-Oriented and Classical software Engineering" , fifth Edition, pp366-377.

[33] The Object Management Group(OMG), OMG Model Driven Architecture, July 2001 .Document ormsc/2001-07-01 , http://www.omg.org.

[34] UML: the unified modeling language. http://www.omg.org/uml/ (2001).

[35] Valentino Lee, Heather schneider and Robbie schell, " Mobile Applications Architecture ,Design , and Development" , first Printing , United States, 2004 , pp1-219.

[36] Wisdom Assen Bv., " programming in mobile world ", white paper, December 2001 , pp3-11.

[37] www.core J2ME.com.

ملخص الرسالة

في هذا البحث ، سنناقش المجال الرئيسي لبداية بناء تطبيق على الهاتف النقال. التطبيق يكون لبناء دليل يحتوي علي رقم الهاتف الشخصي ، والاسم ، والبريد الالكتروني، مع بعض الوظائف التي منها إمكانية عرض كل السجلات المخزنة في الدليل . الاستفسار حول المسجل الحالي، وأيضا بعض العمليات مثل إضافة سجل جديد، حذف سجل موجود.

التحليل والتصميم سينفذان باستخدام لغة النمذجة الموحدة (UML) لوضوح تعريفها لمراحل النظام، وخصائص استخدام الأهداف الموجهة مع مختلف أنواع نماذج (UML) مثل:

(UseCase diagram, Class diagram, and Sequence diagram)

مع مخططات أخرى عديدة التي سوف تكون مستخدمة أيضا، في حالة من لا منهجية مستخدمة، والتمثيل للنظام باستخدام تقنيان J2ME كلغة برمجة.

J2ME تكون من أكثر التقنيات ملائمة لتطبيقات النقال اليوم. حيث تحتوي على عدة مكونات جاهزة لتطبيقات النقال التي ستصبح بيئة البرمجة لبحثي. هذه البيئة تتطلب ما يسمى بأداة عدة لاسلكي (WKTool) وبيئة مجموعة تطوير البرامج وتسمى (SDK) التي تكون متوفرة بسهولة للاستخدام مع أي نظام تشغيل اليوم. التطبيق سيكون مختبر على أنظمة تشغيل مختلفة مثل (Windows XP or Windows 98) مع أجهزة نقال مناسبة التي تكون محدودة جدا في الذاكرة، حجم الشاشة الصغير وقابلية إدخال البيانات.

الجماهيرية العربية الليبية الشعبية الاشتراكية العظمى

جامعة التحدي

سرت

لأن الديمقراطية ليست غاية في حد ذاتها
وإنما الغاية هي خلق الإنسان الديمقراطي الحقيقي

التاريخ : ....
الموافق : ....
الرقم الإشاري : ....

كليــة العلــوم

قســم الحاســوب

عنـوان البحـث

تحليــل و تصميــم و تطويــر تطبيــق علــى الهاتــف النقــال

مقــدمة مــن الطالبــة

نعيمــة يحيــى أحمــد

** لجنــة المناقشــة :

1 – د. محمــد أحمــد اخليــف .... [illegible]
(مشــرفاً)

2 – د. عمــر مصطــفى الصلابي .... [illegible]
(ممتحنــاً داخليــاً)

3 – د. مجــدي علي محمــد التــومي .... [illegible]
(ممتحنــاً خارجيــاً)

يعتمــد

د.أحمد فرج المحجوب

[illegible]

أمين اللجنة الشعبية لكلية العلوم

جامعة التحدي كلية العلوم

WWW.altahadi.edu.ly

info@altahadi.edu.ly 674 054/60361-62151 054/60363-65704

جامعـــة التحـــدي

كـليـة العـلوم

تحليل وتصميم وتطوير تطبيق على الهاتف النقال باستخدام تقنيات الجافا

هذه الرسالة مقدمة لقسم الحاسوب كمتطلب جزئي للحصول على درجة الماجستير في علوم الحاسوب

مقدمة من الطالبة :نعيمة يحيى أحمد

إشـراف الدكتور: محمد أحمد اخليف

العام الجامعي 2007-2008