Palestine Polytechnic University

College of Engineering & Technology
Electrical & Computer Engineering Department

Graduation Project

Simulation of Automated Meter Reading System

Project Team

Ala'a Al-Dawadeh

Sami Ghannam

Thaher Abu-Hamdan

Project Supervisor
Eng. Radwan Tahboub

Hebron – Palestine

June , 2006

Abstract

اسم المشروع
النظام المحاكي لقراءة عدادات الطاقة

عمل الطلاب

سامي محمد غنام و علاء إسماعيل الدودة و ظاهر محمد أبو حمدان

تخصص هندسة أنظمة الحاسوب

جامعة بوليتكنك فلسطين – 2006

بإشراف

م.رضوان طهبوب .

من الملاحظ أن أعداد البشر في العالم في ازدياد مطرد و مستمر مع الزمن، و أن مدى احتياجهم للمصادر المختلفة للطاقة في تزايد كبير، لذلك جاءت شركات الطاقة بأنواعها المختلفة لتنظيم عملية توزيعها على سكان العالم مقابل مبلغ معين لقاء خدمة تزويدهم بالطاقة، إلا إن العقبات كانت و ما تزال تواجههم في هذا المجال.

مع التقدم الحاصل في مجال الشبكات والتكنولوجيا و الطرق البرمجية، بدأت تظهر حلول و بدائل للمشكلات التي كانت تواجه شركات الطاقة في عملية جمع البيانات الخاصة بكمية استهلاك الطاقة من قبل المشتركين و طرق إدخالها إلى الحاسب و حساب قيمتها و إصدار الفواتير المتعلقة بها بسرعة و سهولة كبيرين و الاستغناء عن الطرق التقليدية القديمة التي كانت متعبة بالنسبة للمشتركين و شركات الطاقة، حيث أن الأخطاء كانت دائمة الحدوث بشكل كبير.

و من هذا المنطلق، قمنا بعمل هذا المشروع ليكون احدي البدائل لحل مشكلة شركات الطاقة في هذا المجال و ليكون خدمة للمشتركين في هذا النظام، من حيث جمع قراءات الطاقة و إدخالها للحاسب و تنظيمها و تخزينها بصورة آمنة بعيدة عن مجال الخطأ الحاصل بالطرق التقليدية، الأمر الذي يجعل من هذا المشروع طريقة ديناميكية في غاية الأهمية.

Abstract

Project Name
Simulation of Automated Meter Reading System

Designed by

Ala' Isma'el Al-Dawadeh **Sami Mohmmad Ghannam**
Thaher Mohammad Abu Hamdan

Palestine Polytechnic University – 2006

Supervisor

Eng. Radwan Tahboub

The increasing human beings in the world affect on consuming and the way of distribution the power among them, and the technology used for years ago become unpractical way according to the power company in collecting data manually by groups of manpowers and entering these data by another huge groups, this is one of critical problem from many of them faces power companies today, and the good question that each company asked is about the solutions.

As new technologies and new programming methods appear, the solutions become available for collecting data by using a network technology without depend basically on increasing human beings and used mobile agent technology as intermediate software between those human and power companies.

So, in our project we will explain these new technologies present a good solution for that power companies and how the world will be change to accommodate new recent century.

Table of Contents

فهرس الجداول
List of Tables

فهرس الأشكال التوضيحية
List of figures

Chapter One

Introduction

Chapter One

Introduction

1.1 Overview

Over the past few years, it becomes clear that world has changed forever. We are now in the information age – the second industrial revolution. This revolution talk about the way people worked through networking and data communications.

The value of high-speed data communication network is that it brings people together in a way never before possible. Today, people can communicate and access information anywhere in the world regardless of their location. In fact, the problem today is that we cannot handle the quantities of information we receive.

In our world the computer stands at the top of human civilization in this era to make their life more comfortable in many fields, here we will talk about the Automated Meter Reading (AMR) Field, and how this system using networking technology that will present clear effect to serve human beings in their real life.

Automatic Meter Reading helps the consumer, the power provider and energy services provider access to the latest and accurate information from the metering software simulation setup in each hosting computer and also provides savings in time, manpower, and helps in efficient energy management. Energy Tracking and Automatic Meter Reading (AMR) provides immediate benefits for power energy.

The data is transmitted to server that collects the information using client-server technologies. The customers access this information by using a dedicated network that is connected to the Internet. All meters transmit the stored data over the network to the data collection server that store, calculate and aggregate the energy information at the web page. Briefcase is created which bring all the metered data to a centralized view. The data is automatically collected and aggregated; one does not need spreadsheets or spend time entering energy usage for multiple areas for energy management. As the role of the networking is getting an important, new tools are needed to cope with its distributed nature. Software Mobile Agents present methods for maintaining and using distributed systems.

In many applications software, agents changed the way client-server systems are designed. They provide simple and effective methods for dealing with and distributed networks in Java Runtime Environment (JRE). A mobile agent is a program (code and data) that can migrate from machine to machine in a network of heterogeneous computer systems. It works independently and communicates with other agents and host systems.

In this project, we will introduce the term of the Automatic Meter Reading system using Mobile Agent method and the client server method for collecting data.

Then, we will present a Simulation and implementation testing to the AMR system and shows the applicability of this approach in improving the bandwidth usage for number of networked Power meters, hence a better performance compared to traditional client-server approach and mobile agent approach.

1.2 General idea

The overall idea of this project is build a simulation for the Automated meter reading system by using different approach and then comparing the performance and the bandwidth by using different approaches (client-server approach and mobile agent approach) to determine the better performance, speed and other factors on the network with each apprach.

The Automated meter reading system (AMR) will read the power energy consumed by each hosting machine (consumer), automatically, using the two approaches, independently, along the network .

Each approach has a special way to collect the readings (data) from each power meter reader which is connected to the hosting machine that impact directly on the system performance.

1.3 The Project Importance

Meter Reading, storing reading and produces Billing considers the most time consuming functions performed in utilities. These functions have a major influence on the utilities cost, efficiency, productivity and structure as well.

In the past, solutions based on collecting and recording readings manually, then entering it into a central billing system are time consuming, prone to errors and delays in delivering bills to customers.

There are many methods involved in the meter reading process; this includes traditional manual methods up to fully automatic meter reading systems.

1.3.1 Traditional Systems

In the past, there are several techniques to collect the meter readings, At first, the power companies depend on manpower to travel from one place to another for collecting data from each meter.

For a while, some power companies in other countries depend on a customer rather than the manpower for reading the meter that have it, and tell the company about its reading by the phone or before pay the previous bill.

These ways to collect data readings are very time consuming and does not satisfy the business requirements for the power company, and less accuracy in the reading operation.

1.3.2 Hand-Held units

Hand-Held units for meter reading and billing system: speed, accuracy, and cost effectiveness are the strongest features in this System. In addition of providing a solution that is able to deal with a current billing system which exists or to be integrate with the lunched system; therefore the presented solution is able to provide a middle program that works as translator between any billing system and the Hand-Held Units. The delay between reading the meter and delivering the bill to customers is reduced to zero, but it still not powerful way to the consumers in all time; because it still need man reader.

1.3.3 Automated Meter Reading System

As mentioned above, the previouse ways to perfom the major utilities fuctions is not a proper solution for both power energy side and consuumer side, so, the search on another alternative way becomes necessary for both sides as a suitable solution to pass the drawbacks that appear in old ways.

Nowadays, the direction towards to the automated meter reading system as a perfect solution to release from all previous problems appear in old system are increasing rabidly to perfom the utilities of system functions in best way.

The Automatic Meter Reading (AMR) technology is a network-based application used by power company to increase performance and reliability of their power transmission and distribution systems.

The automated meter reading system (AMR) used for many purposes, including theft detection, outage management, customer energy management, load management, on/off services, and distributed automation .

In real world, AMR system has three major components: the meter interface unit (MIU), central office system and communication system (responsible for data transmission between central office and MIU). The meter interface unit (MIU) collects data from the meter and manages communications.

In this project ,we will study the the AMR system using suggested approach , then measures the performance factors for each approach.

"A PERFORMANCE ANALYSIS FRAMEWORK FOR MOBILE-AGENT SYSTEMS"

This paper prepared by Cyprus university, It focuses on some objectives like:

- Study & argue about performance issues of mobile-agent based systems.
- Compare mobile-agent platforms quantitatively.
- Discover potential performance bottlenecks.
- Monitor MA-based systems' performance.

"USING MOBILE AGENTS TO IMPROVE PERFORMANCE OF NETWORK MANAGEMENT OPERATIONS"

This paper prepared by Illinoise University, and it organized to show some subject related with the mobile agent implementation and performance issues in network management describe a proposed architecture for mobile agent in network management, explain the performance model for network management using Health Function, and presents a proposed method in mobile agent implementation using simulation.

2. Related with Automated meter reading system

"Power meter reading through internet connection"

This paper prepared by kambridge University in U.K., it gives good information about how the system can read the power meter value through the internet, and it explain the idea of AMR system.

"Power meter reading system based mobile agent"

this paper talk describe AMR system in detail that used a mobile agent technology start with creating agent until it returns to the source and explain how this technology solving many problems related with the network bandwidth and collecting reading.

1.5 Estimated Cost

In this section we can describes the hardware and the support software required to carry out the development.

So, we can estimate the price of these machines and cases if this hardware and software are bought and we put these prices in schedule which can include all what we needed.

1.5.1 Hardware Resources cost

In our project, we will assume that the existing network is the network we will use without any additional components, and the machines that connected to the network are responsible to generate "consumed power reading random numbers" as a simulation instead of using the physical meter hardware ,so there is no hardware cost we can mention here.

1.5.2 Software Resources cost

In this section, we will mention the requirement of software applications used in our project, and there is :

Table 1.1: project software cost.

Requirements	Cost
Java Application	$150
Jade Package	Free
Java Runtime Environment	$50
Total	$200

1.5.3 Human Resources cost

We will mention here the requirement of the human payments that create and build this project.

We have three engineering with salary 800$ per month for each.
So, the total cost = 3 * $800 * 8 month = $19200.

The Total Cost is:

Software cost + Human Cost = $200 + $19200

= **$19400.**

1.6 Project Schedule:

In this section we want to specify the tasks with its flags to give knowledge to which this system likes.

1.6.1 Phase One: First Semester

T1 → gathering theoretical information related with JAVA environment using JADE package, AMR systems and mobile agent, then organize it.

T2 → study the implementation JAVA language that are use and JADE package theoretically and programming.

T3 → build simple network that contain two computers as client-server and setup JADE packages in JAVA languages.

T4→ represent a different system models and determine the system components.

T5→built a simple program using JADE programming.

T6→ writing a chapter one from project.

T7→ writing a chapter two from project.

T8→ writing a chapter three from project.

T9→ printing the final copy of project.

Table 1.2: First semester project schedule

Task	Duration in Weeks															
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
T1																
T2																
T3																
T4																
T5																
T6																
T7																
T8																
T9																

1.6.2 Phase Two: Second Semester

T1→ Start the Programming of the system components

T2→ Debugging each component alone to verify that it works well.

T3→ Gathering components in one program and apply it in real network.

T4→ Modify the result errors that may occurs.

T5→ Write full documentation and print it.

Table 1.3: Second semester project schedule

Task	Duration in Weeks															
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
T1																
T2																
T3																
T4																
T5																

1.7 Risk Management

In this section we want to describe the possible project risks and how we solve these problems.

From search operations and predict to some risks which may be affect on the way and creation system, we suggest some risks which require to create security planning to avoid these risks or to reduce its effect on each stage, and it's effect on performance and the quality of the system .

There are two types of risks:

1. Team risks when we begin.
2. Project risks.

1.7.1 Risks that face the project team:

1. May be the university studies condition cause to make the team busy for this project study, and this affect on our time scheduler to submit our project, and to complete writing.

2. Loss of resources related to our project, this made us to take more time in searching about resources near to our project in the idea.

3. Study new programming approach that we are not dealing with it previously so, the problems face us take more time to solve it, especially the problems related with the network.

4. Geographical distance among project team cause difficult to manage our meeting.

1.7.2 Risks which face the project

1. Any development in network that belongs to the system may cause badly in system performance.

2. This project needs special environment that must be setup on each client computer that is connected to the system.

Some suggestions solving to avoid or reduce the effective of these risks which affect on this system are:

1. Determine suitable time for team members meeting with supervisor that not conflict with lecture in university or the team member themselves.

2. According to the special environment it must be consider as system requirements for each client connected to this system.
3. We try to accommodate the development of the networking by create program support to most types of network.

1.8 Report Content

This project includes Six Chapters divided into Two Semesters explain it as follow:

Chapter One: The Introduction

This chapter includes the general idea about the project, its importance, literature review on this field, the estimated cost, time schedule and the risk management.

Chapter two: Theoretical Background

This chapter include the theoretical subjects related to the main idea of the project, the information a bout the special components and the system requirement especially from software viewpoint.

Chapter three: Architectural design

This chapter includes the project objectives, the general block diagram, how system works and the system modeling.

Chapter four: Detailed System Design

This chapter includes the discussing design options and justifying those chosen for the project, detailed description of project components, user interface design and the general algorithms and its flowcharts.

Chapter Five: Implementation and testing

This chapter includes the real implementation and testing for the subsystem (components) of the project.

Chapter Six: Conclusions and future work

This chapter include the list of the problem faced the project team in accomplishing the system and how did they resolved. The conclusions that help the readers are included here with the future works.

Chapter Two

Theoritical Background

Chapter Two

Theoretical Background

2.1 Overview

The concept of networking is that a simple network consists of two computers connected to each other by a cable ,so they can share data, all networking, no matter how sophisticated, stem from that simple system.

The three basic hardware components for a data communications network are server or host computer (e.g., microcomputer, mainframe), client (e.g., microcomputer, terminal), and circuit (e.g., cable, modem) over which message flow.

Both server and client also need special-purpose network software that allows them to communicate.

The server stores data or software that can be accessed by the clients. In client-server computing ,several servers may work together over the network with a client computer to support the business application.

The client is the input/output device at the user's end of a communication circuit, It typically provides users with access to the network and the data and software on the server.

The circuit is the pathway through which the messages travel. It typically a copper wire, although fiber optic cable and wireless transmission are becoming more common. There are many devices in the circuit that perform special function such as hubs, switches, routers, and gateways.

2.2 Theoretical subject related to the main ideas of the project.

There are many different ways to categorize networks along with its geographic scope; these are local area networks, backbone networks, metropolitan networks, and wide area networks.

2.2.1 LANs (local area networks)

Local area networks (LANs) consist of computers, network interface cards, networking media, network traffic control devices, and peripheral devices. LANs make it possible for businesses that use computer technology to make possible communications such as e-mail. They tie together: data, communications, computing, and file servers. LANs support moderately high-speed data transmission, commonly operating at 10 to 100 million bits per second (10-100 Mbps).

LANs are designed to do the following:

- operate within a limited geographic area
- allow many users to access high-bandwidth media
- provide full-time connectivity to local services
- connect physically adjacent devices

2.2.2 WANs (wide area networks)

WANs interconnected LANs, which then provided access to computers or file servers in other locations. As a result of being networked or connected, computers, printers, and other devices on a WAN could communicate with each other to share information and resources, as well as to access the Internet.

Most organizations do not build their own WANs by laying cable, building microwave towers, or sending up satellites (unless they have unusually heavy data transmission needs or highly specialized requirements, such as Department of Defense). Instead, most organizations lease circuits from international communication companies.

2.2.3 Components and Devices

2.2.3.1 Ethernet 10BASE-T

The Ethernet 10BASE-T technologies carry Ethernet frames on inexpensive twisted-pair wiring. The first four components are passive, meaning they require no energy to operate. They are:

- patch panels
- plugs
- cabling
- jacks

The last three are active. They require energy to do their jobs. They are:

- transceivers
- repeaters
- hubs

2.2.3.2 Connectors

The standard 10BASE-T termination (end point, 0 plug, and connector) is the registered jack-45 connector (RJ-45). It reduces noise, reflection, and mechanical stability problems, and resembles a phone plug, except that it has eight conductors instead of four. It is considered a passive networking component because it only serves as a conducting path between the four pairs of Category 5 twisted cable and the prongs of the RJ-45 jack. It is considered a Layer 1 component, rather than a device, because it serves only as a conducting path for bits.

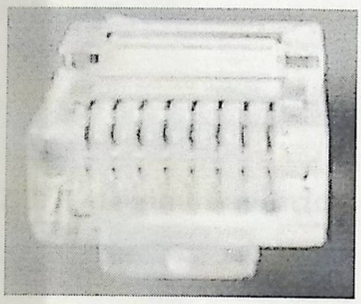


Figure 2.1: RJ-45 Connector[1]

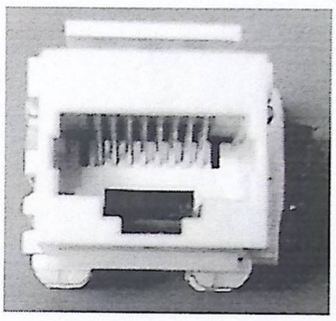


Figure 2.2: RJ-45 FrontView[2]

RJ-45 plugs fit into RJ-45 jacks or receptacles. The RJ-45 jack has eight conductors, which snap together with the RJ-45 plug. On the other side of the RJ-45 jack is a punch down block where wires are separated out and forced into slots with a fork-like tool called a punch-down tool. This provides a copper-conducting path for the bits. The RJ-45 jack is a Layer 1 component.

2.2.3.3 Routers

The router is the first device that will work with that is at the OSI network layer, or otherwise known as layer 3. Working at layer 3 allows the router to make

[1] Network Cisco Academy.

[2] Network Cisco Academy.

decisions based on groups of network addresses (Classes) as opposed to the individual MAC addresses like is done on layer 2. Routers can also connect different layer 2 technologies, such as Ethernet, Token-ring, and FDDI. However, because of their ability to route packets based on Layer 3 information, routers have become the backbone of the Internet, running the IP protocol.

The purpose of a router is to examine incoming packets (layer 3 data), choose the best path for them through the network, and then switch them to the proper outgoing port. Routers are the most important traffic-regulating devices on large networks.

2.2.4 Digital Bandwidth

2.2.4.1 Digital bandwidth measurements

Bandwidth is the measure of how much information can flow from one place to another in a given amount of time. There are two common uses of the word bandwidth: one deals with analog signals, and the other with digital signals.

The most basic unit used to describe the flow of digital information, from one place to another, is the bit. The next term is the one used to describe the basic unit of time. It is the second.

A bit per second is a unit of bandwidth. if communication happened at this rate, 1 bit per 1 second, it would be very slow.

Table 2.1: The units of bandwidth.[3]

Unit of Bandwidth	Abbrev.	Equivalence
Bits per second	bps	1 bps = fundamental unit of bandwidth
Kilobits per second	kbps	1 kbps = 1,000 bps = 10^3 bps
Megabits per second	Mbps	1 Mbps = 1,000,000 bps = 10^6 bps
Gigabits per second	Gbps	1 Gbps = 1,000,000,000 bps = 10^9 bps

2.2.4.2 Media bandwidth differences

Bandwidth is a very useful concept. It does, however, have limitations. No matter how to send a messages, no matter which physical medium used, bandwidth is limited. This is due both to the laws of physics and to the current technological advances.

Figure below illustrates the maximum digital bandwidth that is possible, including length limitations, for some common networking media.

Table 2.2: Typical media according its Bandwidth[4]

Some Typical Media	Bandwidth	Max. Physical Distance
50-Ohm Coaxial Cable (Ethernet 10Base2, ThinNet)	10-100 Mbps	185m
75-Ohm Coaxial Cable (Ethernet 10Base5, ThickNet)	10-100 Mbps	500m
Category 5 Unshielded Twisted Pair (UTP) (Ethernet 10BaseT)	10 Mbps	100m
Other technologies being researched	2400 Mbps (2.400 Gbps)	40km = 40,000m
Wireless	10 Mbps	100m

[3] Network Cisco Academy.
[4] Network Cisco Academy.

2.2.4.3 Data throughput in relation to digital bandwidth

Throughput refers to actual, measured, bandwidth, at a specific time of day, using specific internet routes, while downloading a specific file.

The throughput is often far less then the maximum possible digital bandwidth of the medium that is being used. Some of the factors that determine throughput and bandwidth include the following :

- internetworking devices
- type of data being transferred
- topology
- number of users
- user's computer
- server computer
- power and weather-induced outages

When design a network, it is important to consider the theoretical bandwidth. The network will be no faster than media will allow. When clients actually work on networks, they will want to measure throughput and decide if the throughput is adequate for them.

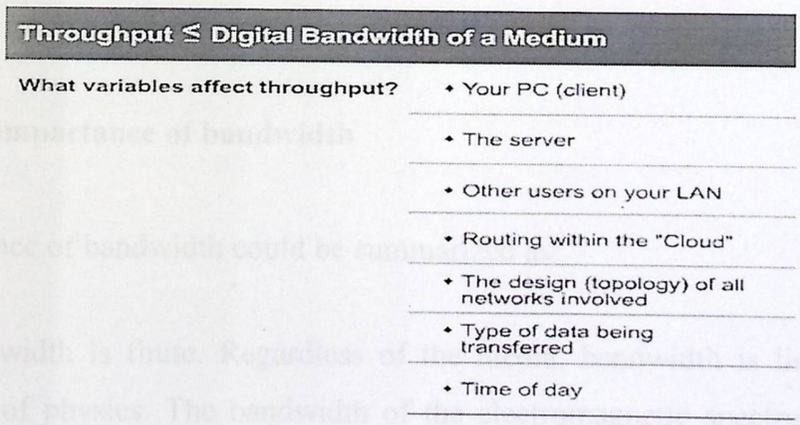


Figure 2.3: Throughput Variables.[5]

[5] Network Cisco Academy.

2.2.4.4 Data transfer calculation

An important part of networking involves making decisions about which medium to use. Figure 2.4 below summarize a simple formula that ***Estimated Time = Size of File / Bandwidth***. The resulting answer represents the fastest that data could be transferred. It does not take into account any of the previously discussed issues that affect throughput, but does give a rough estimate of the time it will take to send information using that specific medium/application.

Best Download $T = \frac{S}{BW}$	Typical Download $T = \frac{S}{P}$
BW =	Maximum theoretical bandwidth of the "slowest link" between the source host and the destination host.
P =	Actual throughput at the moment of transfer.
T =	Time for file transfer to occur.
S =	File size in bits.

Figure 2.4: File Transfer Time Calculation[6]

2.2.4.5 The importance of bandwidth

The importance of bandwidth could be summarized as:

1. Bandwidth is finite. Regardless of the media, bandwidth is limited by the laws of physics. The bandwidth of the electromagnetic spectrum is finite - there are only so many frequencies in the radio wave, microwave, and

[6] Network Cisco Academy.

infrared spectrum. Because this is so, the FCC has a whole division to control bandwidth and who uses it. Optical fiber has virtually limitless bandwidth. However, the rest of the technology to make extremely high bandwidth networks that fully use the potential of optical fiber are just now being developed and implemented.

2. Knowing how bandwidth works, and that it is finite, can save lots of money.

3. It is a key to measure and understand the network design and performance.. They are major factors in analyzing network performance. In addition, as a network designer of brand new networks, bandwidth will always be one of the major design issues.

4. It is a key to understanding the Internet. So, there are two major concepts related to the "information superhighway". The first is that any form of information can be stored as a long string of bits. The second is that storing information as bits, while useful, is not the truly revolutionary technology.

5. It is not uncommon that once a person or an institution starts using a network, they eventually want more and more bandwidth. New multimedia software programs require much more bandwidth than those used in the mid-1990s.

The programmers are busily designing new applications that are capable of performing more complex communication tasks, thus requiring greater bandwidth.

2.2.5 The OSI Reference Model

Computer networks are designed in a highly structured way to reduce their design complexity. Most networks are organized as a series of layers or levels. The number of layers, the name of each layer, and the function of each layer differs from network to network. However, in all networks, each layer clearly defines various data communication functions and logical operations. Each level is functionally independent of the others, but builds on its predecessor. In order to function, higher levels depend on correct operation of the lower levels.

2.2.5.1 Encapsulation

All communications on a network originate at a source, and are sent to a destination, and that the information that is sent on a network is referred to as data or data packets. If one computer (host A) wants to send data to another computer (host B), the data must first be packaged by a process called encapsulation.

Encapsulation wraps data with the necessary protocol information before network transit. Therefore, as the data packet moves down through the layers of the OSI model, it receives headers, trailers, and other information.

2.2.5.2 TCP/IP protocol Model

The TCP/IP model emphasizes maximum flexibility, at the application layer, for developers of software. The transport layer involves two protocols - transmission control protocol (TCP) and user datagram protocol (UDP).. The lowest layer, the network layer, refers to the particular LAN or WAN technology that is being used.

In the TCP/IP model, regardless of which application requests network services, and regardless of which transport protocol is used, there is only one network protocol (internet protocol, or IP). This is a deliberate design decision. IP serves as a universal protocol that allows any computer, anywhere, to communicate at any time.

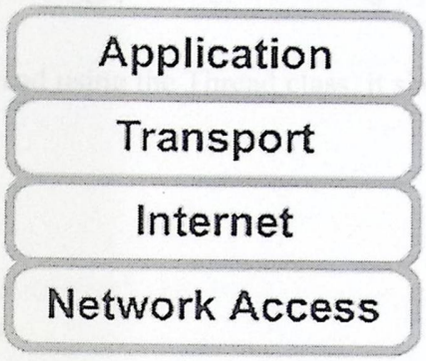


Figure 2.5: TCP/IP Model[7]

2.2.6 Threads

The thread is defined as a sequence of a program which runs a certain function within a program. Some operating systems can run multiple threads at one time, allowing for fast execution of an application.

A thread is similar to the sequential programs ,it has a beginning, a sequence, and an end. At any given time during the runtime of the thread, there is a single point of execution. However, a thread itself is not a program; it cannot run on its own, rather a thread runs within a program.

threads -in real- is not a single sequential exactly ; it's about the use of multiple threads running at the same time and performing different tasks in a single program.

[7] Network Cisco Academy.

As a sequential flow of control, a thread must own some resources within a running program like execution stack and program counter.

Basic support for threads in the Java platform is in the class java.lang.Thread. It provides a thread API and all the generic behaviors for threads. The behaviors include starting, sleeping, running, yielding, and having a priority.

To implement a thread using the Thread class, it should be in run method that performs the thread's task.

2.2.6.1 Thread Attributes

Java threads are implemented by the Thread class, which is part of the java.lang package. The Thread class implements a system independent definition of Java threads. the actual implementation of concurrent operation is provided by a system-specific implementation.

The features that determine the threads in general are:

1. Thread Body : All of the action takes place in the thread's body--the thread's run() method. You can provide the body to a Thread in one of two ways: by subclassing the Thread class and overriding its run() method, or by creating a Thread with a Runnable object as its target.

2. Thread State : Throughout its life, a Java thread is in one of several states. A thread's state indicates what the Thread is doing and what it is capable of doing at that time of its life: is it running? is it sleeping? is it dead?

3. Thread Priority : A thread's priority tells the Java thread scheduler when this

thread should run in relation to other threads.

4. Daemon Threads : Daemon threads are those that provide a service for other threads in the system. Any Java thread can be a daemon thread.

5. Thread Group : All threads belong to a thread group. ThreadGroup, a java.lang class, defines and implements the capabilities of a group of related threads.

2.2.7 Information about Special Software Components

2.2.7.1 Java Language

Java was the programming language of choice because of its many attractive features, particularly geared towards object-oriented programming in distributed heterogeneous environments; some of these features are Object Serialization, Reflection API and Remote Method Invocation (RMI).

2.2.7.2 JADE Environment

We choose the Jade Environment to use it in our programming over than other programming language because:

- Distributed agent platform. The agent platform can be split among several hosts only one Java application, and therefore only one Java Virtual Machine, is executed on each host. Agents are implemented as Java threads and live within Agent Containers that provide the runtime support to the agent execution.

- Graphical user interface to manage several agents and agent containers from a remote host.
- Debugging tools to help in developing multi agents applications based on JADE.
- Intra-platform agent mobility, including transfer of both the state and the code (when necessary) of the agent.
- Support to the execution of multiple, parallel and concurrent agent activities via the behavior model. JADE schedules the agent behaviors in a non-preemptive fashion.
- FIPA-compliant Agent Platform, which includes the AMS (Agent Management System), the DF (Directory Facilitator), and the ACC (Agent Communication Channel). All these three components are automatically activated at the agent platform start-up.
- Many FIPA-compliant DFs can be started at run time in order to implement multi-domain applications, where a domain is a logical set of agents, whose services are advertised through a common facilitator. Each DF inherits a GUI and all the standard capabilities defined by FIPA (i.e. capability of registering, deregistering, modifying and searching for agent descriptions; and capability of federating within a network of DF's).
- Efficient transport of ACL messages inside the same agent platform. In fact, messages are transferred encoded as Java objects, rather than strings, in order to avoid marshalling and unmarshalling procedures. When crossing platform boundaries, the message is automatically converted to/from the FIPA compliant syntax, encoding, and transport protocol. This conversion is transparent to the agent implementers that only need to deal with Java objects.
- Library of FIPA interaction protocols ready to be used.
- Automatic registration and deregistration of agents with the AMS

JADE comes bundled with some tools that simplify platform administration and application development. The following tools are available:

- *Remote Management Agent* (*RMA)* acting as a graphical console for platform management and control. A first instance of an RMA can be started with a command line option ("*-gui*"), but then more than one GUI can be activated.
- The *Dummy Agent* is a monitoring and debugging tool, made of a graphical user interface and an underlying JADE agent. Using the GUI it is possible to compose ACL messages and send them to other agents; it is also possible to display the list of all the ACL messages sent and\or received.
- The *Sniffer* is an agent that can intercept ACL messages while they are in flight, and displays them graphically using a notation similar to UML sequence diagrams.
- The *Introspector* is an agent that allows to monitor the life cycle of an agent, its exchanged ACL messages and the behaviours in execution.
- The *DF GUI* is a complete graphical user interface that is used by the default Directory Facilitator (DF) of JADE and that can also be used by every other DF that the user might need.
- The *LogManagerAgent* is an agent that allows setting at runtime logging information, such as the log level, for both JADE and application specific classes that use Java Logging.
- The *SocketProxyAgent* is a simple agent, acting as a bidirectional gateway between a JADE platform and an ordinary TCP/IP connection. ACL messages, travelling over JADE proprietary transport service, are converted to simple ASCII strings and sent over a socket connection. Viceversa, ACL messages can be tunnelled via this TCP/IP connection into the JADE platform.

2.3 System Requirement

2.3.1 Java Virtual Machine (JVM) And Java Development Kit(JDK)

Software requirement to execute the system is the Java Run Time Environment (JRE) ,In order to build the system, the JDK1.4 is sufficient because pre-built the Interface Definition Language (IDL) stubs and Java parser classes are included with the JADE source distribution. Those users, who wish to regenerate IDL stubs and Java parser classes, should also have the JavaCC parser generator, and the IDL to Java compiler. Notice that the old "IDL to java" compiler available with JDK1.2 generates wrong code and it should never be used, instead the new "IDLJ" compiler, that is distributed with JDK1.3, should be used.

JVM is executed on each host. Each JVM setup on hosts is a basic container of agents that provides a complete run time environment for agent execution and allows several agents to concurrently execute on the same host. The main-container, or front-end, is the agent container where the AMS and DF lives and where the RMI registry, that is used internally by JADE, is created. The other agent containers, instead, connect to the main container and provide a complete run-time environment for the execution of any set of JADE agents.

The Java Runtime Environment software size nearby 131MB.

2.3.2 The Network

The Automated Meter Reading System(AMR) dealing with a distributed systems ,and the machines in this system should be connected to each other by a network .

The network error prone ,specifications and distribution is not our field project , we suppose that network exists and works properly and reach to the each machine connected to the system.

We will use the existence network in IT-center building in the Palestine Polytechnic University to carry out our system and ensure it will work properly.

2.3.3 JADE Environment

JADE (Java Agent Development Framework) is a software development framework aimed at developing multi-agent systems and applications conforming to FIPA standards for the agents. It includes two main products: a FIPA-compliant agent platform and a package to develop Java agents.

In our project, the jade packages will be installed fully in the power company computer , and some class jade package needed in consumers' computers to create a suitable environment for agent to work probably, the size of the jade packages software that needed to activate the environment nearby 12.6MB .

2.3.4 Agents Hosting software environment

In our system ,we build a special software package installed at host side instead of using the whole jade environment which contain a requirement classes to do its work properly.

The agent hosting packages mainly responsible to create a container in the main-container that exist in a power company side to be able dealing with agents moving.

This software "Agent hosting software" package size nearby 799KB , so, it reduces the requirement about 12MB .

Chapter Three

System Requirements

Chapter Three

System Requirements

Chapter Three

Architectural Design

3.1 Project Objectives

The goals from this project could be summarized in the points bellow :

1. To check the aplicability of the Automated meter reading system by using different approaches.
2. Find easy way to collect meter reading rather than using manpower or hand held unit to collect and store them for calculation.
3. Save the collecting time of data and compare between the different methods used.
4. solve the problem that result in collecting and storing the meter reading by using different techniques.
5. Analyzing and comparing the performance, bandwidth and effectiveness of Automated meter reading system by using mobile agent and the client/server technologies to find which is better.

3.2 General Block Diagram

In this section we will explain the general diagram for each phases in our automated meter reading system(AMR) and its idea to perform its function as a whole.

3.2.1 Phase One: AMR system using Mobile Agent

In our project, the mobile agent in the AMR system will visit each consumer's power meter to collect data of the power meter along a network and then return to Power Company again as below.

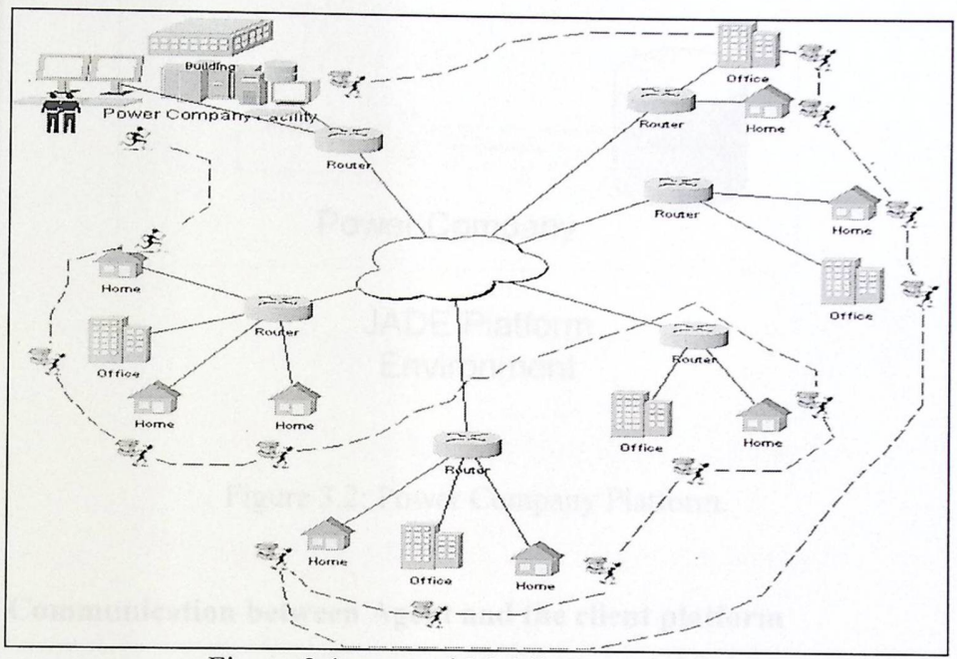


Figure 3.1: general Mobile Agent's Idea.

3.2.1.1 Power Company Software Component

At the power company side, the agent will be created at specific time by the process that build in its main-container, then allocate a required table fields that needed by the agent , specially the static IP addresses of each client connected to the system, note that all clients exist at same platform, after that occur, the process will send the Agent to the network as shown in figure (3.2).

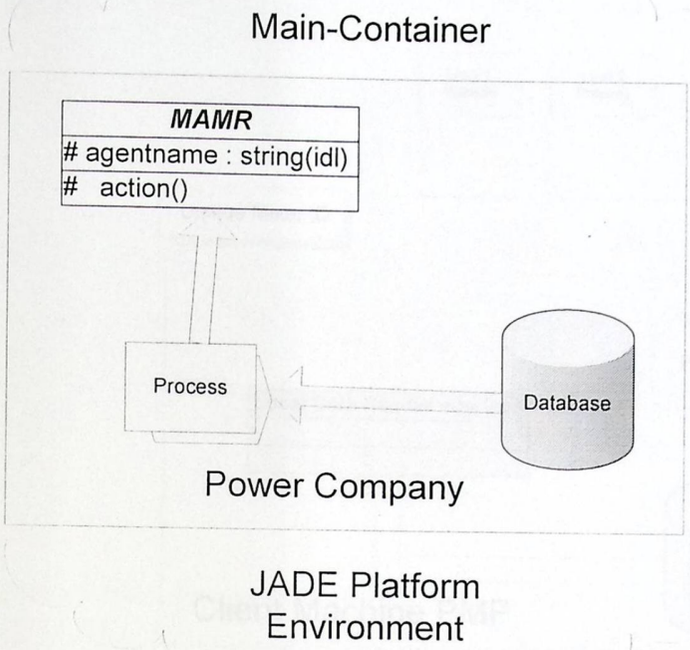


Figure 3.2: Power Company Platform.

3.2.1.2 Communication between Agent and the client platform

When the agent reaches its destination (host platform) according the IP address table carry it, the class in host platform have some operations need to do when agent enter the platform, there are:

- Verify the agent platform if it is come from the power company.
- Verify the IP address and meter ID.
- Execute the agent required task.

After agent tasks finished, it be ready to leave the platform to another one as figure (3.3).

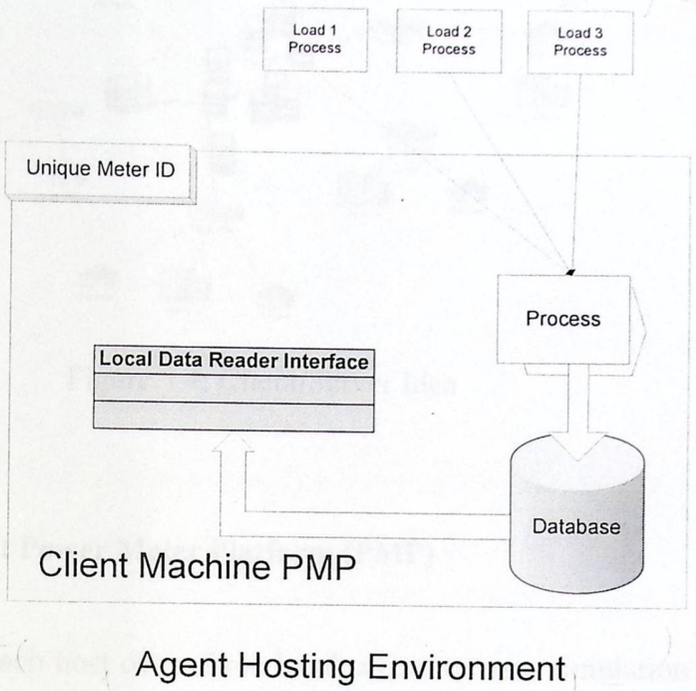


Figure 3.3: communication client with platform

3.2.2 Phase Two: AMR system using client/server

In this section, each clients(host) in the AMR system will send an http message contains the readings of power meter to the server to gathering all clients' readings through the network in its database for future use, the idea of this phase shown in figure (3.4).

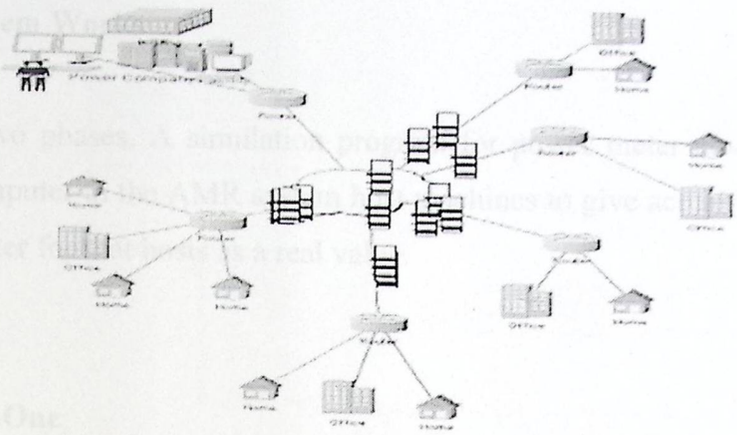


Figure 3.4: Client/Server Idea

3.2.3 Simulation of Host Power Meter Platform (PMP)

At the AMR system,each host owns three local processes as a simulation to the real loads connected with main process to perform a real power meter peripheral as a whole.

At this power meter inside ,there is a database that build to store the simulation loads value every period time to be collected in future.

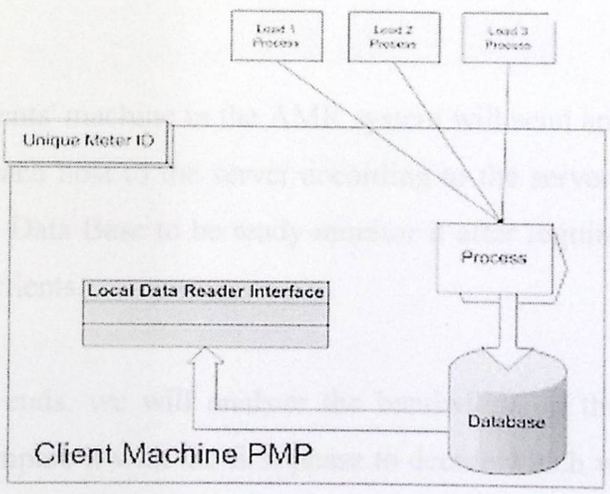


Figure 3.5: Simulation of Host (client) Power Meter Platform (PMP).

3.3 How System Works

For the two phases, A simulation program for power meter device installed to each host computer in the AMR system host machines to give actual consumption of the power meter for that hosts as a real value.

3.3.1 Phase One

At the AMR system using mobile agent technique ,agent collector will be create from the main-container that contain the main container at specific time from each month, and then "move" in the network to reach each client computer, one by one until collect all readings from the clients, then, the agent collector comeback to the server with all readings and fill it automatically in the Database.

After this operation completed, we will calculate the bandwidth on the network that the operation needed.

3.3.2 Phase Two

In this phase , the clients' machine in the AMR system will send an http message carry the readings from each host to the server according to the server-ID , then fill the readings in the server Data Base to be ready monitor it after required calculation on some interface for all clients.

After this operation ends, we will analyze the bandwidth on the network by using this method and compare it with the first phase to decide which way is the best and the difference between them.

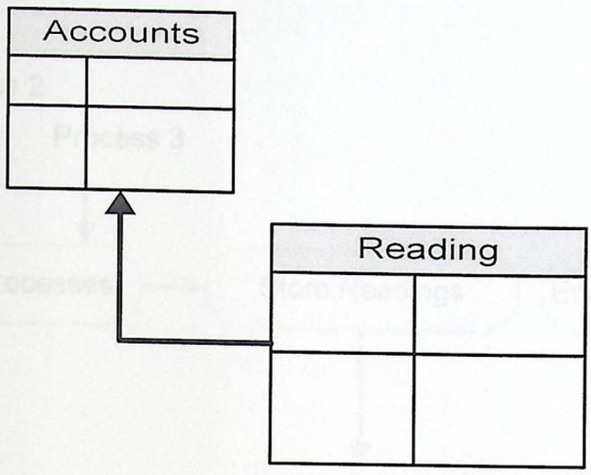


Figure 3.7: power company database.

The figure above explains the tables and its relationship at power company sides, these tables contain a data that used by a company to make its calculation to issues a bill for a clients in future.

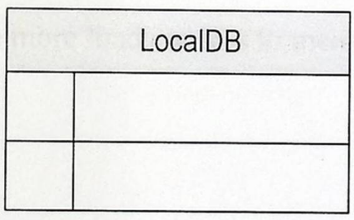


Figure 3.8: local Database for power meter Agent

The figure above shown the Database that should the visitor agent carried to fill the readings takes from each visited clients and the client ID that the visitor agent takes from.

3.4.2 Data Flow Diagrams at Clients(Hosts) Sides

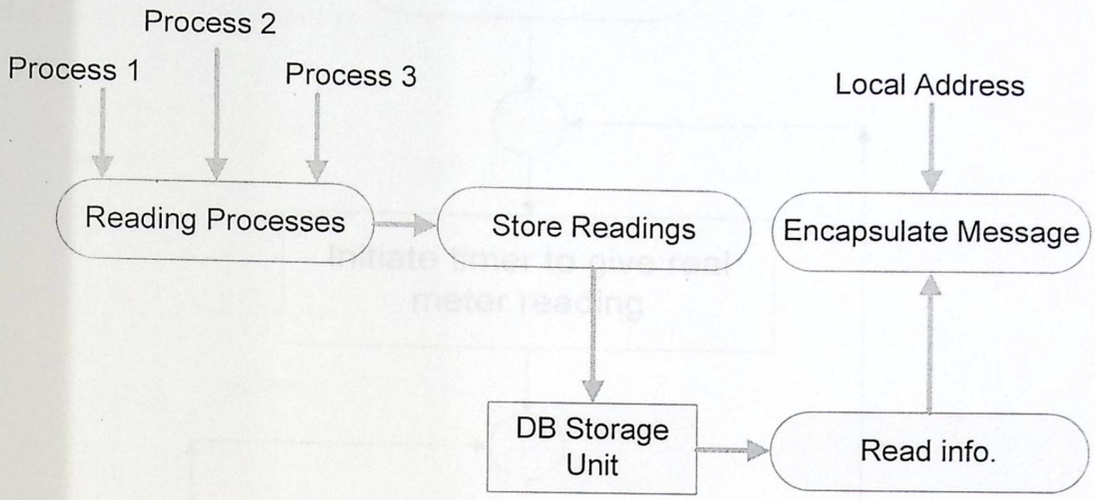


Figure 3.9: DataFlow of a power meter that collect the sub process (loads) values.

The figure above shows the power meter at client(host) machine that collects the consumed power meter value from the sub process(Load) which indicate a real timer value for loads, we assume that exist three loads consumption to measure in our application, we could use more loads or less to measure them.

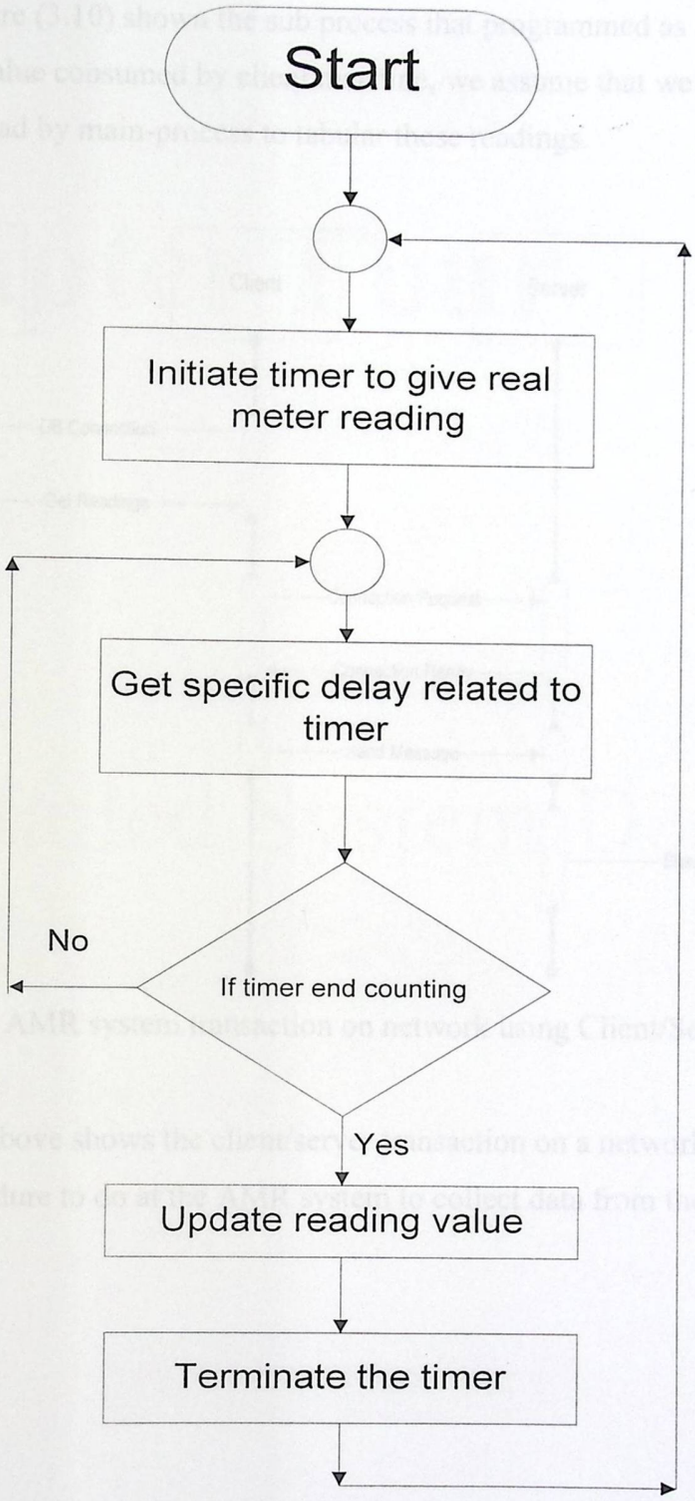


Figure 3.10: Flowchart of the sub process in each client's machine with time delay.

The figure (3.10) shown the sub process that programmed as a timer to express loads value consumed by client machine, we assume that we have 3-load (sub process) read by main-process to tabular these readings.

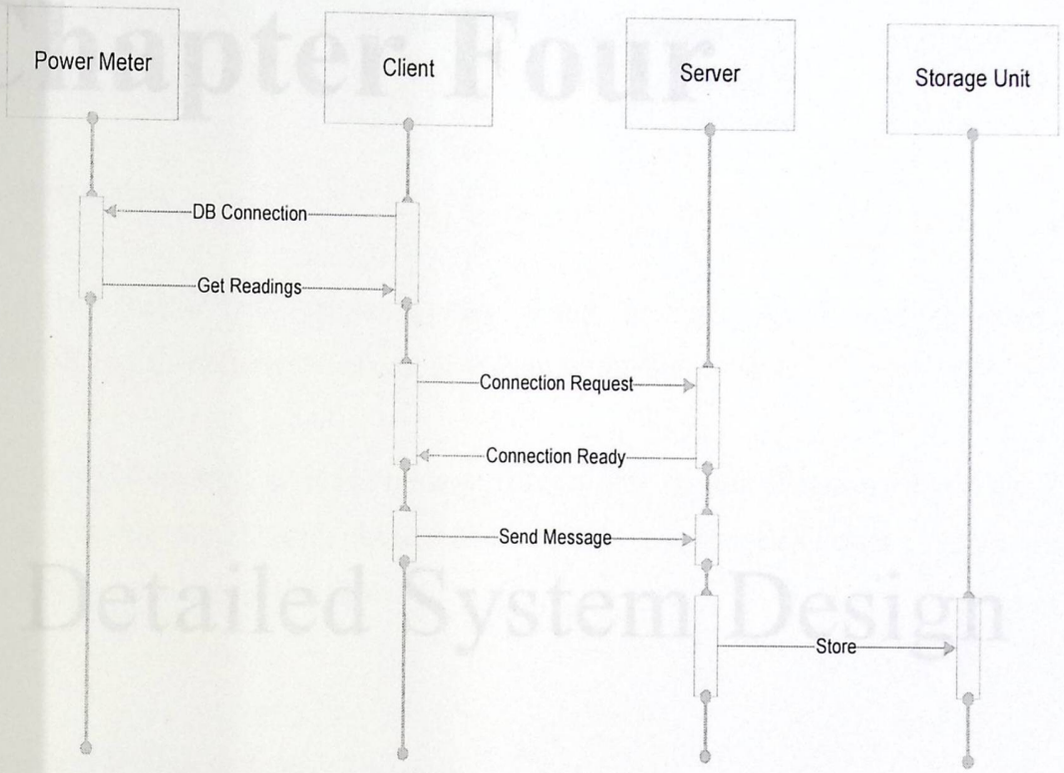


Figure 3.11: AMR system transaction on network using Client/Server system

The figure above shows the client/server transaction on a network and its functional procedure to do at the AMR system to collect data from the system.

Chapter Four

Detailed System Design

Chapter Four

Detailed System Design

4.1 Introduction

This chapter will present system design in more detail ,user interface , algorithms and flowcharts for each component of the hole system.

Our system well divided into two phases , first for the client/server and the other is a mobile agent, each phase split into sub component that interact each other in some way to achive their objective .

4.2 General Design options

4.2.1 Software application options

There are many application can be used to programming our system like Java, and C++, but we choose Java application for many reasons, some reasons related with system programmers, other related with libraries that Java application can support, and the main reasons related with the easy use and simple configuration for the application and that it is a standard.

- **programmers reasons**

The programmers of the system have a good background knowledge and good experience that can help them to install, configure and deal with Java application in a good way rather than other application.

Because the application consider as object-oriented programming, the programmers could divided the system work among the them and then collect all subprograms into one programm this option ables programmers to share in building the system and save the time that takes comparing to the serial way in programming.

- **Application Libraries and good userguide**

Java application has a huge build-in library in it, so, it can help the programmer for programming system in easily way, and so, java can configure to install other library if needed easily without confusion.

The java also support userguide file that help the programmer to use application and gives them the necessary assistant while programming.

4.2.2 Network options

The system idea depend on collecting data from different machines which may exist in diffent locationm so, there is necessary to connect that machines by a network.

A network maybe use different protocol, so the system should compatible with these different and also compatible with heterogeneous machines that maybe

appear at network terminals, Our system designed and programmed to support all these consideration .

4.2.3 Environment options

The most powerful system is the system which build with less possible requirement or with requirements that exist in most machines.

Our system uses Runtime environment because our system depend on java programming, so this environment used to execute these java program as well.

Another option is a jar file to able the mobile move from one machine to another as well.

4.3 Detailed description of system components and related flowcharts

This chapter describes the system design, including the application, the database. Topics that are to be covered in this chapter :

1. Functional design: This part will be implemented by means of flow charts and a description of interface and constraint.
2. Database Design: This part will present the complete design for the system database tables, fields and relations.

4.3.1 Functional Design

This section describes the functional design for the modules in the software system, it also present the description of the I/O Exceptions, the constraints. All of these are covered in means of diagrams to make a better understanding of the system services and functions.

4.3.1.1 senderClient.java

- Description

It is a program setup on client side responsible for creating a specific socket on itself to connect with server socket where the client can write its data on its socket and transfer immediately to the server socket after insure there is a connection between client and server machine, and opening a streams between them for reading and writing the clients on a server socket to enable the server from reading the data in the client message.

A connection between the client machine and its database will create and take copy of data that stored in database to a message with some valuable information related with that client for a server, client Identifier; subprocesses Identifier with reading that express a bout the consumed power.

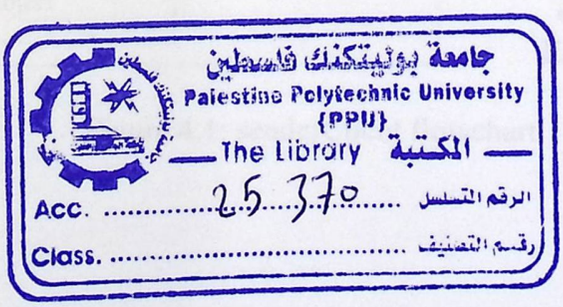
The Query instructions determine the reading from database, and then transfer read data to a message.

After sending message to server, the connection closed with the database, streams and the socket.

- Constraints

a- Should create a socket to connect with server.

b- Should be a specific Driver for used Database, and this cause in connect the bridge between the ODBC that special for Database itself and JDBC that special for the Java program.



- Flowchart

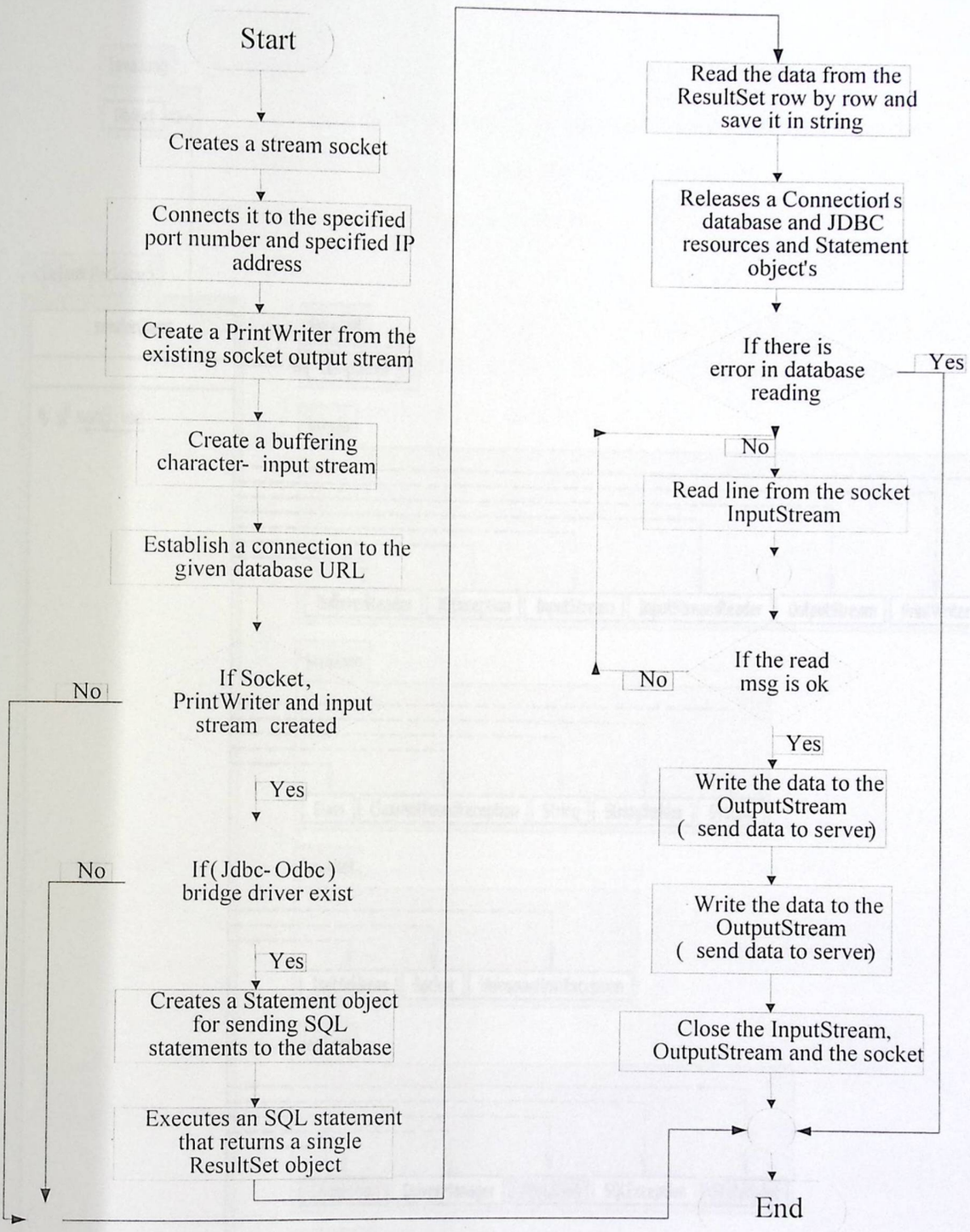


Figure 4.1: senderClient flowchart

- Unified Model Language(UML design)

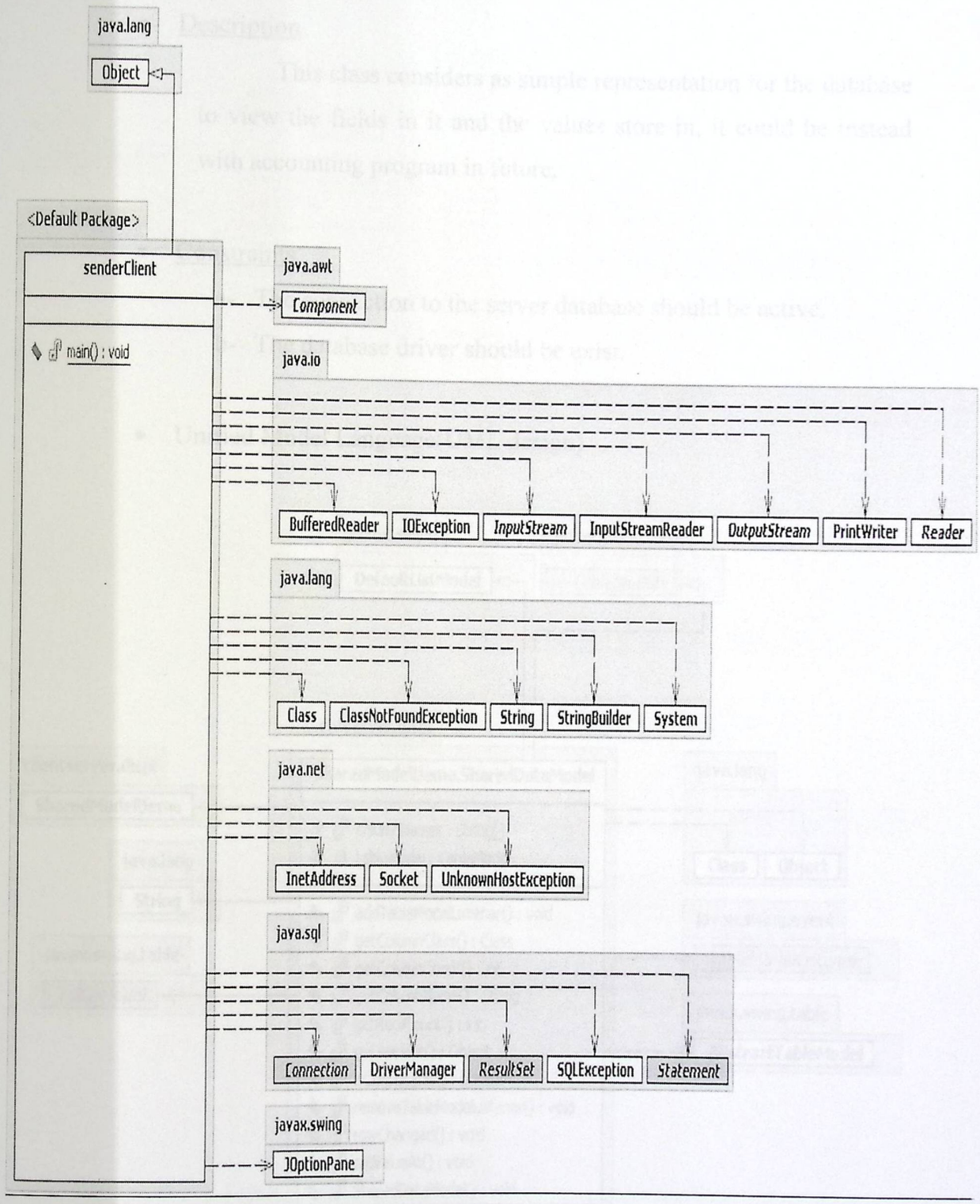


Figure 4.2: senderClient UML design

4.3.1.2 SharedModelDemo.java

- Description

  This class considers as simple representation for the database to view the fields in it and the values store in, it could be instead with accounting program in future.

- Constraints

  a- The connection to the server database should be active.

  b- The database driver should be exist.

- Unified Model Language(UML design)

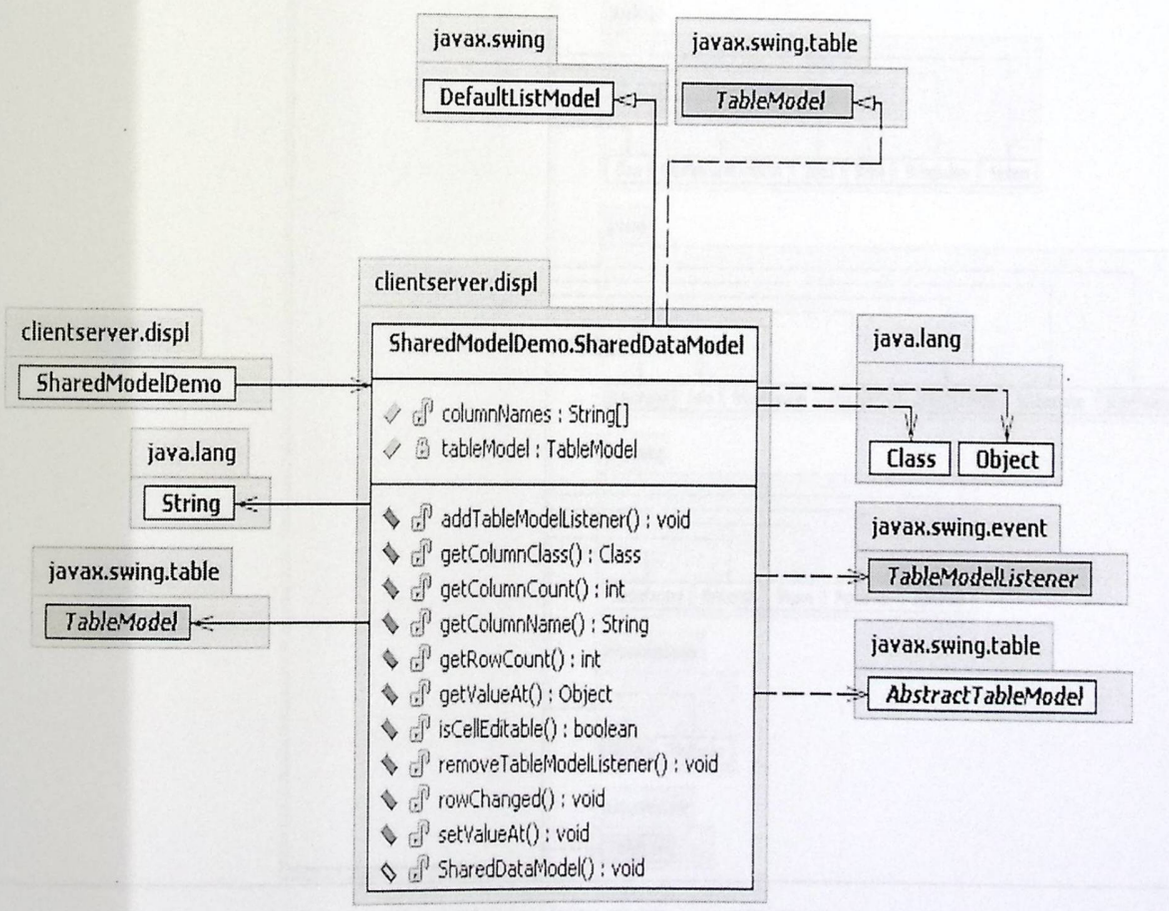


Figure 4.3: SharedModelDemo.SharedDataModel UML design

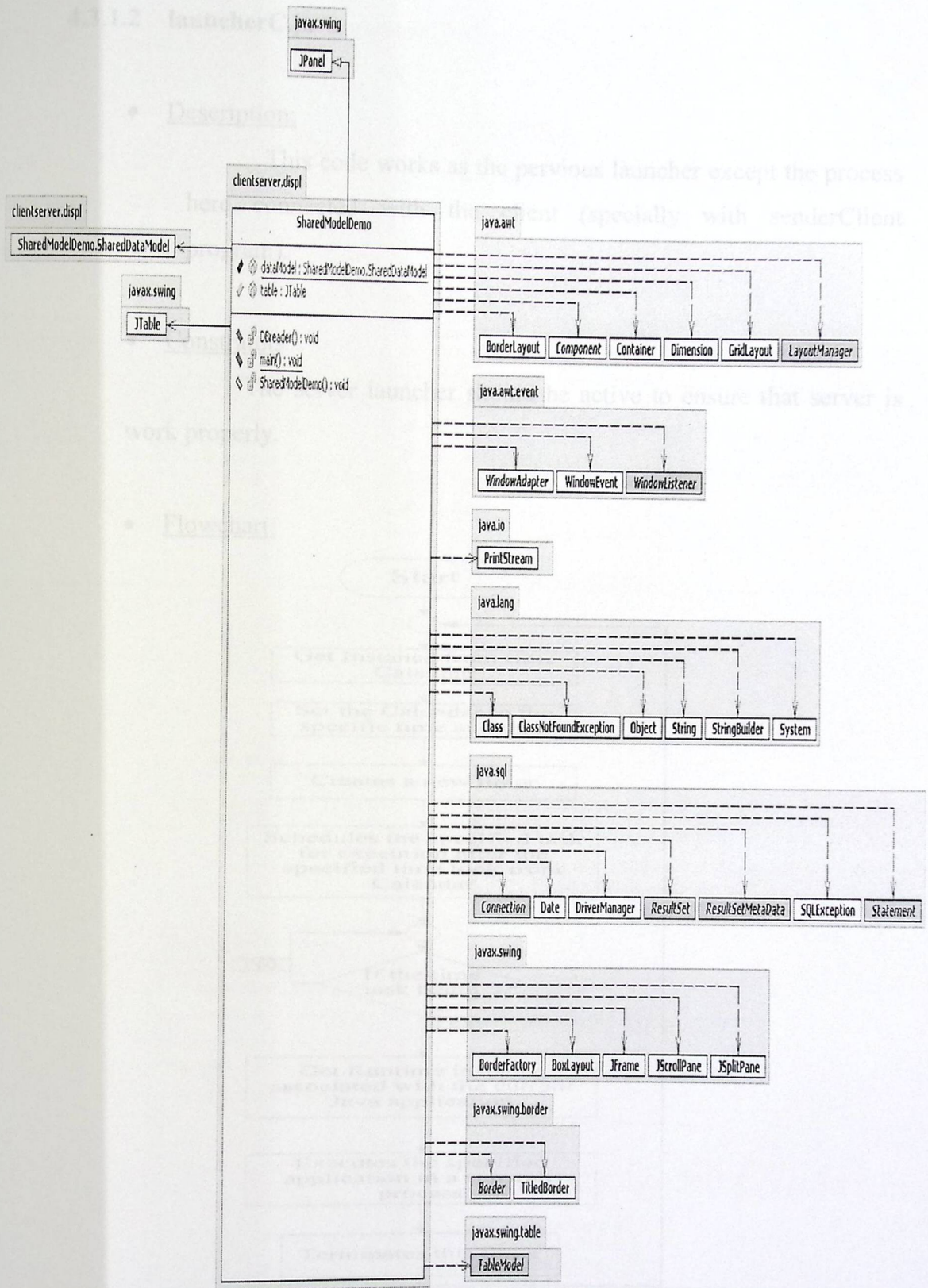


Figure 4.4: SharedModelDemo UML design

4.3.1.2 launcherC.java

- Description:

This code works as the pervious launcher except the process here connected with the client (specially with senderClient program).

- Constraints

The server launcher should be active to ensure that server is work properly.

- Flowchart:

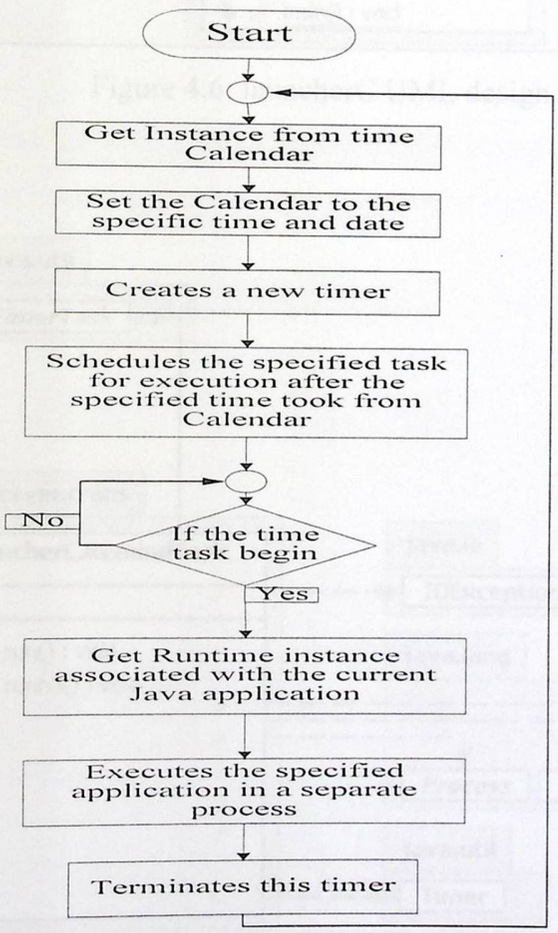


Figure 4.5: launcherC flowchart

- Unified Model Language(UML design):

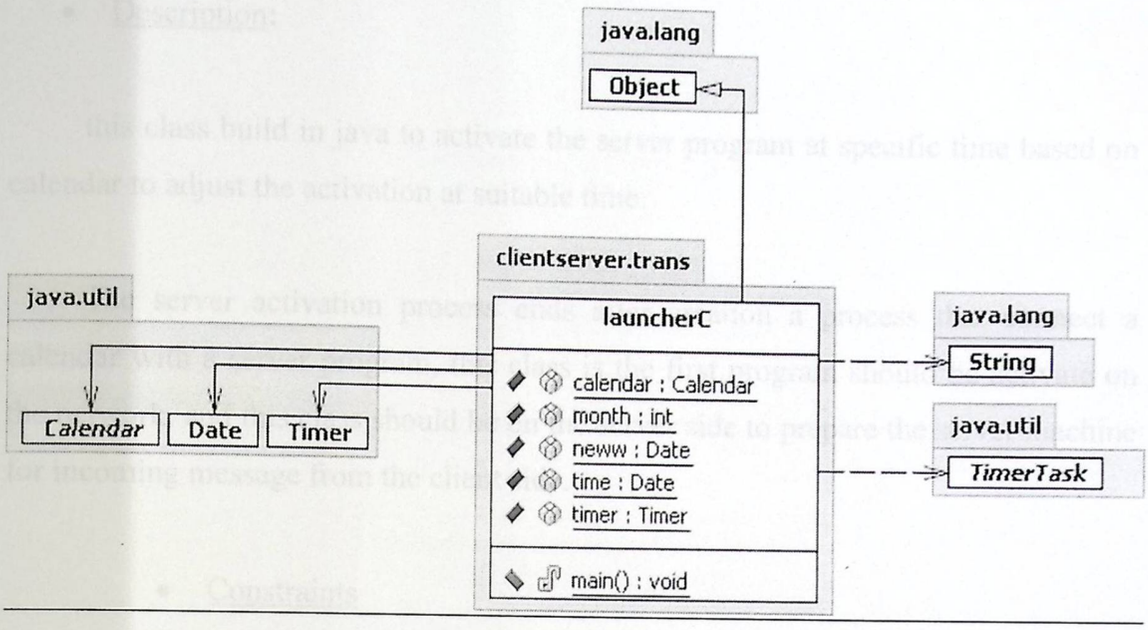


Figure 4.6: launcherC UML design

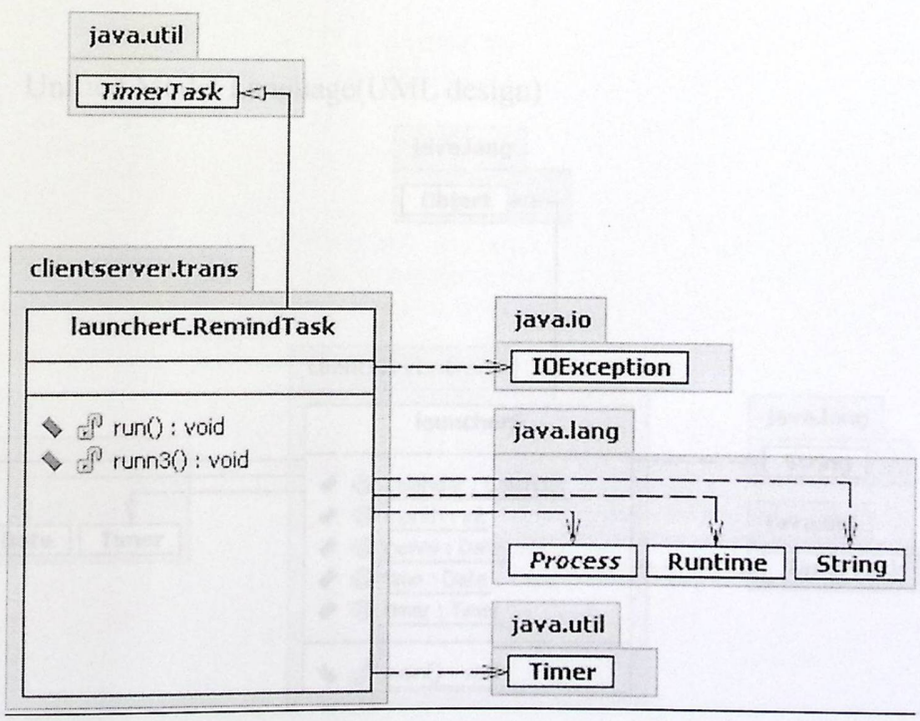


Figure 4.7: launcherC.RemindTask UML design

4.3.1.3 launcherS.java:

- Description:

this class build in java to activate the server program at specific time based on calendar to adjust the activation at suitable time.

The server activation process ends after creation a process that connect a calendar with a server program, this class is the first program should be activate on the network, and this class should be on the server side to prepare the server machine for incoming message from the client side.

- Constraints

The class should has ability to interact with user by create a message to tell if there is a problem in connecting process with the server program.

- Unified Model Language(UML design)

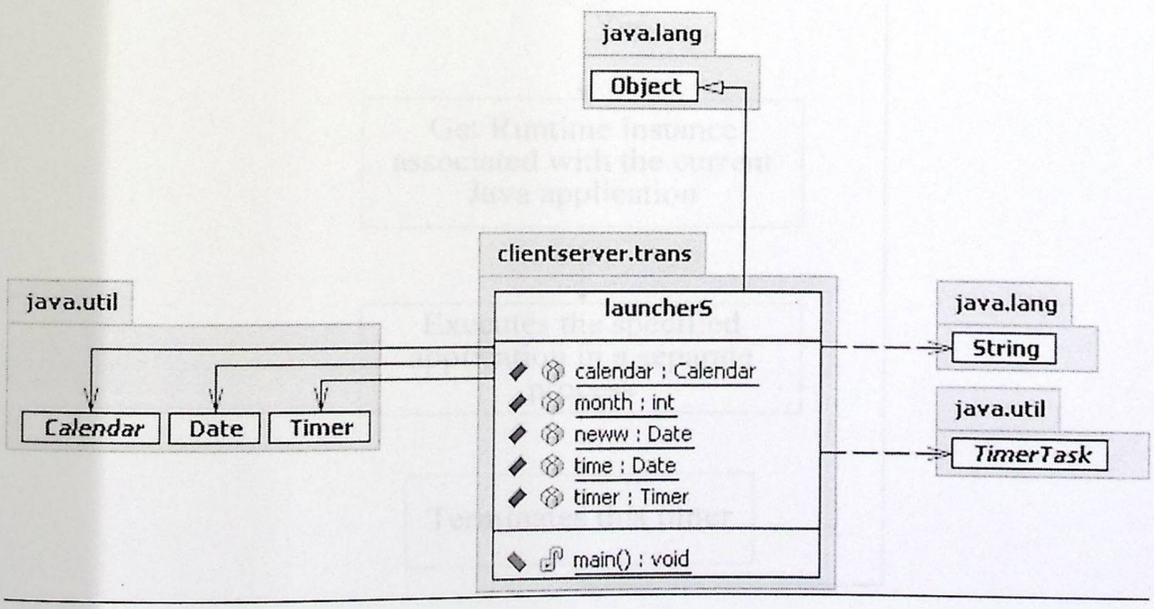


Figure 4.8: launcherS.RemindTask UML design

- Flowchart

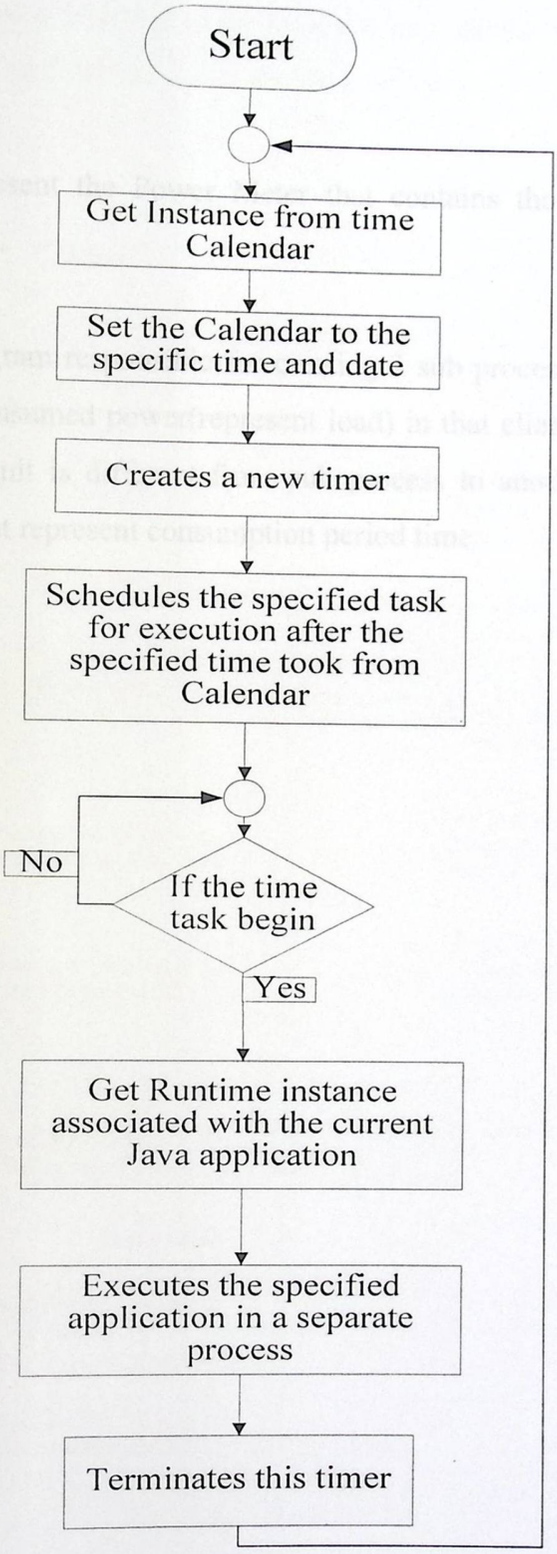


Figure 4.9: launcherC UML design

4.3.1.4 mainP.java

- Description:

This code represent the Power Meter that contains the User Interface to explain the Power work.

The mainP program responsible for creating 3 sub processes , each of them represent amount of consumed power(represent load) in that client, the cost of each consumed power per unit is different from sub process to another this difference result from the timer that represent consumption period time.

- Constraints:

  a- the connection to the local database should be active.

  b- The database driver should be exist.

- Flowchart

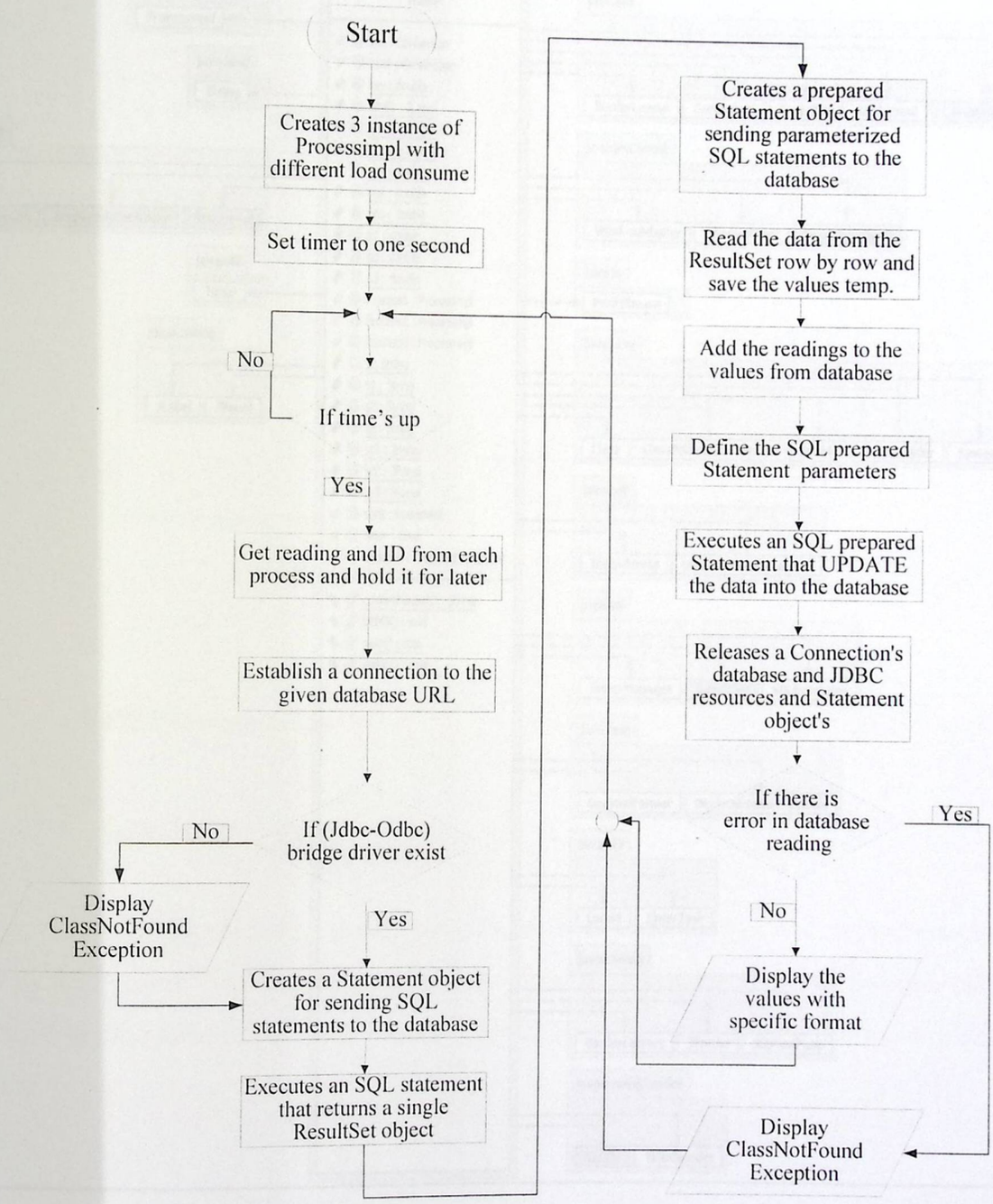


Figure 4.10 mainP flowchart

- Unified Model Language(UML design)

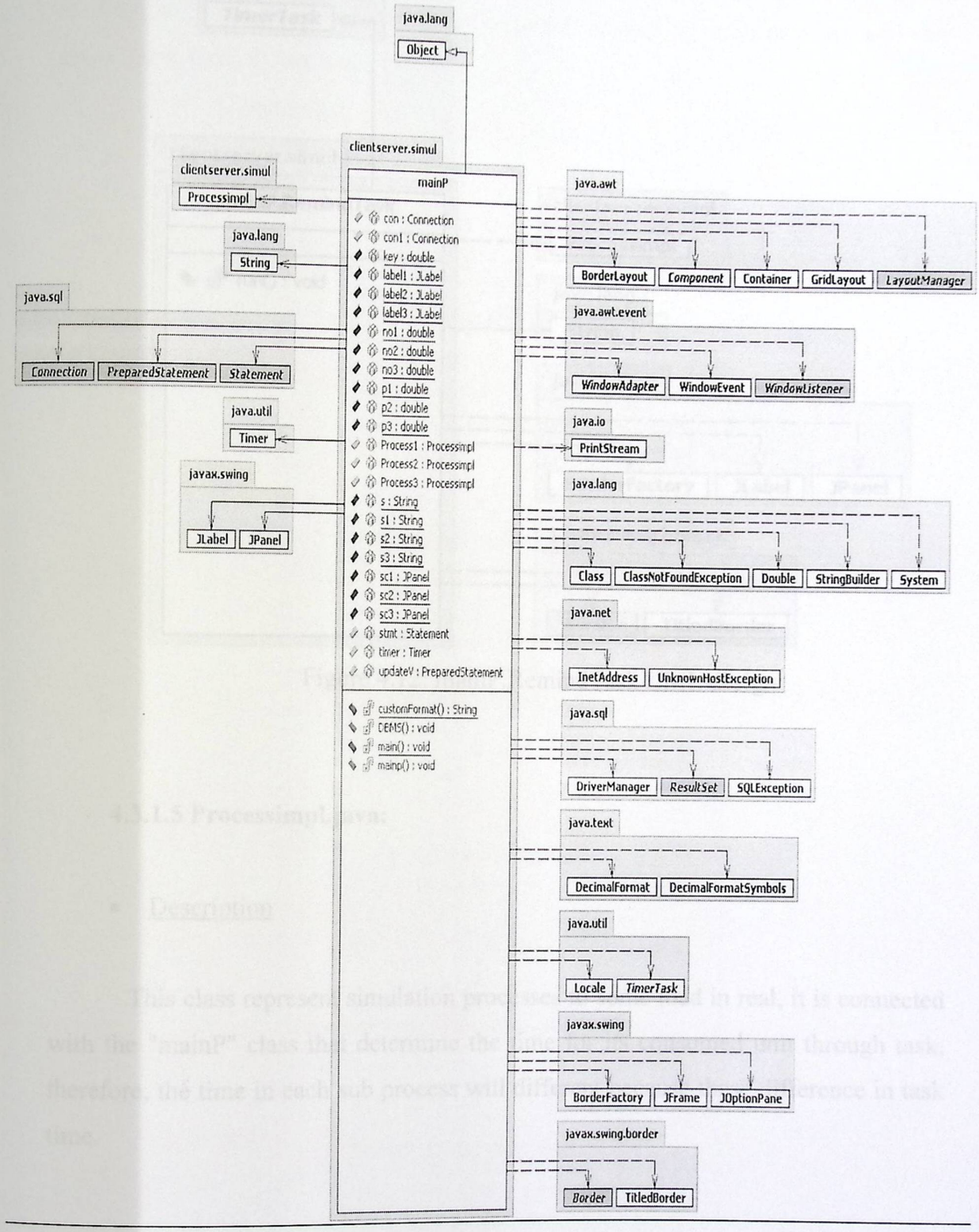


Figure 4.11: mainP UML design

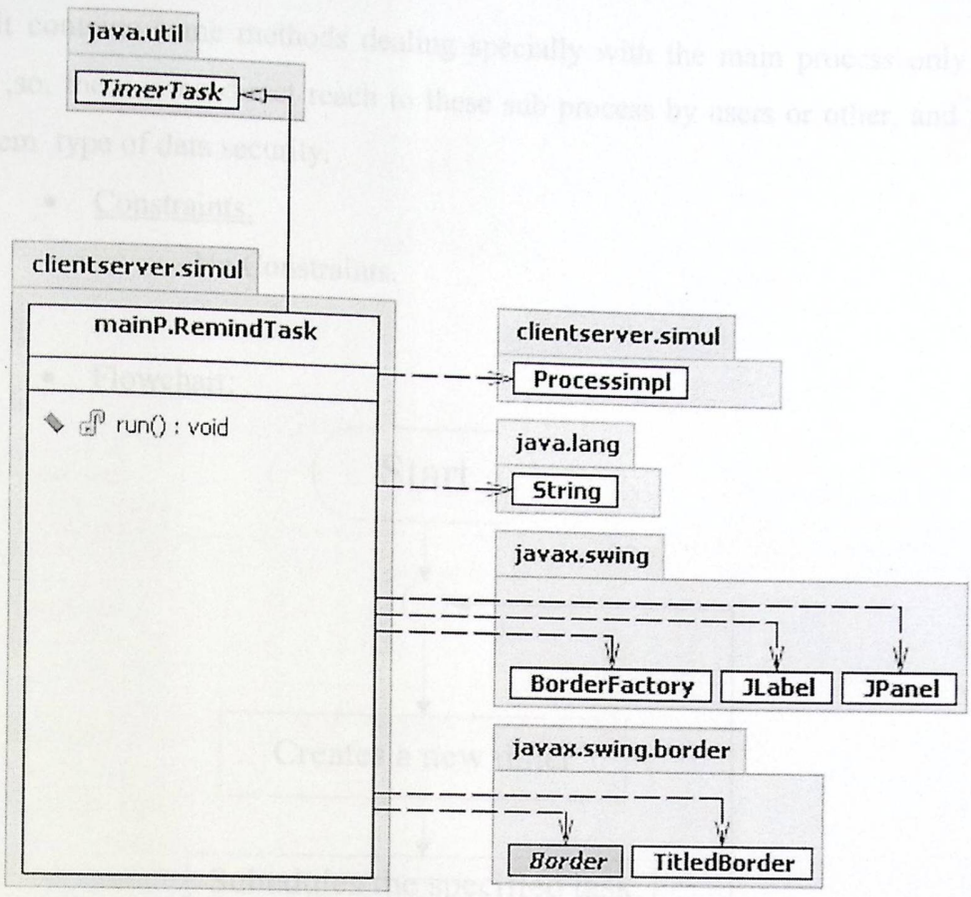


Figure 4.12: mainP.RemindTask UML design

4.3.1.5 Processimpl.java:

- Description

This class represent simulation processes to some load in real, it is connected with the "mainP" class that determine the time for its consumed unit through task, therefore, the time in each sub process will different because these difference in task time.

It contains some methods dealing specially with the main process only for reading ;so, there is no direct reach to these sub process by users or other, and this gives them type of data security.

- Constraints:

  No Constraints.

- Flowchart:

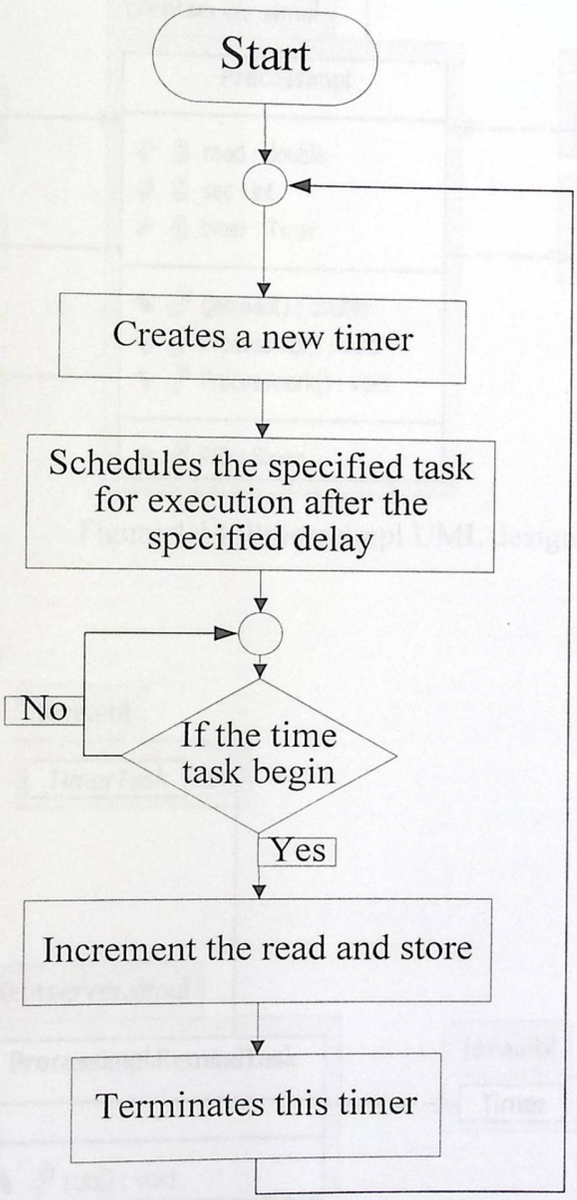


Figure 4.13: Processimpl flowchart

- Unified Modeling Language(UML design):

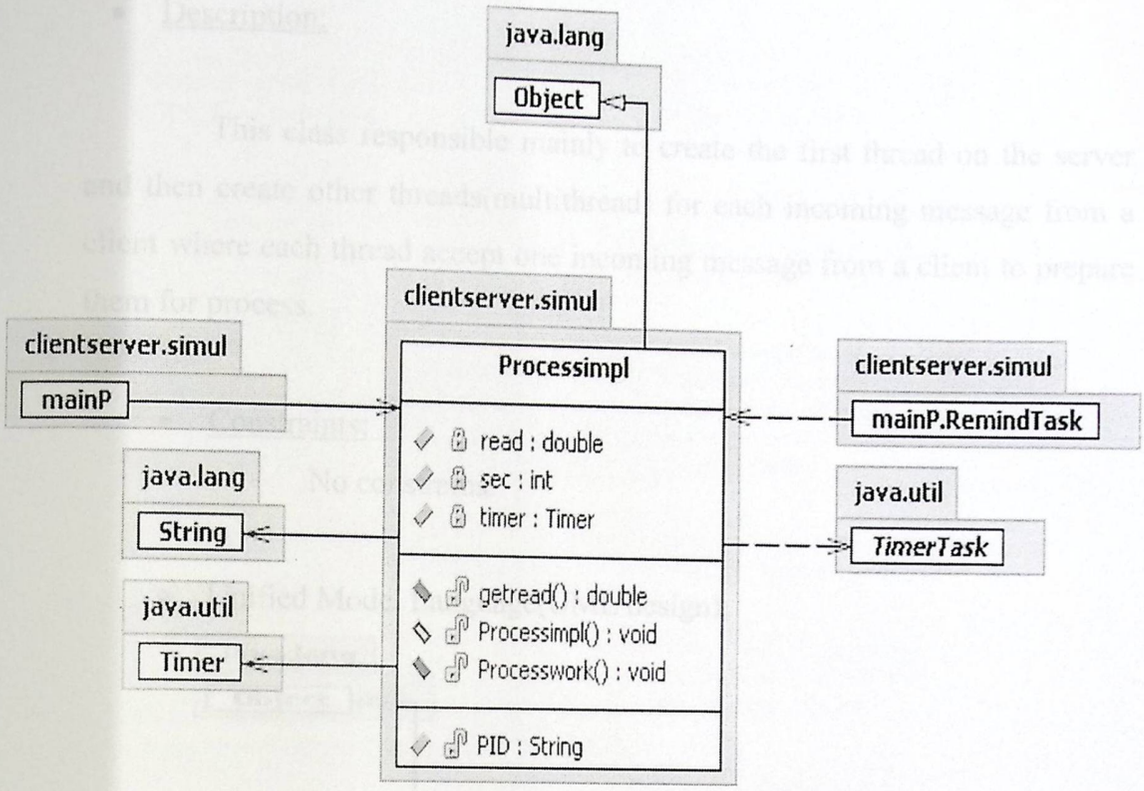


Figure 4.14: Processimpl UML design

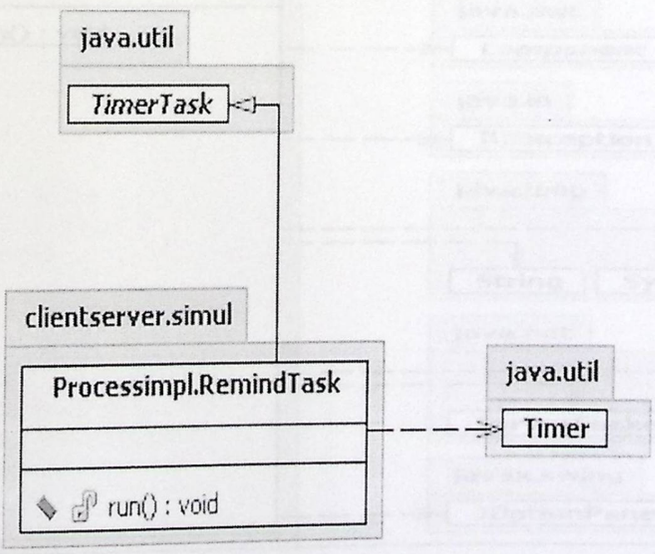


Figure 4.15: Processimpl.RemindTask UML design

4.3.1.6 KKMultiServer.java:

- Description:

This class responsible mainly to create the first thread on the server and then create other threads(multithread) for each incoming message from a client where each thread accept one incoming message from a client to prepare them for process.

- Constraints:

No constrains.

- Unified Model Language(UML design):

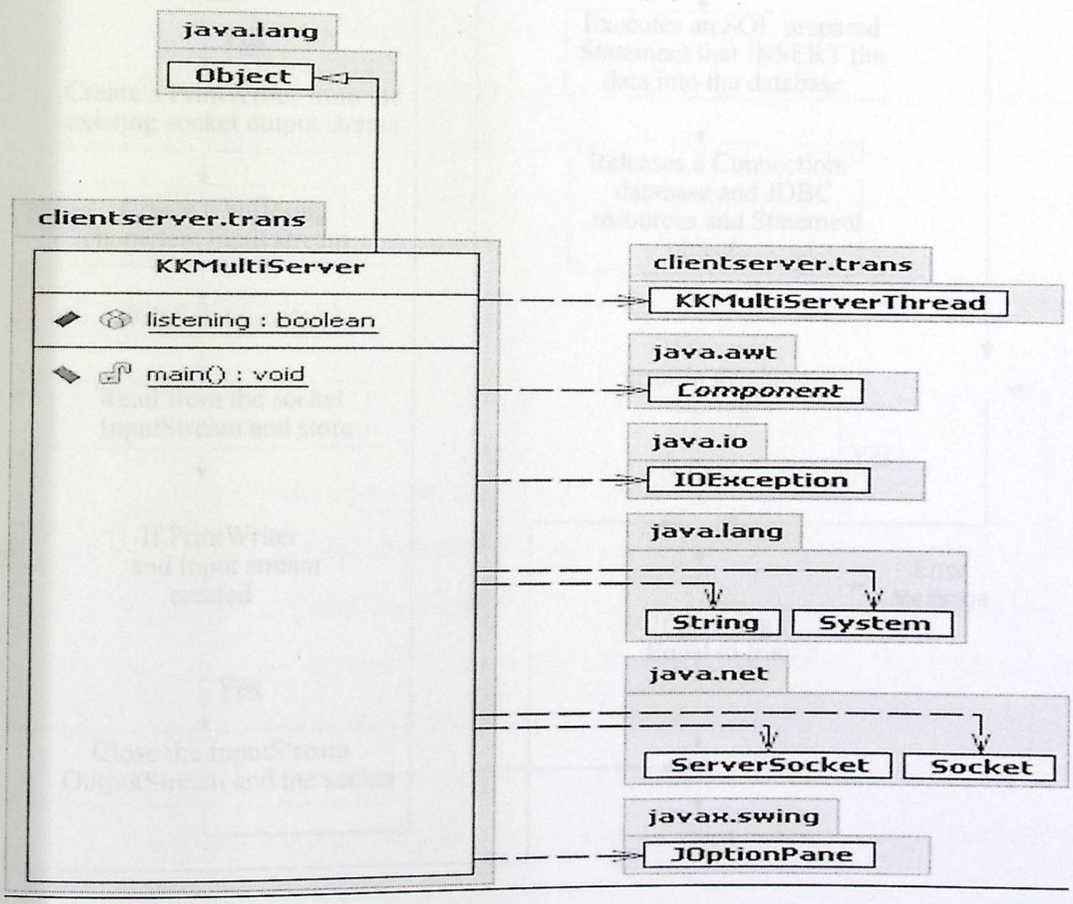


Figure 4.16 KKMultiServer UML design

- Flowchart:

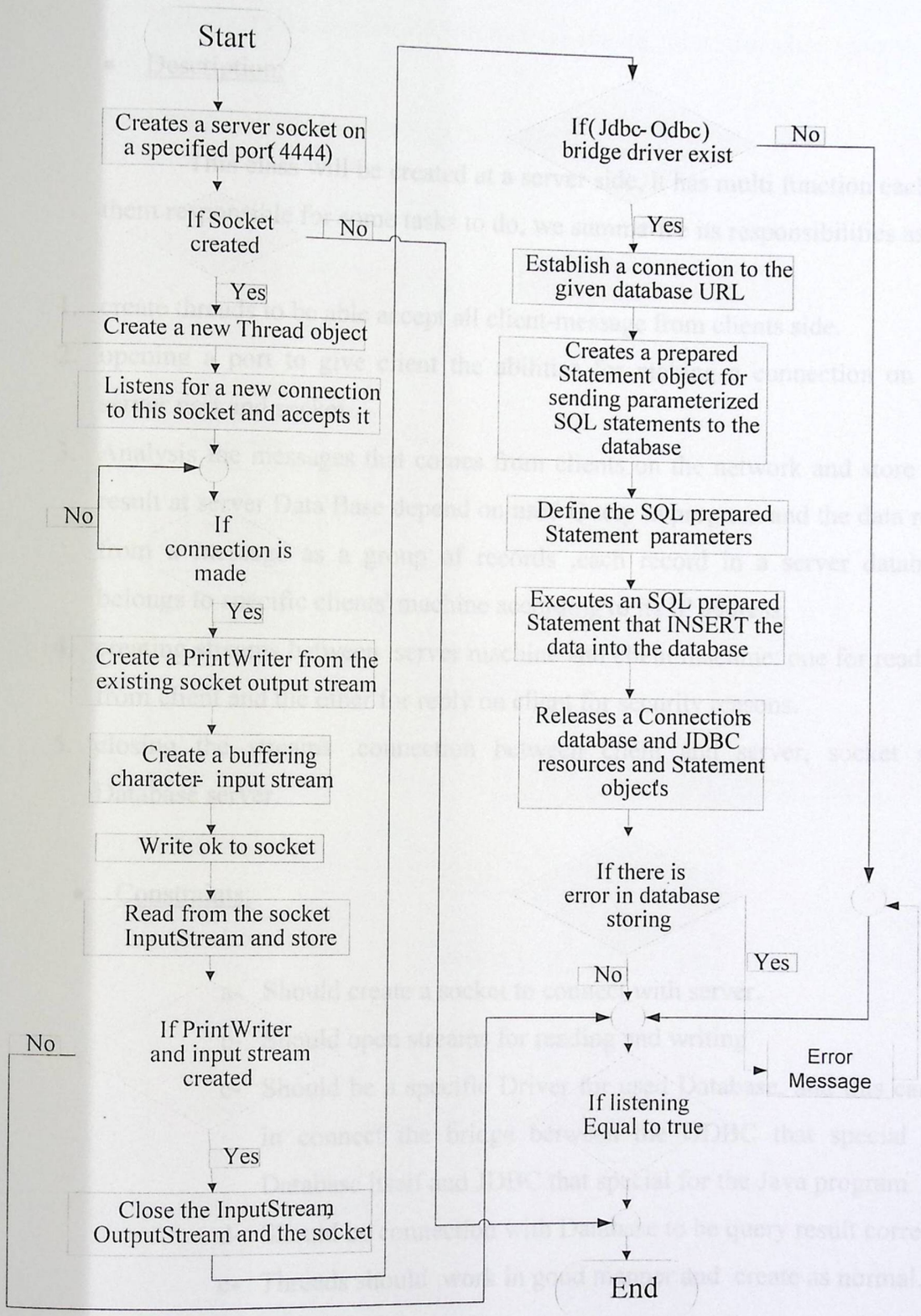


Figure 4.17 KKMultiServer flowchart

4.3.1.7 KKMultiServerThread.java:

- Description:

This class will be created at a server side, it has multi function each of them responsible for some tasks to do, we summarize its responsibilities as:

1. create threads to be able accept all client-message from clients side.
2. opening a port to give client the abilities for making a connection on the server port and socket.
3. Analysis the messages that comes from clients on the network and store the result at server Data Base depend on used Query in program and the data read from a message as a group of records ,each record in a server database belongs to specific clients' machine according to its IP address.
4. creating streams between server machine and client machine, one for reading from client and the other for reply on client for security reasons.
5. closing the streams ,connection between client and server, socket and Database server.

- Constraints:

a- Should create a socket to connect with server.
b- Should open streams for reading and writing
c- Should be a specific Driver for used Database, and this cause in connect the bridge between the ODBC that special for Database itself and JDBC that special for the Java program.
d- Should be connection with Database to be query result correct.
e- Threads should work in good manner and create as normal.

- Flowchart:

  This class inhiret from KKMultiserver classm so, its flowchart exist there

- Unified Model Language(UML design):

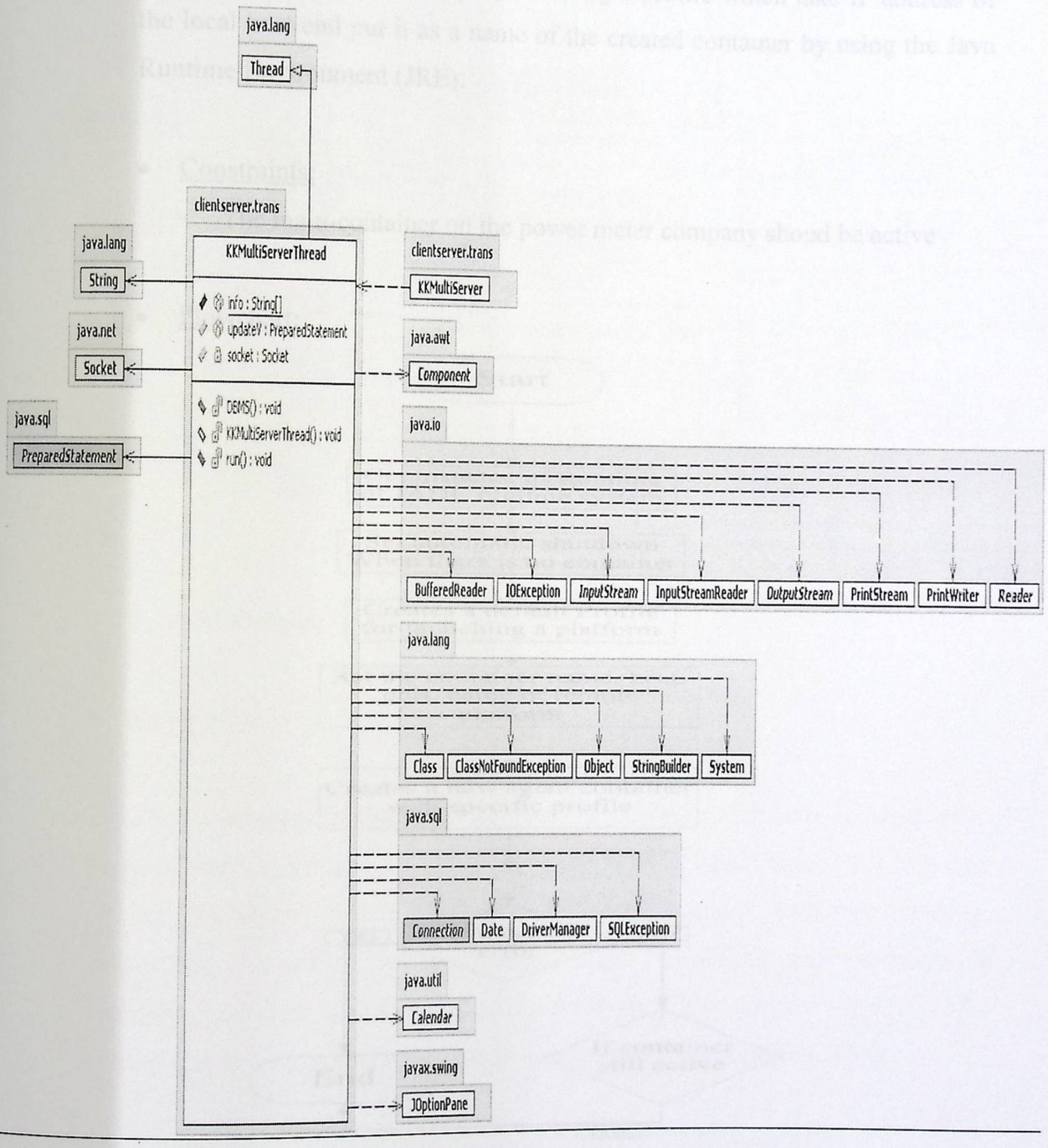


Figure 4.18: KKMultiServer UML design

4.3.1.8 Containercreat

- Description

This class consider as small program which create a container in Power company platform through creating a profile which take IP address of the local host and put it as a name of the created container by using the Java Runtime Environment (JRE).

- Constraints:

The main-container on the power meter company shoud be active

- Flowchart:

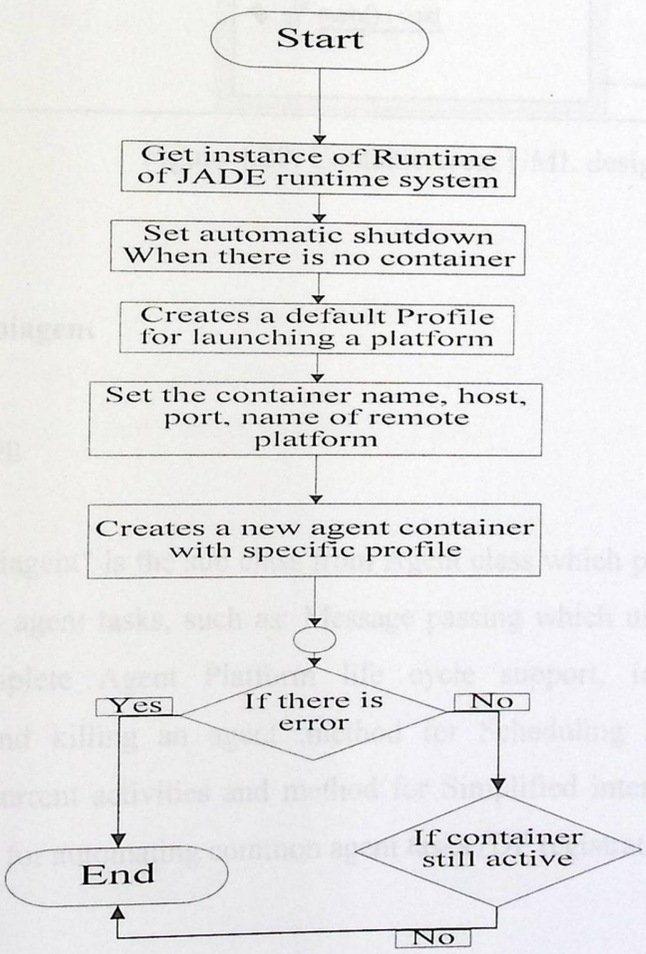


Figure 4.19: Containercreat flowchart

- Unified Modeling Language

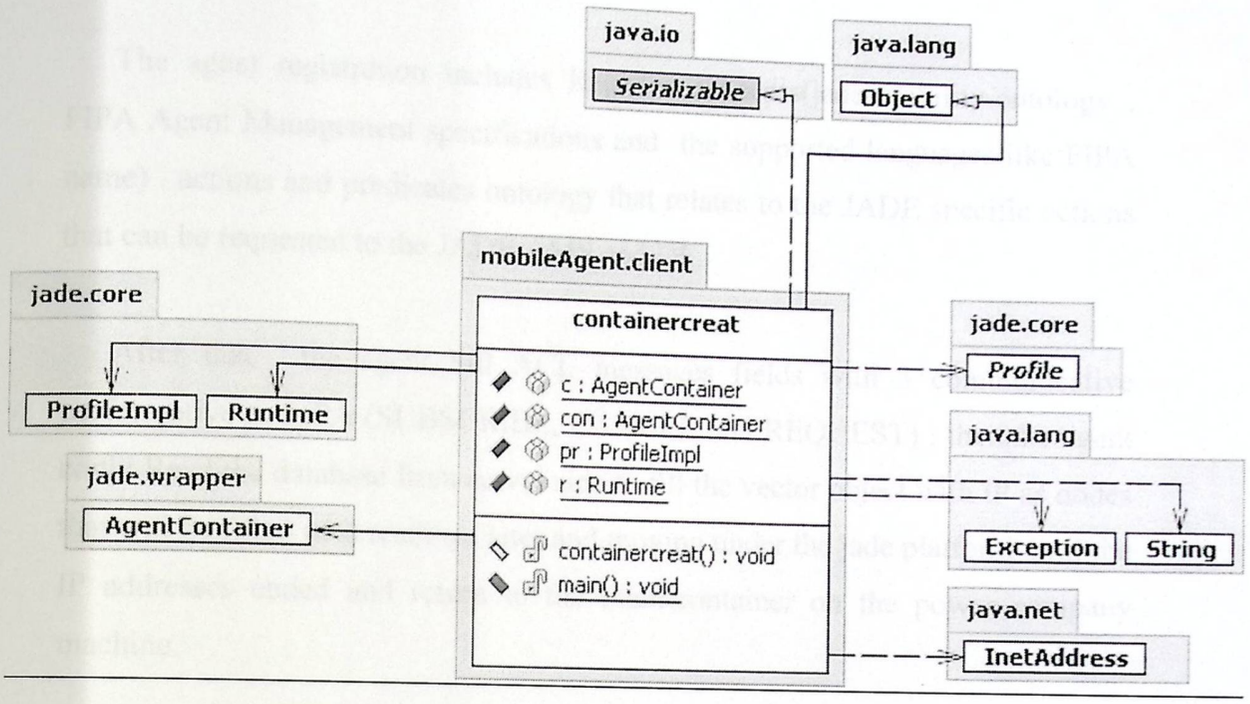


Figure 4.20: Containercreat UML design

4.3.1.9 Mobiagent

- Description

The "mobiagent" is the sub class from Agent class which provide methods to perform basic agent tasks, such as: Message passing which using ACLMessage objects ,Complete Agent Platform life cycle support, including starting, suspending and killing an agent ,method for Scheduling and execution of multiple concurrent activities and method for Simplified interaction with *FIPA* system agents for automating common agent tasks (DF registration, etc.).

The agent be active in jade run time after adding registration, Loading Table and Moving behaviours.

The agent registration includes Register concepts(jade-mobility-ontology , FIPA Agent Management specifications and the supported languages like FIPA name) , actions and predicates ontology that relates to the JADE specific actions that can be requested to the JADE AMS and DF.

After that , the agent Fill ACL messages fields with 3 communicative performative to FIPA (SUBSCRIBE, CANCEL and REQUEST) , then,the agent could Read the database from server side to fill the vector object with IP as nodes that will be filled with readings later and moving under the jade platform until the IP addresses ended and return to the main-container on the power company machine.

- Constraints

a. Jade environment should be active.
b. MA Should be registered at the jade environment.
c. Should contain setup() method in there.

- Flowchart

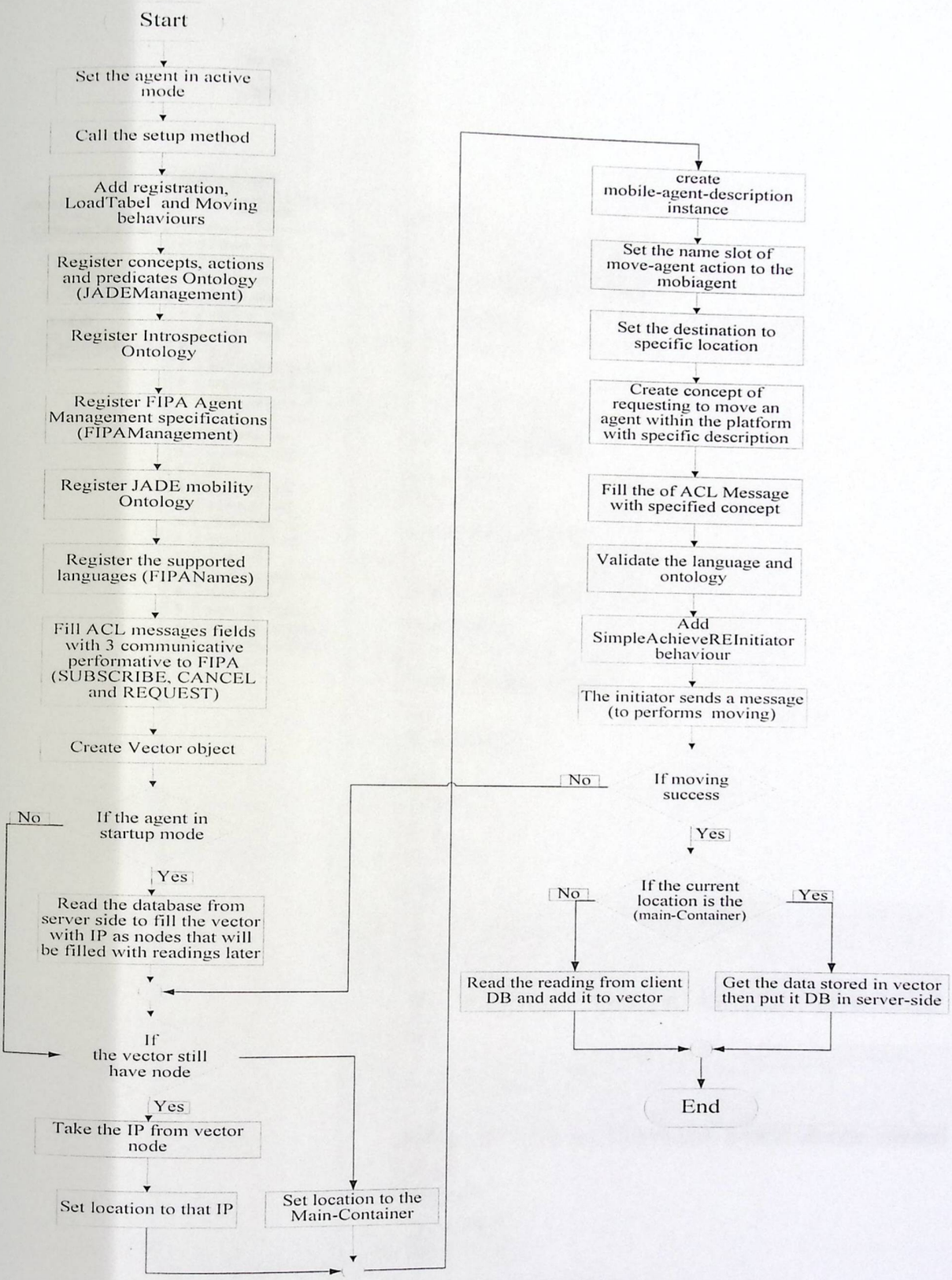


Figure 4.21: Mobiagent flowchart

- Unified Modeling Language

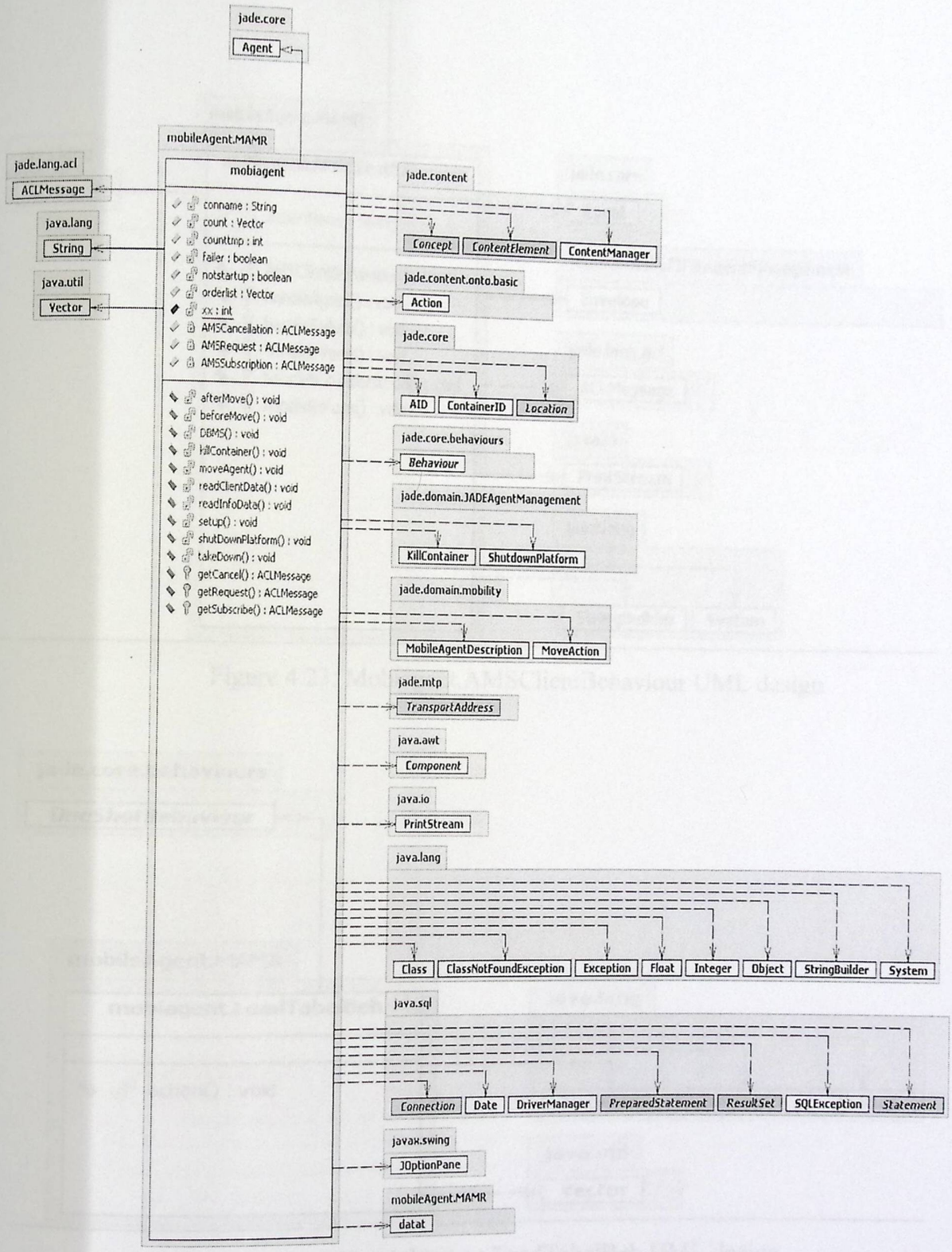


Figure 4.22: Mobiagent UML design

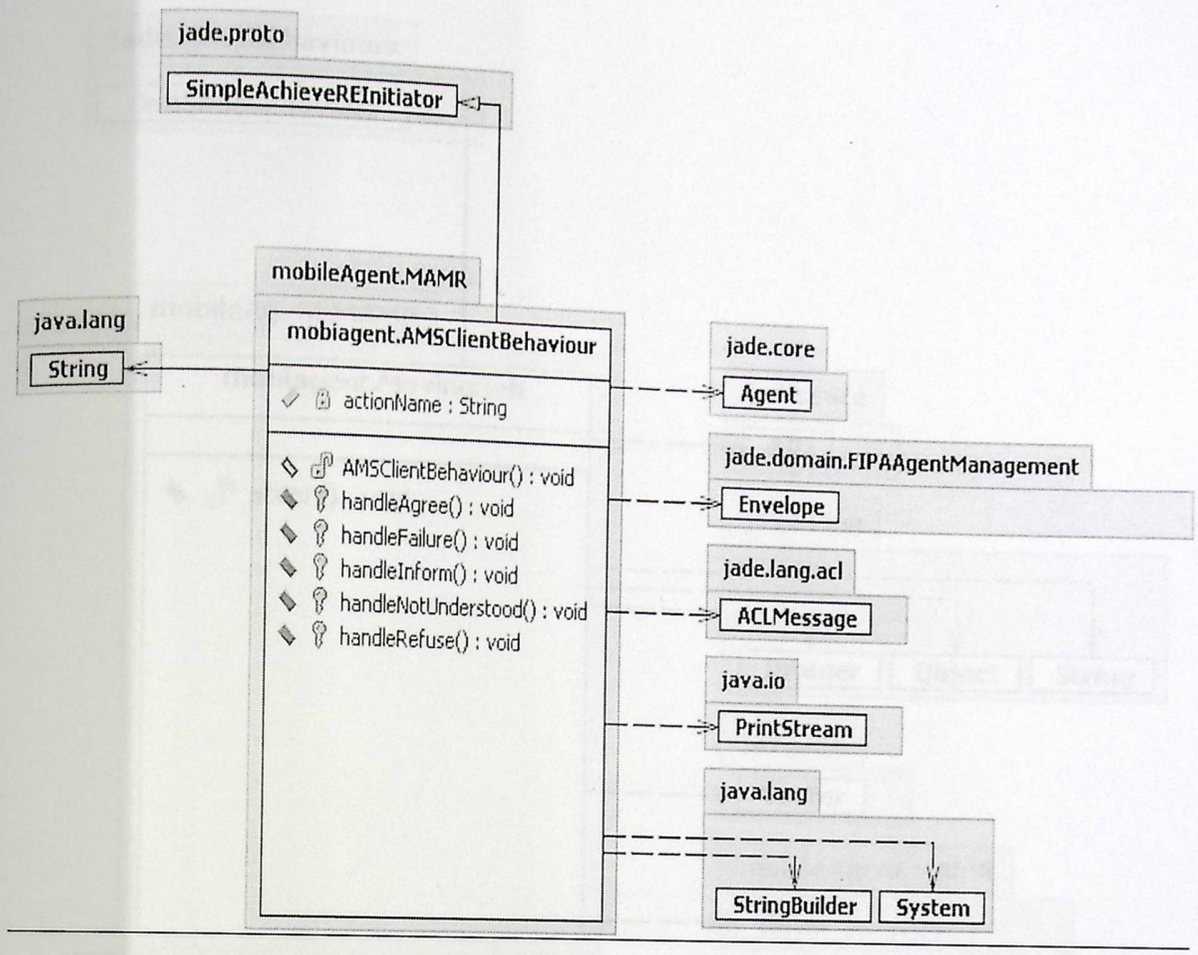


Figure 4.23: Mobiagent.AMSClientBehaviour UML design

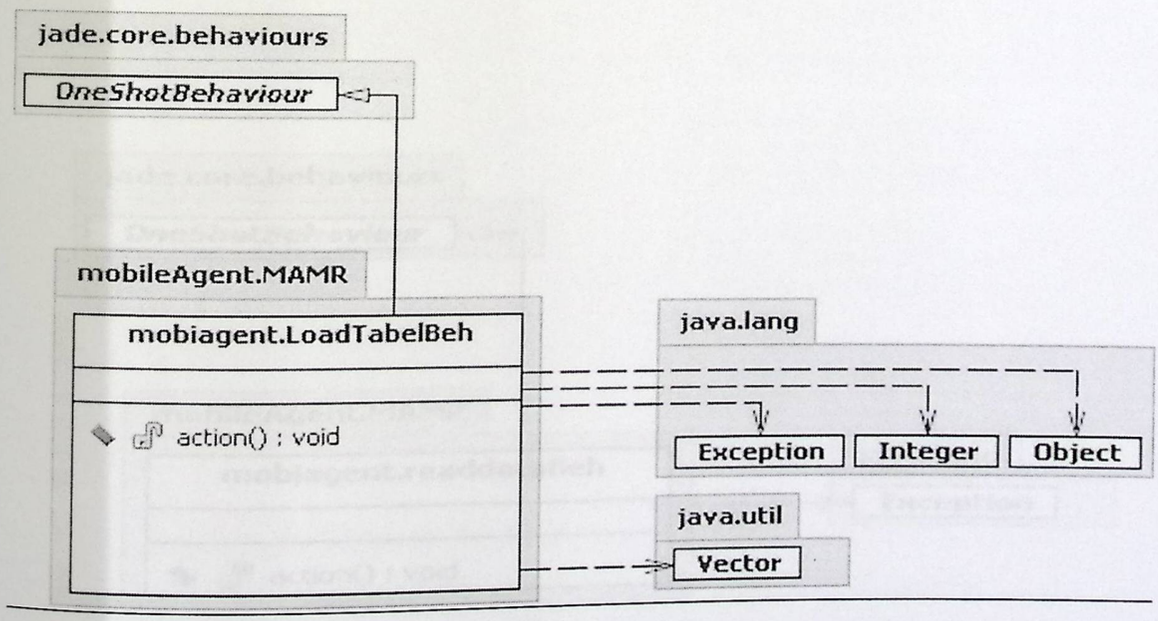


Figure 4.24: Mobiagent.LoadTabelBeh UML design

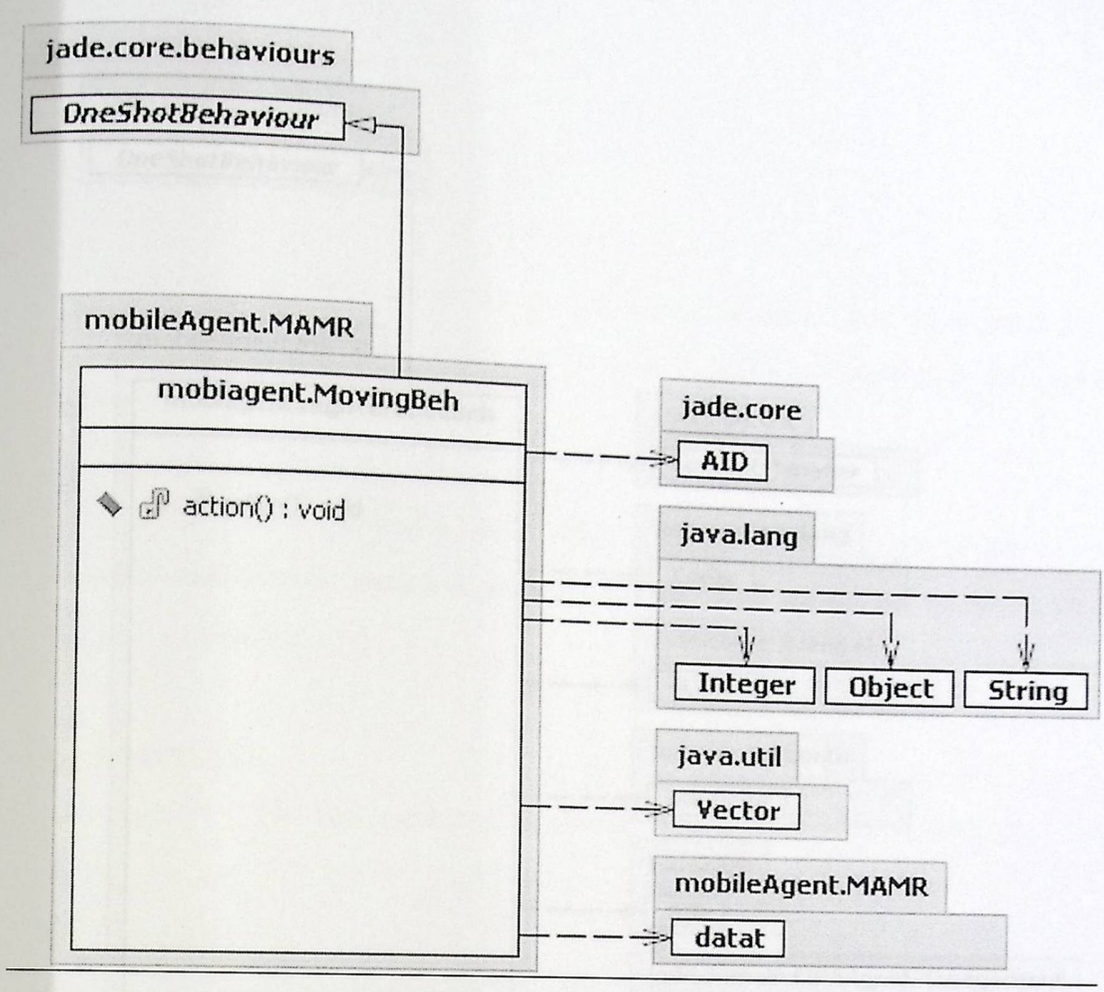


Figure 4.25: Mobiagent.MovingBeh UML design

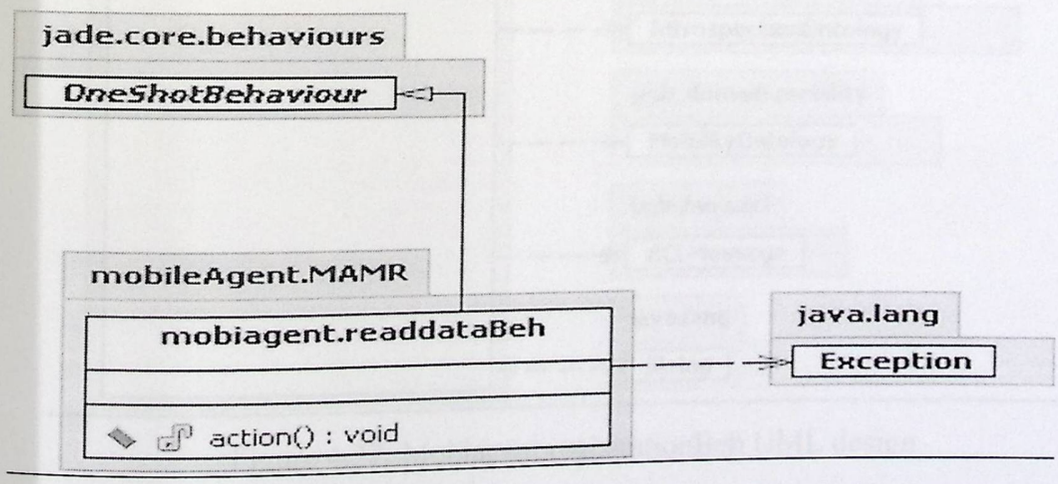


Figure 4.26: Mobiagent.readdataBeh UML design

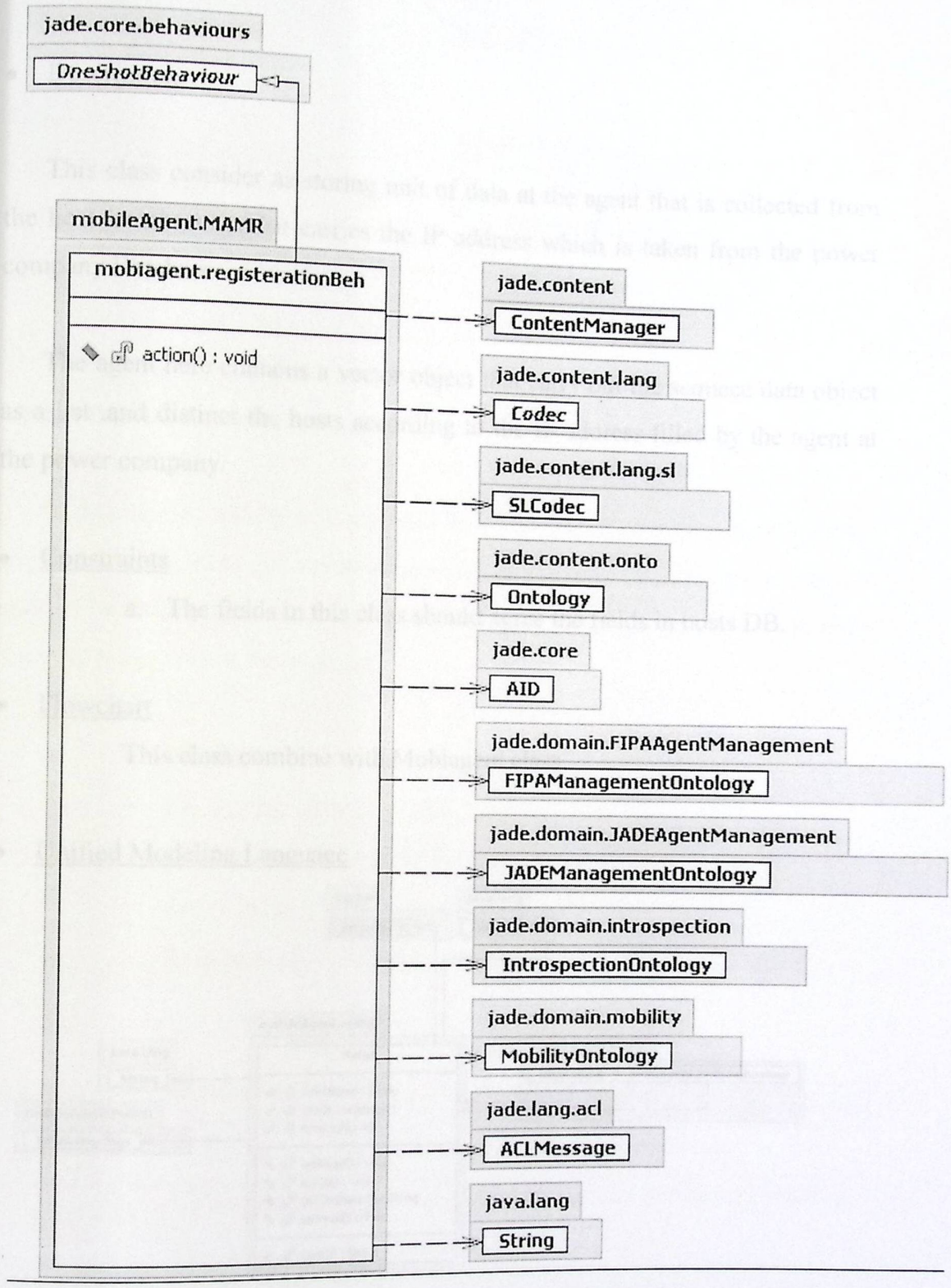


Figure 4.27: Mobiagent.registrationBeh UML design

4.3.1.10 Datat

- Description

This class consider as storing unit of data at the agent that is collected from the host Database ,and it carries the IP address which is taken from the power company Database.

The agent here contains a vector object that carry out the sequece data object as a list ,and distinct the hosts according to the IP address filled by the agent at the power company.

- Constraints

a. The fields in this class should agree the fields in hosts DB.

- Flowchart

This class combine with Mobiagent class .

- Unified Modeling Language

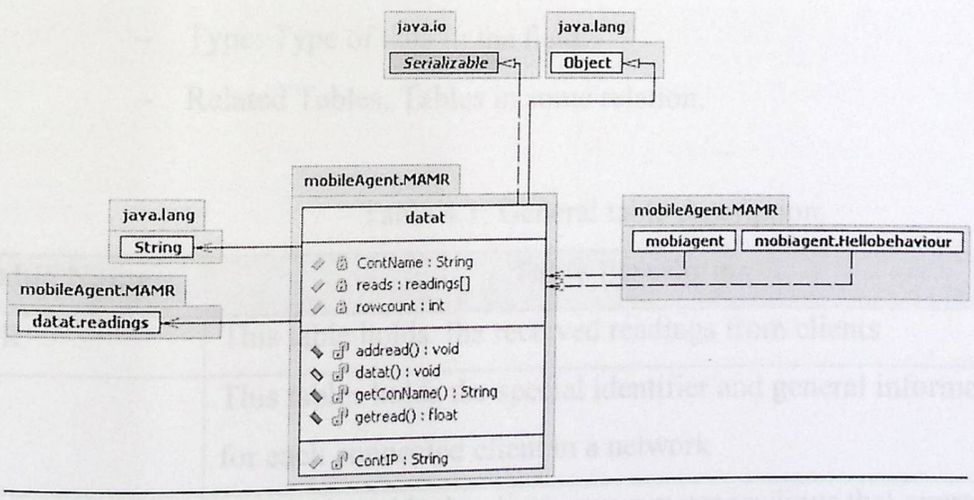


Figure 4.28: Datat UML design

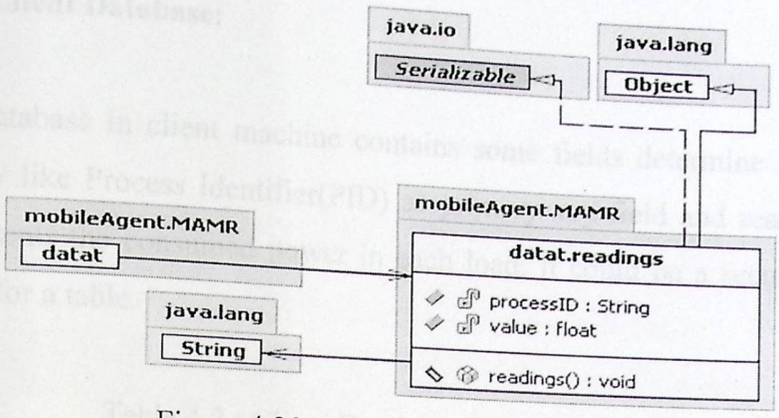


Figure 4.29: Datat.readings UML design

4.3.2 Database Design

In this section database design will be discussed in details, including tables descriptions, fields, keys and relations between tables. For our system the database consists of two table that are related through relations and keys The following notes represent naming used in this chapter:

- PK: Primary Key of a table.
- FK: Foreign Key of a table.
- Length: Size of the field in bites.
- Type: Type of data in the field.
- Related Tables: Tables in some relation.

Table 4.1: General table description.

Table Name	Table Description
PMPR	This table holds the received readings from clients
Info	This tables holds the special identifier and general information for each connected client in a network
PMPT	This table holds the client power meter readings that express the consumed client power.

4.3.2.1 Client Database:

The database in client machine contains some fields determine the clients' personality like Process Identifier(PID) as primary key field and readings field that represents the consumed power in each load, it could be a secure by set a Password for a table.

Table 4.2 : Client Database description fields

Field	Type	Length	PK	FK	Description
PID	Text	10	Yes	NO	Process Identifier
Read	Text	10	NO	NO	Simulation consumed Power

4.3.2.2 info Database:

This tables exist in a sever , it holds a general information for the consumers who owns the meter device.

Table 4.3 : info Database description fields

Field	Type	Length	PK	FK	Description
IP	Text	50	Yes	No	Internet Protocol Address
Name	Text	50	No	No	Consumer Name
Address	Text	50	No	No	Consumer Location
Phone#	Text	16	No	No	Phone Number
Sign_Update	Date	-	No	No	Registration Date

4.3.2.3 PMPR Table

The table 4.3 holds the meter readings from each client in specified date for future manipulation.

Table 4.3 : PMPR database description fields.

Field	Type	Length	PK	FK	Description
IP	Text	50	Yes	Yes	Internet Protocol Address
Date	Date	-	Yes	No	Registration Date
P1	Text	10	No	No	First Simulated Process
P2	Text	10	No	No	Second Simulated Process
P3	Text	10	No	No	Third Simulated Process
Total	Text	16	No	No	Total consumed Power by all processes

4.3.2.4 Database Modeling

The following diagram (Figure 4.30) shows the database tables, fields, keys and relations using the UML (Unified Modeling Language) notations.

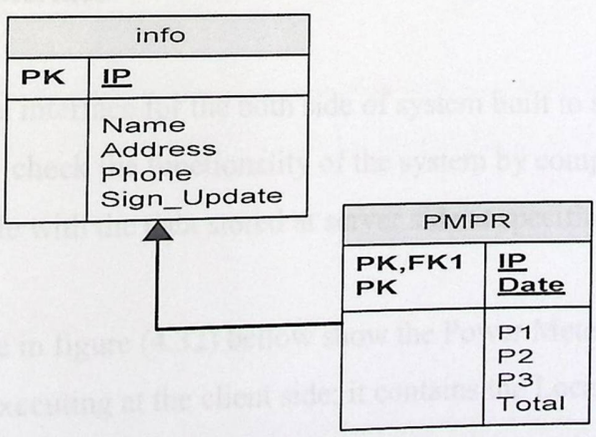


Figure 4.30: Database relation ,fields and keys.

PMPT	
PK	PID
	Read

Figure 4.31: Database fields and keys.

4.4 User-Interface design

4.4.1 Introduction

In this section, we will present the system interface related with database and the error messages that may appear while executing the system in a real world to check the functionals task of the system work properly .

Because our system doesn't accept input data from client or server side, so, there isn't exsist real interface that deal with them.

4.4.2 Database Interface

The database interface for the both side of system built to show the data that are collected and check the functionality of the system by comparing the data stored in client side with the data stored at server side at specific moment of time.

The interface in figure (4.32) bellow show the Power Meter Reading processes which executing at the client side; it contains the Local Address(express the IP address) of the client machine and the consumed power for each process,

these two fields stored in client database at specific time determine in programming to be send to the server database.

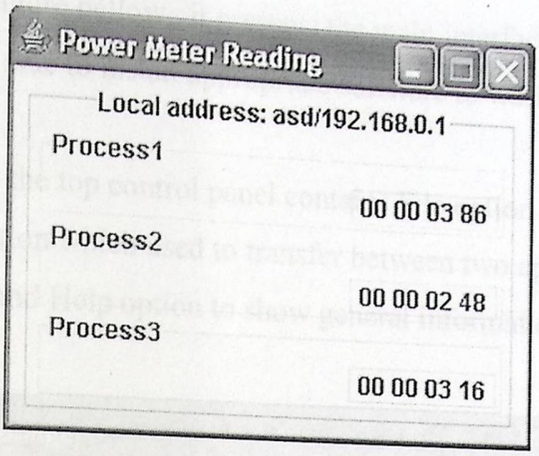


Figure 4.32 :Power Meter Reading Interface for client database.

The figure (4.33) shows the content of the database at the server side , it contains the IP address of the client machine ,the date that data collected from client ,the process values(P1 ,P2 ,P3) and the summation of process values

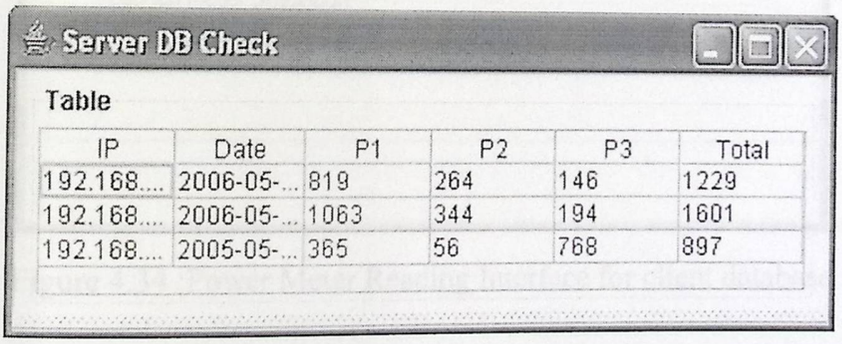


Figure 4.33 :Power Meter Reading Interface for client database

4.4.3 Client/server Interface

In this section ,we will present the general interfaces that appears and the error message that maybe at program setup time.

4.4.3.1 Client/server AMR Manager Interface

As shown in figure bellow , it presents the main interface(control panel) for client side and server side to install appropriate software to work the system as well.

the toolsbar at the top control panel contains File option which contain exit button , Configure option which used to transfer between two approaches (client/ server,mobile agent) and Help option to show general information about program.

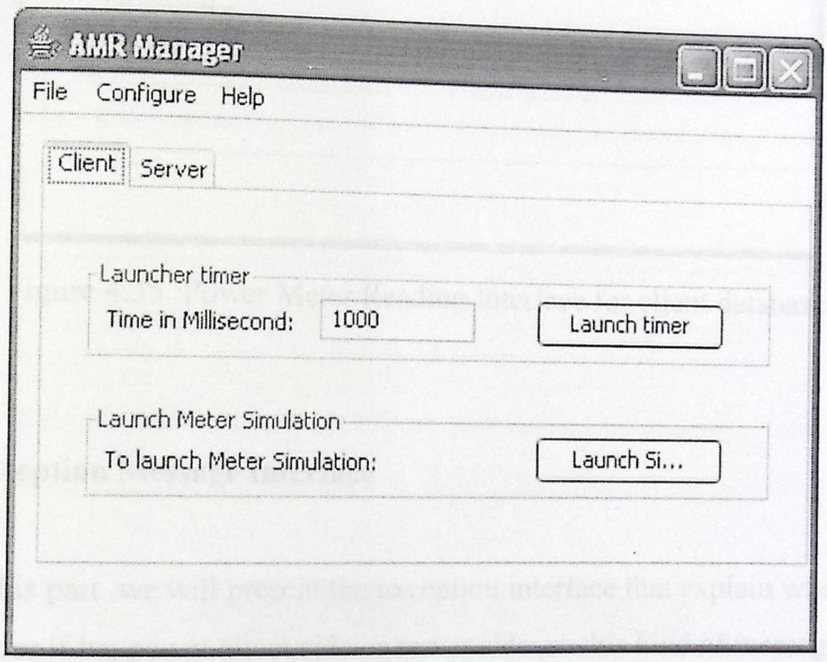


Figure 4.34 :Power Meter Reading Interface for client database

The field Launch time used to determine the time to send http message to server, and the field Launch simulation used to meter simulation at client side.

The figure bellow shows the server side options to install m the field DB simulation used to present the Database on the server side.

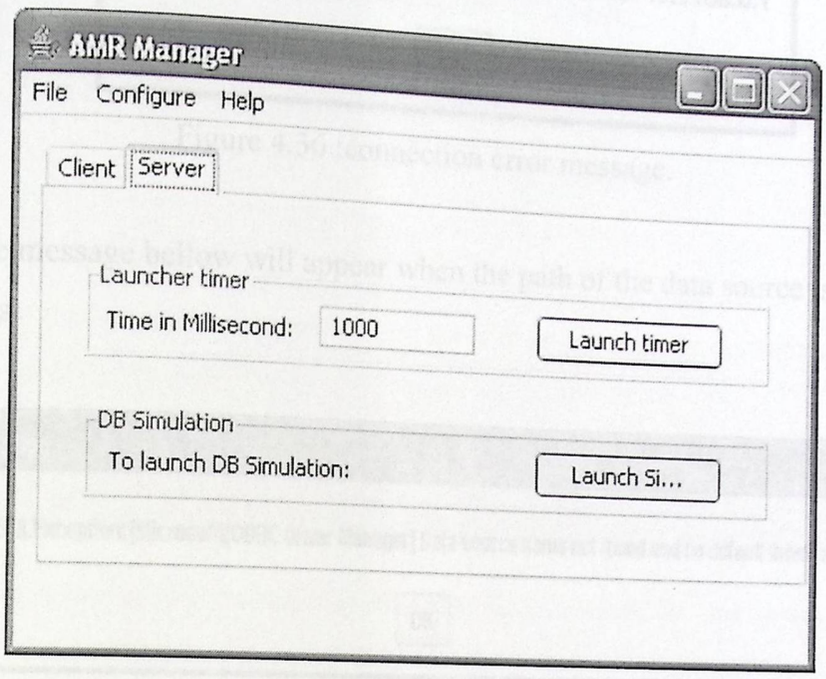


Figure 4.35 :Power Meter Reading Interface for client database

4.4.3.2 Exception Message Interface

In this part ,we will present the exception interface that explain where the error be occur if happen at client side or server side ,so,this kind of message help the user to know about error.

The figure bellow explain that there is no connection to server machine whose IP address is 192.168.0.1 .

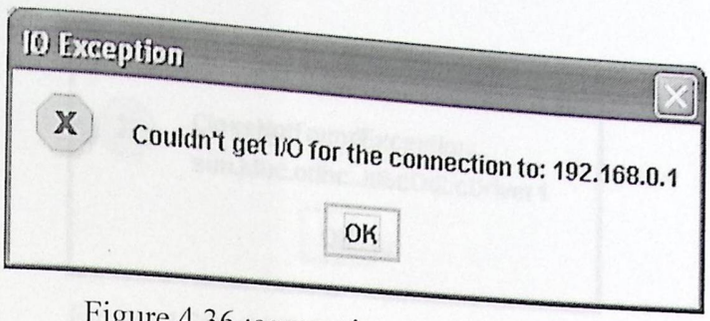


Figure 4.36 :connection error message.

The message bellow will appear when the path of the data source is not found in windows.

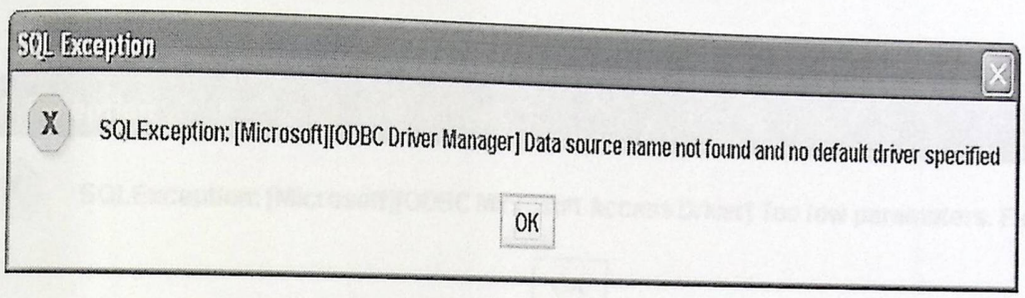


Figure 4.37 :error database path message.

The figure bellow explain that there is no driver in a machine suitable the database type , in our project , we use the ODBC driver to access database.

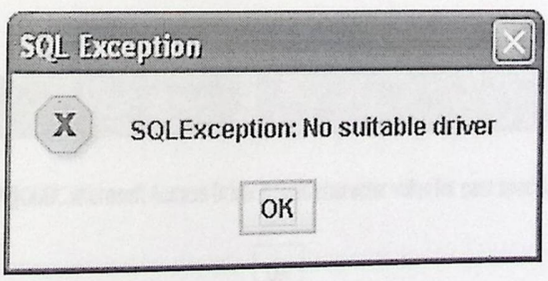


Figure 4.38 :driver error message.

The figure bellow explain that there is no suitable driver found on a machine , therefore , the database on a machine will be undefined .

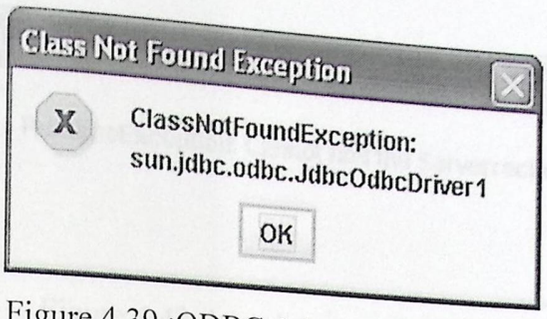


Figure 4.39 :ODBC driver error message.

The message bellow appears when there is an error in the construction of table ,name and fields type.

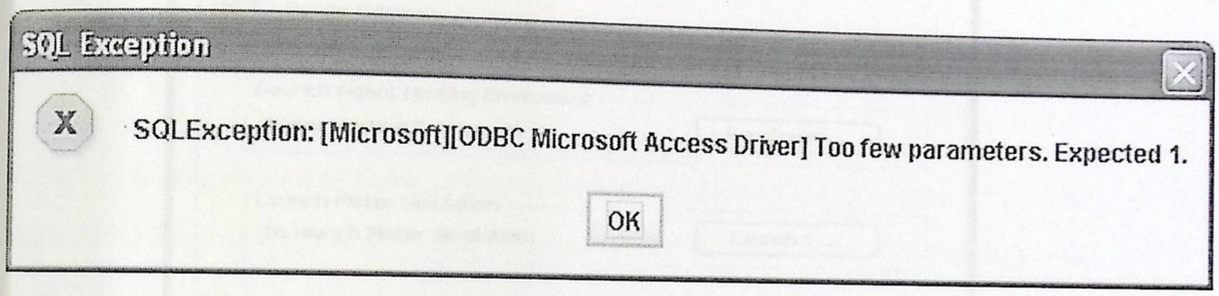


Figure 4.40 : Power Meter Reading Interface for client database

The error massege bellow will happen when there is no (PID) field in a table that used .

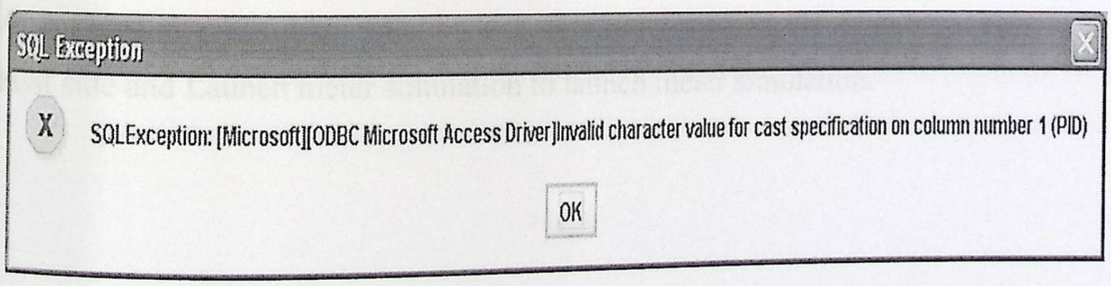


Figure 4.41 :Table field error message.

The error message bellow will appear if the launcher don't fid the program that to be launched at the machine.

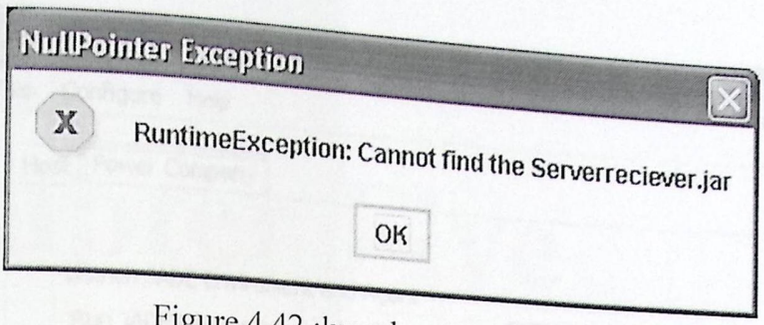


Figure 4.42 :luancher error message.

4.4.3.3 Mobile agent AMR system Manager Interface

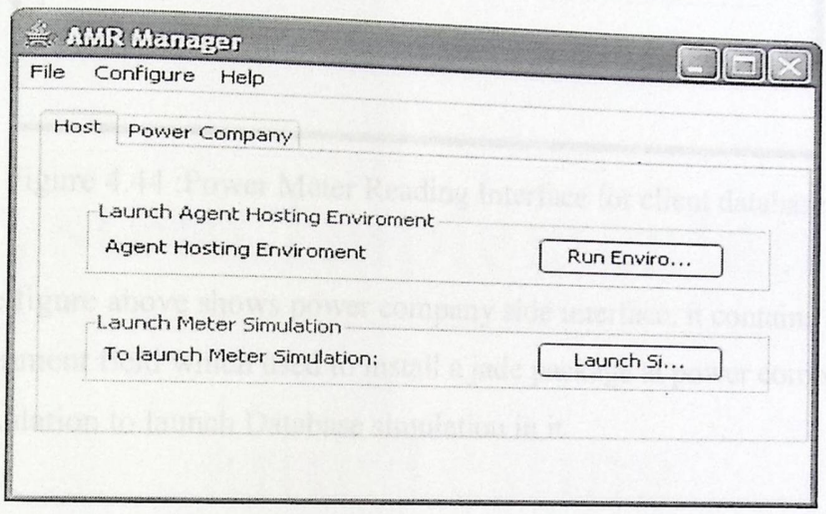


Figure 4.43 : MAMR Interface for host side

The figure above shows the second option in the figure option, it contains Agent hosting environment field which used to install a necessary jade package at host side and Launch meter simulation to launch meter simulation.

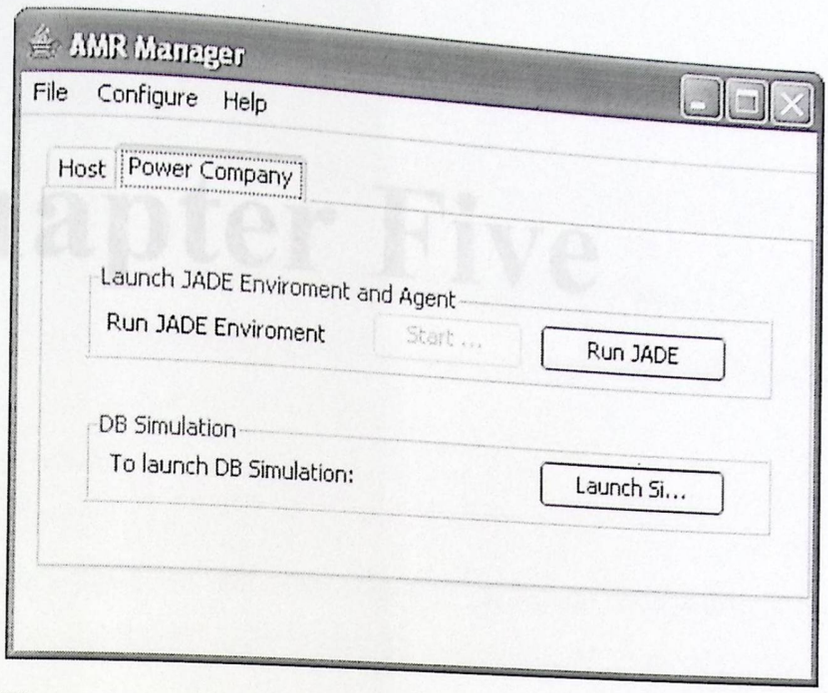


Figure 4.44 :Power Meter Reading Interface for client database

The figure above shows power company side interface, it contains Luanch jade environment field which used to install a jade package at power company side and DBsimulation to launch Database simulation in it.

- **Simple mobile agent fualt tolerance**

There is a proplem occure when the agent visit a hosting machine that are shut down ,but no error message will appear in this kind of problem, so the agent well visit the next hosting machine depending on the IP address exist at the table (the agent could not stop his task,and go to next move).

Chapter Five

System Implementation and Testing

Chapter Five

Implementation and Testing

5.1 Implementation

In this section we will discuss how to implement our system on machines for each phase independently and identify the coding and program of the system to work as well.

5.1.1 Development Environment

5.1.1.1 Hardware Environment

In order to complete the system implementation process successfully, the following hardware were used(we suppose it is exist):

1. Computer laboratory
2. Network connections among the computers.

5.1.1.2 Software Environment

The software that is used in the development process to execute the system are:

1. Java Runtime Environment (JRE) : This program is free-download software used in our project to execute the java code and should installed in each computers.

2. Jade Package : this package installed in each computer when implements the mobile agent phase, it fully-installed in "central" computer and abstracted-installed in other computer.The package include the mobile agent environment ,creating platform and the containers which the mobile agent needed it to work as well.

3. Java application to program the system code.

4. MS-access application to create a database tables that needed to store the meter reading data.

5.1.2 Development Process

Here ,we will discuss in detail how the system implement and configure on a machines to acheive its goals.

5.1.2.1 Phase One: AMR system using Client/Server technique

The development process of the AMR system by using client/server technique be as the following steps :

1. put the database of each client and server in known place for the JDBC-ODBC driver on a machine, and this step specified already in the CD software project that determine the database path on a machine.

2. activate the processe and main process for once on a client machine.
3. activate the launcherS,KKMultiserver programs on server machine.
4. activate launcherC program on a client machine.
5. after these steps the AMR system start its work.

5.1.2.2 Phase Two: AMR system using Mobile Agent approach

The development process of the AMR system by using Mobile Agent technique be as the following steps :

1. activate the Jade environment on a power company machine side to prepare a suitable platform for future connected host machine.
2. Activate the agent hosting software on a hosts machine which create a container on the power company side.
3. setup a host database on its machine.
4. setup a simulation power meter reader on a hosts to give a consumed power for each them.
5. Activate the agent : there is two method to activate the agent;first by entering a command to activate it (use Batch file contain a comand),or by adding the agent and datat code to the jade environmevt ,here the agent will activate when the jade environment be active.

5.1.3 The Bandwidth Comparison for AMR system approaches.

To measure the perfomance factors for each approaches in the AMR system, the network should connect a large number of client(host) machines to notice the different for that factors at server side and connect the network to the WAN network as in the real world, so, the router needed for measuring .

In our system implementation, we use a LAN network and use a Task Manager program that exist in Windows XP to measure the performance factors of a network because the computer center in PPU was busy and have no permission to change its network as there in our network system , so, to solve this problem partially, we use a LAB. Computer that contain 11 machine and one server and connected them as a LAN without router.

5.1.3.1 The Client/Server performance

The figure bellow results when the bandwidth of network equal to 10Mbps, uses 11 client machine and one server; the figure shows that the network will be busy all the time, and this make more activity with little information.

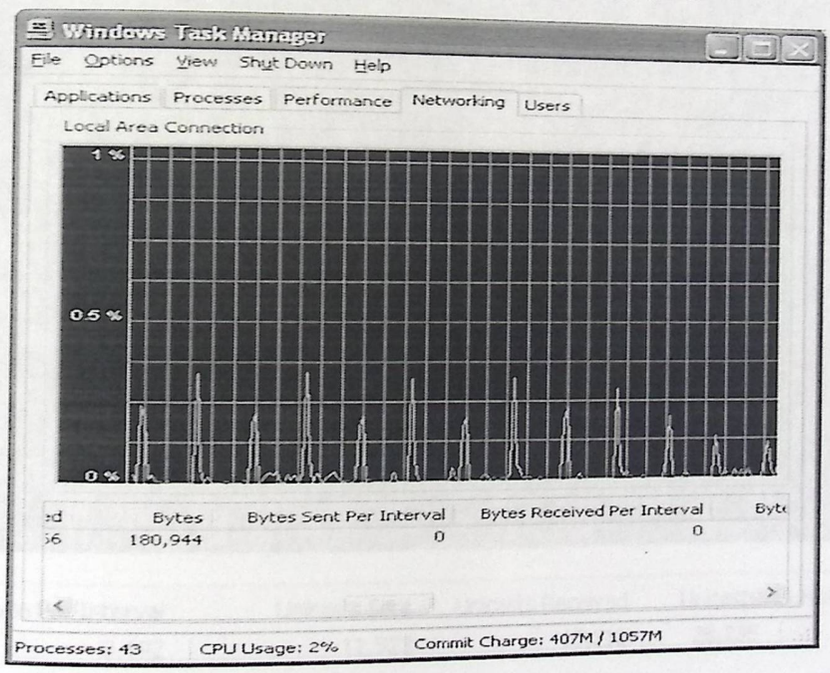


Figure 5.1: Client/Server performance.

5.1.3.2 The Mobile Agent performance

The figure bellow shows that the network at this approach will be at idle state until yhe agent move from one machine to another.

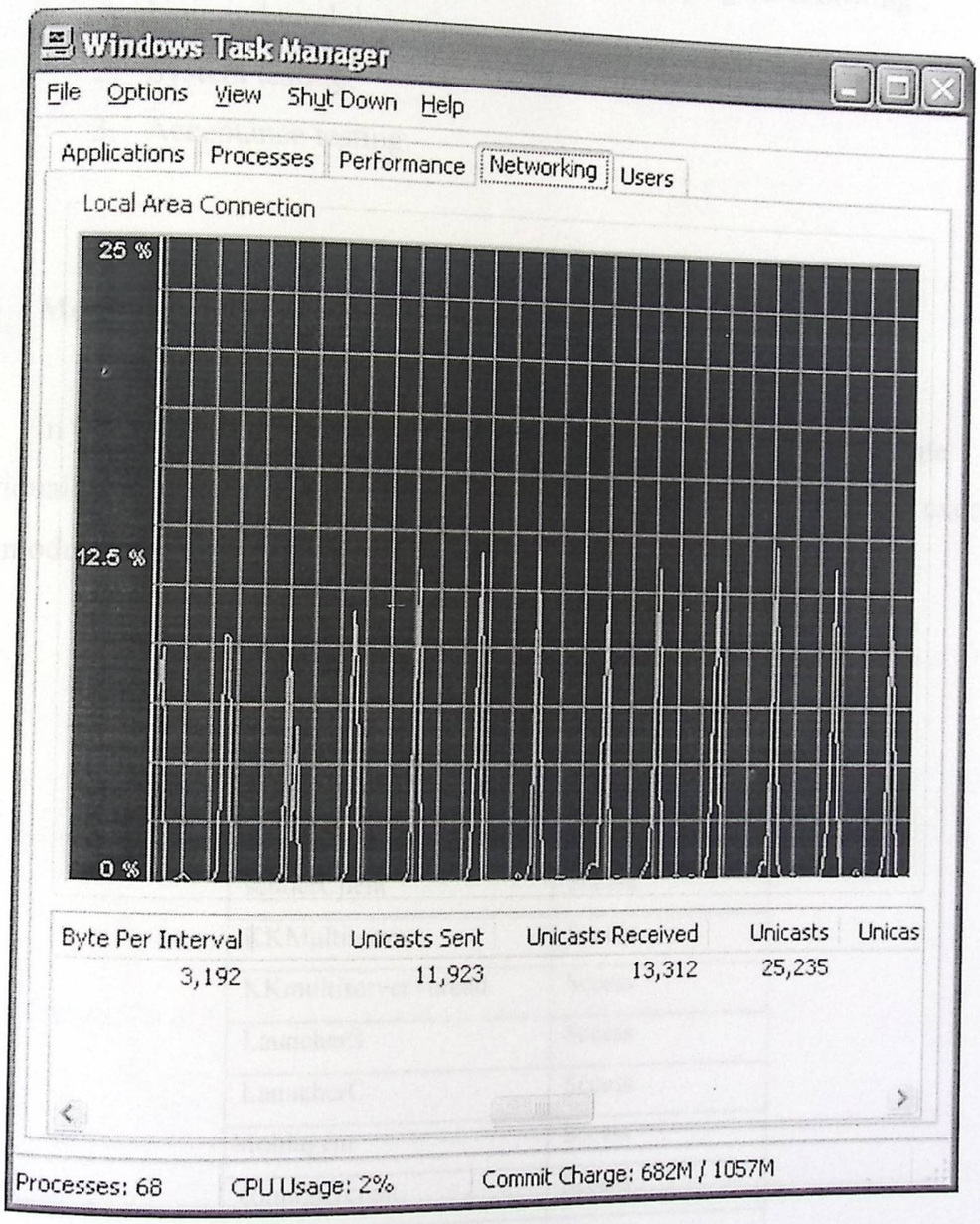


Figure 5.2: Mobile Agent performance.

5.2 Testing

This section will discuss the most important step after coding and implementation which is system testing. System testing is the process of ensuring that the system performs as expected and satisfies the main objectives and requirements. All functions must be tested individually and integrated with each others; so in this section we will test the system by applying the following :

1. Unit and module testing.
2. System testing.
3. Acceptance testing.

5.2.1 Module & Unit Testing

In this section units and modules are tested by executing each module individually and test the actual output message that appear while setup and execute each module on a host or company machines.

Table 5.1 : Module Testing

Module	Result
Processimpl	Sccess
mainP	Sccess
senderClient	Sccess
KKMultiserver	Sccess
KKmultiserverThread	Sccess
LauncherS	Sccess
LauncherC	Sccess
Mobiagent	Sccess
containercreat	Sccess
datat	Sccess

5.2.2 System testing

In this section, the AMR system using two techniques will be tested as a whole, the testing technique be by view the database to check if the meter reading data reach to the power company machine and entered correctly.

Table 5.2 : System Testing

AMR system approaches	Result Testing
Client/Server approach	Succes
Mobile Agent approach	Success

5.2.3 Phase One (AMR system using client/server approach)

The way to test if the system work correctly be in check the database of the server.

After install the system on the client and server machines, we will check the database of the server to be empty at first, then, when the system execute and no error message appear, we will check the database of the clients and the server visually to sure that meter reading reach and enter in database fields correctly and it is the same of meter reading on client database at that moment.

5.2.4 Phase Two(AMR system using Mobile Agent approach)

This technique tested visually as the client/server technique, we tested it by show the database fields before and after executing the system, if the meter reading

data filled correctly in power company machine so the system work correct, but that not enough in this technique.

Here, we will be sure that the agent moved serially(depend on the IP address table) on the network and reach them , this can be tested by the Jade environment on the power company machine where all containers to all machine are there, and we could notice agent clearly how it is move along the network as shown bellow :

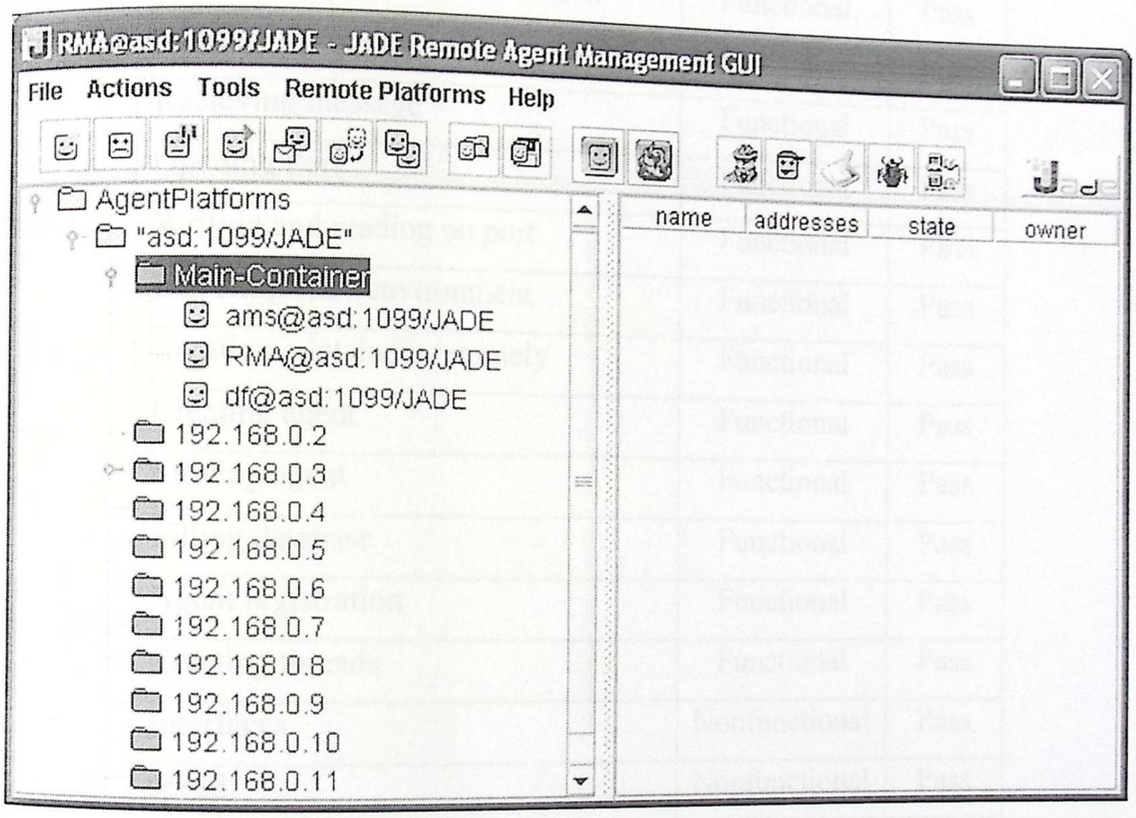


Figure 5.3 : Moving agent in the container at jade environment

5.2.5 Acceptance Testing

The system was tested against requirement functional and nonfunctional specifications; the following table shows the requirement, its type and the testing results:

Table 5.3 : Module functionality Testing

Requirement	Type	Result
Running simulation meter (Process)	Functional	Pass
Executing processes	Functional	Pass
Launching client database	Functional	Pass
Launching server database	Functional	Pass
Openning stream and connection	Functional	Pass
Sending messege	Functional	Pass
Recieving messege	Functional	Pass
Opening Port	Functional	Pass
Writing and reading on port	Functional	Pass
Launchig Jade environment	Functional	Pass
Creating containers remotely	Functional	Pass
Creating agent	Functional	Pass
Moving agent	Functional	Pass
Filling database	Functional	Pass
Agent registration	Functional	Pass
Creating threads	Functional	Pass
Interfaces	Nonfunctional	Pass
speed	Nonfunctional	Pass
Error handling	Nonfunctional	Pass
usability	Nonfunctional	Pass

Chapter Six

Conclusion and Future Work

Chapter Six

Conclusions and future works

6.1 Introduction

This system aimed to grant a better chance for power company in serving their cost,time and error prone that result from collecting client's power meter data; by comparing the performance factor for different data-collecting method and achieving fitting between power company and clients machines.

But, there are some considerations that should be taken in mind that caused the system not to be as perfect as required. Some of these considerations are:

- This system has been experienced with a limited time period and limited resources for the development ,implementation phases and book resources on the subject of system..
- This system depends mainly on the team and what they think of such a system.
- The system need to be joined with many parts of its environment and this need to deal with a database, ports and containers.

6.2 Conclusions

Through the system development process the work team has concluded the following:

- The performance of mobile agent technique on a network is better than client/server technique in collecting data from power meter device that holds in client's side according its speed ,throuhput and bandwidth.

- The system consider as automated system, so there is no direct dealing with company and consumer , andso , no need for the consumer to wait the consuming power bill or power employee to come anymore.

6.3 Recommendations and Future works

The work team recommends the following actions as a future work for the system :

- Development the database in both sides of system (consumer, company) and connected it to the web page to view bills of client according specific criteria insure a security.

- Development the static IP address table for the agent to be a dynamic IP tables at programming side.

- Development the security for code and data in both phases (client/server , mobile agent).

- Development the agent to be intelligence in choose the best path to visit clients machine power meter device.

- Development the agent to deal with different platform instead with one platformm and deal with fault tolerance that may happen through its moving from one machine to another.
- Development a simulation process to be connected in real as a hardware.

References

REFERENCES

Books

M. Yeary, B. Swan, J. Sweeney, C. Gulp,An, Internet Based Power Measurement Technique, IEEE Instrumentation and Measurement Technology Conference, Budapest, Hungary, May 21-23, 2001.

Gray R., A flexible and secure mobile-agent system. Department of Computer Science. Dartmouth College. 1996.

B. Fabio, C. Giovanni, T. Tiziana , JADE PROGRAMMER'S GUIDE, 10 July-2004.

L. Bettini, Klava. , A Java framework for mobile code, Germany, 2003.

Peine H. and Stolpman T., Meter reading system, Department of Computer Science, University of Kaiserslautern. 1997.

Internet

http://agent.cs.dartmouth.edu

http://www.omg.org/technology/documents/formal/mobile_agent_facility.htm

http://www.recursionsw.com/mobile_agents.htm

http://www.gurnee.il.us/info_sys/is_amrs.html

http://www.sioux-city.org/customer_service/automatic_meter.asp

Appendix

APPENDIX- (A)

senderClient.class

```
//needed senderClient
import java.io.*;
import java.net.*;
import java.sql.*;
import javax.swing.*;
import java.awt.event.*;

public class senderClient extends JFrame{
  public static void main(String[] args) throws IOException {

    //create jframe to make the program is visible
    senderClient now =new senderClient();
    now.addWindowListener(new WindowAdapter(){
      public void windowClosing(WindowEvent e){System.exit(0);}});
    now.setTitle ("senderClient");
    now.setSize (200,0);
    now.setLocation (100,100);
    now.setResizable (false);
    now.setVisible (true);

    //IO and network configuration
    Socket socket = null;
    PrintWriter out = null;
    BufferedReader in = null;

    //Specify the name of DataSource
    String url = "jdbc:odbc:clientjdk";

    Connection con;
    String createString;

    //Server IP address
    String Hostname="192.168.0.1";
    String info[]=new String[8];

    //create socket and conect it to server socket and open streams
    try {

      socket = new Socket(InetAddress.getByName(Hostname), 4444);
      out = new PrintWriter(socket.getOutputStream(), true);
```

```
        in = new BufferedReader(
        new InputStreamReader(socket.getInputStream()));
    } catch (UnknownHostException e) {

        //error if there is no server
        JOptionPane.showMessageDialog(null,"Don't know about host: "
                                +Hostname,
                                "Unknown Host Exception",
                                JOptionPane.ERROR_MESSAGE);

        System.exit(1);
    } catch (IOException e) {

        //error if there is no connection to server
        JOptionPane.showMessageDialog(null,
                                "Couldn't get I/O for the "+
                                "connection to: "+Hostname,
                                "IO Exception",
                                JOptionPane.ERROR_MESSAGE);

        System.exit(1);
    }

    //determine the specific Query
    createString = "select * from PMPT ORDER BY PID ASC";
    Statement stmt;

    try {
        Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
    } catch(java.lang.ClassNotFoundException e) {

        //error if ther is no jdbc:odbc bridge driver
        JOptionPane.showMessageDialog(null,"ClassNotFoundException: \n"
                                +e.getMessage(),
                                "Class Not Found Exception",
                                JOptionPane.ERROR_MESSAGE);
    }

    //connect to database and read the neaded data
    try {
        con = DriverManager.getConnection(url);
        stmt = con.createStatement();
        ResultSet rs =stmt.executeQuery(createString);
```

```
            int i=2;
            while (rs.next()) {
               info[i]=rs.getString(1);
               i++;
               info[i]=rs.getString(2);
               i++;
            }

            //close the connection to database
            stmt.close();
            con.close();

            //add the local IP and Local Hostname
            info[0]=InetAddress.getLocalHost().getHostAddress();
            info[1]=InetAddress.getLocalHost().getHostName();
         } catch(SQLException ex) {

            //error if any SQL error happen
            JOptionPane.showMessageDialog(null,"SQLException: " +
                                 ex.getMessage(),
                                 "SQL Exception",
                                 JOptionPane.ERROR_MESSAGE);
         }
         //send the message as string
         String fromServer;
         String fromUser;
         if ((fromServer = in.readLine()) != null) {
            fromUser = info[2];
            while(fromServer.compareTo("ok\n")==0)fromServer = in.readLine();
            if (fromUser != null) {
               for(int k=0;k<8;k++)
                  out.println(info[k]);
            }
         }
         //close the streams and the socket
         out.close();
         in.close();
         socket.close();
         //exit
         System.exit(1);
      }
   }
```

launcherC.class

```
//needed library
import java.util.Timer;
import java.util.TimerTask;
import java.util.Calendar;
import java.lang.Process;
import java.lang.Runtime;
import java.text.DateFormat;
import java.util.*;
import java.io.*;
import javax.swing.*;
import java.awt.event.*;
import java.net.URL;

public class launcherC extends JFrame{

  static Timer timer;
  static int month;
  static Calendar calendar;
  static Date time,neww=new Date();
  static long tm;
  static boolean startup=false;

 static class RemindTask extends TimerTask {
   public void run() {
     String[] x={String.valueOf (tm)};

     //terminate the timer
     timer.cancel();
     try{
       startup=true;

       //to launch specific program (excute specific command
       Runtime runtime = Runtime.getRuntime();
       runtime.gc ();
       Process process = runtime.exec(
            "javaw.exe -jar "+this.getClass().getResource(
                 "Clientsender.jar").getPath().substring(6));
       System.out.println ("sho");


    }catch(Exception ex){
```

```
            //error if it is wrong command
            JOptionPane.showMessageDialog(null,"RuntimeException: "
                                +"Cannot find the"+
                                " Clientsender.jar",
                                "NullPointer Exception",
                                JOptionPane.ERROR_MESSAGE);
        }
        runn3();

        //do timer again
        launcherC.main(x);
    }

    public void runn3(){

    }
}

public static void main(String args[]) {

  //startup in the first==false
  if(!startup){

     //create jframe to make the program is visible
     launcherC now=new launcherC();
     now.addWindowListener(new WindowAdapter(){
        public void windowClosing(WindowEvent e){System.exit(0);}});
     now.setTitle ("Launcher for Client");
     now.setSize (250,0);
     now.setLocation (100,100);
     now.setResizable (false);
     now.setVisible (true);
  }

  //get and set the calendar to specific date
  calendar = Calendar.getInstance();
  //calendar.clear ();
  //calendar.set(Calendar.MONTH,28);
  //calendar.set(Calendar.HOUR, 2);
  //calendar.set(Calendar.MINUTE, 1);
  calendar.set(Calendar.SECOND,59 );
  calendar.set(Calendar.MILLISECOND,999);
```

```
        time = calendar.getTime();
        if(args[0]==null)
          tm=time.getTime ();
        else
          tm=Long.parseLong (args[0]);

        //create timer and give it the time and timertask
        timer = new Timer();
        timer.schedule(new RemindTask(),tm );

    }
}
```

launcherS.class

```
//needed library
import java.util.Timer;
import java.util.TimerTask;
import java.util.Calendar;
import java.lang.Process;
import java.lang.Runtime;
import java.text.DateFormat;
import java.util.*;
import java.io.*;
import javax.swing.*;
import java.awt.event.*;

public class launcherS extends JFrame {
   static Timer timer;
   static int month;
   static Calendar calendar;
   static Date time,neww=new Date();
   static long tm;
   static boolean startup=false;

   static class RemindTask extends TimerTask {
      public void run() {
         String[] x={String.valueOf (tm)};

         //terminate the timer
         timer.cancel();
```

```
        runn3();

        //do timer again
        launcherS.main(x);
    }

    public void runn3(){

        try{
            startup=true;

            //to launch specific program (excute specific command
            Runtime runtime = Runtime.getRuntime();
            Process process = runtime.exec( "javaw.exe -jar "+
                                this.getClass().getResource(
                              "Serverreciever.jar").
                                getPath().substring(1));

        }
        catch(Exception ex){

            //error if it is wrong command
            JOptionPane.showMessageDialog(null,"RuntimeException: " +
                                "Cannot find the "+
                                "Serverreciever.jar",
                                "NullPointer Exception",
                                JOptionPane.ERROR_MESSAGE);
        }

    }
}

public static void main(String args[]) {

    //startup in the first==false
    if(!startup){

        //create jframe to make the program is visible
        launcherS now=new launcherS();


        now.addWindowListener(new WindowAdapter(){
            public void windowClosing(WindowEvent e){System.exit(0);}});
        now.setTitle ("Launcher for Server");
```

```
        now.setSize (250,0);
        now.setLocation (100,100);
        now.setResizable (false);
        now.setVisible (true);
      }

      //get and set the calendar to specific date
      calendar = Calendar.getInstance();
      //calendar.set(Calendar.MONTH,28);
      //calendar.set(Calendar.HOUR, 2);
      //calendar.set(Calendar.MINUTE, month);
      calendar.set(Calendar.SECOND,59 );
      calendar.set(Calendar.MILLISECOND,999);


      time = calendar.getTime();
      if(args[0]==null)
        tm=time.getTime ();
      else
        tm=Long.parseLong (args[0]);

      //create timer and give it the time and timertask
      timer = new Timer();
      timer.schedule(new RemindTask(), tm);

    }
}
```

KKMultiServerThread.class

```
//needed library
import java.net.*;
import java.io.*;
import java.sql.*;
import java.util.*;
import javax.swing.JOptionPane;

public class KKMultiServerThread extends Thread {

   private Socket socket = null;
   static String info[]=new String[8];
   PreparedStatement updateV;
```

```
//specify the thread name
public KKMultiServerThread(Socket socket) {
   super("KKMultiServerThread");
   this.socket = socket;
}

public void run() {

   //open streams to transmit and recieve
   try {
      PrintWriter out = new PrintWriter(socket.getOutputStream(), true);
      BufferedReader in = new BufferedReader(
      new InputStreamReader(socket.getInputStream()));
      String inputLine, outputLine;
      outputLine = "ok";
      out.println(outputLine);

      //read the recieved message
      int i=0;
      while ((inputLine = in.readLine()) != null) {
         if(inputLine==null)break;
         if(i==8)break;
         info[i]=inputLine.trim();
         i++;

      }

      //close connection Stream and socket
      out.close();
      in.close();
      socket.close();

      DBMS(info);

   } catch (IOException e) {

      //error if happened in connection
      JOptionPane.showMessageDialog(null,e.getMessage (),
                              "IO Exception",
                              JOptionPane.ERROR_MESSAGE);
      e.printStackTrace();
}
```

```
}
public void DBMS(String[] infor){

  //Specify the name of DataSource
  String url = "jdbc:odbc:serverjdk";
  Connection con;

  try {
    Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
  }catch(java.lang.ClassNotFoundException e) {

    //error if ther is no jdbc:odbc bridge driver
    JOptionPane.showMessageDialog(null,"ClassNotFoundException: \n"
                        +e.getMessage (),
                        "IO Exception",
                        JOptionPane.ERROR_MESSAGE);
    System.err.print("ClassNotFoundException: ");
    System.err.println(e.getMessage());
  }
  try {

    con = DriverManager.getConnection(url);

    //determine the specific prepareStatement
    updateV = con.prepareStatement(
        "INSERT INTO PMPR VALUES (?,?,?,?,?,?)");

    //insert the data to database
    updateV.setString(1,infor[0]);
    updateV.setDate(2,new java.sql.Date(
        Calendar.getInstance().getTimeInMillis()));
    updateV.setString(3,infor[3]);
    updateV.setString(4,infor[5]);
    updateV.setString(5,infor[7]);
    updateV.setString(6,
            String.valueOf (Double.parseDouble (infor[3])
                        +Double.parseDouble (infor[5])
                        +Double.parseDouble (infor[7])));

    updateV.executeUpdate();

    //close the connection to database
    con.close();
```

```
    } catch(SQLException ex) {

      //error if any SQL error happen
      JOptionPane.showMessageDialog(null,"SQLException: \n"
                          + ex.getMessage(),
                          "IO Exception",
                          JOptionPane.ERROR_MESSAGE);

      System.err.println("SQLException: " + ex.getMessage());
    }


  }
}
```

KKMultiServer.class

```
//needed library
import java.net.*;
import java.io.*;
import javax.swing.*;
import java.awt.event.*;

public class KKMultiServer extends JFrame{

  static boolean listening = true;
  public static void main(String[] args) throws IOException {

    //create jframe to make the program is visible
    KKMultiServer now =new KKMultiServer();

    now.addWindowListener(new WindowAdapter(){
      public void windowClosing(WindowEvent e){System.exit(0);}});
    now.setTitle ("KKMultiServer");
    now.setSize (200,0);
    now.setLocation (100,100);
    now.setResizable (false);
    now.setVisible (true);

    ServerSocket serverSocket = null;

    //create server socket
```

```
    try {
       serverSocket = new ServerSocket(4444);

    } catch (IOException e) {

       //error if couldn't create the socket
       JOptionPane.showMessageDialog(null,
                                "Could not listen on port: 4444.",
                                "IO Exception",
                                JOptionPane.ERROR_MESSAGE);
       System.exit(-1);
    }

    //create threads to get more than one client
    while (listening)
       new KKMultiServerThread(serverSocket.accept()).start();

    //close the socket
    serverSocket.close();
  }
}
```

Processimpl.class

```
//needed library
import java.util.Timer;
import java.util.TimerTask;

public class Processimpl {
   private  String PID;
   private Timer timer;
   private int sec;
   private  double read=0;

   //constructor to initiate the load consuming and PID
   public Processimpl(int seconds,String PID){

      sec=seconds;
      this.PID=PID;
      Processwork();

   }
```

```
public void Processwork(){

   //create timer and give it the time and timertask
   timer = new Timer();
   timer.schedule(new RemindTask(), sec*1000);

}

class RemindTask extends TimerTask {

   public void run() {
      if(read<99999999)
         read+=1;
      else read=0;
      timer.cancel(); //Terminate the timer thread
      Processwork();
    }

  }

  //to get the process ID
  public  String getPID(){
     return PID;
  }

  //to get the process read
  public  double getread(){
     return read;
  }
}
```

mainP.class

```
import java.util.Timer;
import java.util.TimerTask;
import javax.swing.*;
import javax.swing.border.*;
import java.awt.event.*;
import java.util.*;
import java.text.*;
import java.awt.*;
import java.sql.*;
import java.net.*;
```

```
public class mainP {

   Timer timer;
   static JLabel label1,label2,label3;
   static JPanel sc1,sc2,sc3;
   static String s="",s1,s2,s3;
   static double no1,no2,no3,key=0;
   static double p1,p2,p3;
   PreparedStatement updateV;
   Processimpl Process1,Process2,Process3;
   Connection con,con1;
   Statement stmt;

   public void mainp() {

      //create timer and give it the time and timertask
      timer = new Timer();

      //update every 100 miliseconds
      timer.schedule(new RemindTask(), 0,1*1000);

   }

   class RemindTask extends TimerTask {

      public void run() {

         //get the reading of processes and ID's
         p1=Process1.getread();
         p2=Process2.getread();
         p3=Process3.getread();
         s1=Process1.getPID();
         s2=Process2.getPID();
         s3=Process3.getPID();

         DBMS();


         sc1.setBorder(BorderFactory.createTitledBorder(s1));
         sc2.setBorder(BorderFactory.createTitledBorder(s2));
         sc3.setBorder(BorderFactory.createTitledBorder(s3));

         //show the values in GUI in custom formate
```

```
        s=customFormat("000000,00", p1);
        label1.setText(s);

        s=customFormat("000000,00", p2);
        label2.setText(s);

        s=customFormat("000000,00", p3);
        label3.setText(s);


    }
}

public void DBMS(){

    //Specify the name of DataSource
    String url = "jdbc:odbc:clientjdk";

    String s,Query,n;

    //determine the specific Query
    Query= "select * from PMPT";

    try {
        Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
    }catch(java.lang.ClassNotFoundException e) {

        //error if ther is no jdbc:odbc bridge driver
        JOptionPane.showMessageDialog(null,"ClassNotFoundException: \n"
                                      +e.getMessage (),
                                      "IO Exception",
                                      JOptionPane.ERROR_MESSAGE);
        System.err.print("ClassNotFoundException: ");
        System.err.println(e.getMessage());
    }

    //connect to database and read the neaded data
    try {
        con = DriverManager.getConnection(url);
        con1 = DriverManager.getConnection(url);
        stmt = con.createStatement();

        //determine the specific prepareStatement
        updateV = con1.prepareStatement("UPDATE PMPT SET Read"+
```

```
                    " = ? WHERE PID like ?");
ResultSet rs =stmt.executeQuery(Query);

//read the stored data from database (old values)
if(key==0){
   rs.next();
   s = rs.getString(1);
   n = rs.getString(2);
   no1=Double.parseDouble(n);

   System.out.println(s + "   " + n);

   rs.next();
   s = rs.getString(1);
   n = rs.getString(2);
   no2=Double.parseDouble(n);

   System.out.println(s + "   " + n);

   rs.next();
   s = rs.getString(1);
   n = rs.getString(2);
   no3=Double.parseDouble(n);

   System.out.println(s + "   " + n);
   key=1;
}

p2+=no1;
p1+=no2;
p3+=no3;
if(p1>99999999)p1=0;
if(p2>99999999)p2=0;
if(p3>99999999)p3=0;

//update the database data
updateV.setString(1,String.valueOf((int)p1));
updateV.setString(2,s1);
updateV.executeUpdate();

updateV.setString(1,String.valueOf((int)p2));
updateV.setString(2,s2);
updateV.executeUpdate();
```

```
        updateV.setString(1,String.valueOf((int)p3));
        updateV.setString(2,s3);
        updateV.executeUpdate();

        //close the connection to database
        stmt.close();
        con.close();
        con1.close ();

      } catch(SQLException ex) {

        //error if any SQL error happen
        JOptionPane.showMessageDialog(null,"SQLException: \n"
                                + ex.getMessage(),
                                "IO Exception",
                                JOptionPane.ERROR_MESSAGE);
        System.err.println("SQLException: " + ex.getMessage());
      }


}

//method to manipulate the formate of reading
static public String customFormat(String pattern, double value ) {

    DecimalFormatSymbols decimalFormate=
       new DecimalFormatSymbols(new Locale("en", "US"));
    decimalFormate.setGroupingSeparator(' ');
    DecimalFormat myFormatter = new DecimalFormat(pattern,decimalFormate);
    String output = myFormatter.format(value);

    return output;
}


public static void main(String args[]) {

    //create GUI of simulation for power meter reading
    JFrame DialogFrame;
    String localad="";
    label1=new JLabel();
    label2=new JLabel();
    label3=new JLabel();
```

```
mainP mainpr=new mainP();

JPanel zx=new JPanel(new GridLayout(0,1));
JPanel zz=new JPanel();
JPanel content=new JPanel(new BorderLayout());
sc1=new JPanel(new BorderLayout());
sc2=new JPanel(new BorderLayout());
sc3=new JPanel(new BorderLayout());

try{
   localad=InetAddress.getLocalHost().toString();
}catch(UnknownHostException e) {}
DialogFrame=new JFrame("Power Meter Reading");
DialogFrame.addWindowListener(new WindowAdapter(){
   public void windowClosing(WindowEvent e){

      System.exit(0);
   }
});
sc1.add(label1,BorderLayout.EAST);
sc2.add(label2,BorderLayout.EAST);
sc3.add(label3,BorderLayout.EAST);

label1.setBorder(BorderFactory.createTitledBorder(""));
label2.setBorder(BorderFactory.createTitledBorder(""));
label3.setBorder(BorderFactory.createTitledBorder(""));

zx.add(sc1);
zx.add(sc2);
zx.add(sc3);
zx.setBorder(BorderFactory.createTitledBorder(
   BorderFactory.createEtchedBorder(),"Local address: "
   +localad,TitledBorder.CENTER,TitledBorder.DEFAULT_POSITION ));

content.add(zx,BorderLayout.CENTER);
content.setBorder(BorderFactory.createTitledBorder(""));

DialogFrame.getContentPane().add(content,BorderLayout.CENTER);
DialogFrame.setSize(250+localad.length(),200);
DialogFrame.setLocation(200,100);
DialogFrame.setResizable(false);
DialogFrame.setVisible(true);

//initiate 3 instance of processes
```

```
        mainpr.Process1=new Processimpl(1,"Process1");
        mainpr.Process2=new Processimpl(3,"Process2");
        mainpr.Process3=new Processimpl(5,"Process3");
        mainpr.mainp();


    }
}
```

SharedModelDemo.class

```
import javax.swing.*;
import javax.swing.event.*;
import javax.swing.table.*;
import java.util.*;
import java.awt.*;
import java.awt.event.*;
import javax.swing.border.*;
import java.sql.*;
import java.net.*;

public class SharedModelDemo extends JPanel {

    JTable table;
    static SharedDataModel dataModel;

    //constructor to initialize the GUI
    public SharedModelDemo() {
        super(new BorderLayout());


        DBreader();
        table = new JTable(dataModel);
        JScrollPane tablePane = new JScrollPane(table);

        JSplitPane splitPane = new JSplitPane(JSplitPane.VERTICAL_SPLIT);
        add(splitPane, BorderLayout.CENTER);

        JPanel topHalf = new JPanel();
        topHalf.setLayout(new BoxLayout(topHalf, BoxLayout.X_AXIS));

        JPanel tableContainer = new JPanel(new GridLayout(1,1));
        tableContainer.setBorder(BorderFactory.createTitledBorder("Table"));
```

```
    tableContainer.add(tablePane);
    tablePane.setPreferredSize(new Dimension(300, 100));

    topHalf.setBorder(BorderFactory.createEmptyBorder(5,5,0,5));
    topHalf.add(tableContainer);

    topHalf.setMinimumSize(new Dimension(400, 50));
    topHalf.setPreferredSize(new Dimension(400, 110));
    splitPane.add(topHalf);

    JPanel bottomHalf = new JPanel(new BorderLayout());

    bottomHalf.setPreferredSize(new Dimension(0, 0));
    splitPane.add(bottomHalf);


}

public static void main(String[] args) {

    //create GUI for Server DB Check
    JFrame frame = new JFrame("Server DB Check");
    frame.addWindowListener(new WindowAdapter() {
        public void windowClosing(WindowEvent e) {
            System.exit(0);
        }
    });

    frame.setContentPane(new SharedModelDemo());

    frame.pack();
    frame.setVisible(true);
}

public void DBreader(){

    //Specify the name of DataSource
    String url = "jdbc:odbc:serverjdk";
    Connection con;
    String Query;

    //determine the specific Query
    Query= "select * from PMPR";
    Statement stmt;
```

```
int i;
try {
   Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
}catch(java.lang.ClassNotFoundException e) {

   //error if ther is no jdbc:odbc bridge driver
   JOptionPane.showMessageDialog(null,"ClassNotFoundException: \n"
                        +e.getMessage(),
                        "Class Not Found Exception",
                        JOptionPane.ERROR_MESSAGE);

   System.err.print("ClassNotFoundException: ");
   System.err.println(e.getMessage());
}

//connect to database and read the neaded data
try {
   con = DriverManager.getConnection(url);
   stmt = con.createStatement();
   ResultSet rs =stmt.executeQuery(Query);

   //get the name of columns for GUI purpose
   String[] columnNames = { rs.getMetaData().getColumnLabel(1),
                        rs.getMetaData().getColumnLabel(2),
                        rs.getMetaData().getColumnLabel(3),
                        rs.getMetaData().getColumnLabel(4),
                        rs.getMetaData().getColumnLabel(5),
                        rs.getMetaData().getColumnLabel(6) };
   dataModel = new SharedDataModel(columnNames);


   //read the data from server database
   while (rs.next()) {
      String[] oneData= new String[6];
      i=0;
      while (i<6){
         if(i!=1)
            oneData[i] = rs.getString(i+1);
         else oneData[i] = rs.getDate(i+1).toString();
         i++;
      }
      dataModel.addElement(oneData);

   }
```

```
        //close the connection to database
        stmt.close();
        con.close();

    } catch(SQLException ex) {

        //error if any SQL error happen
        JOptionPane.showMessageDialog(null,"SQLException: "
                                + ex.getMessage(),
                                "SQL Exception",
                                JOptionPane.ERROR_MESSAGE);
        System.err.println("SQLException: " + ex.getMessage());
    }

}


//class for create table for display the database values
class SharedDataModel extends DefaultListModel
        implements TableModel {

    public String[] columnNames;

    public SharedDataModel(String[] columnNames) {
        super();
        this.columnNames = columnNames;

    }

    public void rowChanged(int row) {
        fireContentsChanged(this, row, row);
    }

    //TableModel implementation
    private TableModel tableModel = new AbstractTableModel() {

        //get the name of column
        public String getColumnName(int column) {
            return columnNames[column];
        }

        //get the number of rows (impl)
```

```
  public int getRowCount() {
     return size();
  }

  //get the number of column (impl)
  public int getColumnCount() {
     return columnNames.length;

  }

  //to get the specific value from table (impl)
  public Object getValueAt(int row, int column) {
     String[] rowData = (String [])elementAt(row);
     return rowData[column];
  }

  //to check the cell editable (impl)
  public boolean isCellEditable(int row, int column) {
     return true;
  }

  //to update the value of cpecific cell in table (impl)
  public void setValueAt(Object value, int row, int column) {
     String newValue = (String)value;
     String[] rowData = (String [])elementAt(row);

     rowData[column] = newValue;
     fireTableCellUpdated(row, column); //table event
     rowChanged(row);                   //list event
   }
};

//Implement the TableModel interface.

//get the number of rows
public int getRowCount() {
   return tableModel.getRowCount();
}

//get the number of column
public int getColumnCount() {
   return tableModel.getColumnCount();
}
```

```
    //get the name of column
    public String getColumnName(int columnIndex) {

        return tableModel.getColumnName(columnIndex);
    }

    //get the class of column
    public Class getColumnClass(int columnIndex) {
        return tableModel.getColumnClass(columnIndex);
    }

    //to check the cell editable
    public boolean isCellEditable(int rowIndex, int columnIndex) {
        return tableModel.isCellEditable(rowIndex, columnIndex);
    }

    //to get the specific value from table
    public Object getValueAt(int rowIndex, int columnIndex) {
        return tableModel.getValueAt(rowIndex, columnIndex);

    }

    //to update the value of cpecific cell in table
    public void setValueAt(Object aValue, int rowIndex, int columnIndex) {
        tableModel.setValueAt(aValue, rowIndex, columnIndex);
    }

    //to add table
    public void addTableModelListener(TableModelListener l) {
        tableModel.addTableModelListener(l);
    }

    //to remove the table
    public void removeTableModelListener(TableModelListener l) {
        tableModel.removeTableModelListener(l);
    }
  }
}
```

containercreat.class

```
//needed packages
import jade.core.ProfileImpl;
import java.net.InetAddress;
import jade.wrapper.AgentContainer;
import jade.core.Runtime;
import javax.swing.*;
import java.awt.event.*;

//this class to create container on the power company platform
public class containercreat extends JFrame implements java.io.Serializable {
   static AgentContainer c;
   static AgentContainer con;
   static ProfileImpl pr;
   static Runtime r;

   public containercreat(boolean what)throws Exception{

      //Get instance of Runtime of JADE runtime system
      Runtime r=Runtime.instance ();

      //Set automatic shutdown When there is no container
      r.setCloseVM (true);

      //Creates a default Profile for launching a platform
      //Set the container name, host, port, name of remote platform
      pr=new ProfileImpl("asd",-1,null);

      //set Profile parameter (container name) to local IP address
      pr.setParameter (
         ProfileImpl.CONTAINER_NAME,
         java.net.InetAddress.getLocalHost ().getHostAddress ());

      //Creates a new agent container with specific profile
      con=r.createAgentContainer (pr);

   }

   public static void main(String []args){
      try{

         //create jframe to make the program is visible
         containercreat now =new containercreat(true);
```

```
        now.addWindowListener(new WindowAdapter(){
           public void windowClosing(WindowEvent e){System.exit(0);}});
        now.setTitle ("senderClient");
        now.setSize (200,0);
        now.setLocation (100,100);
        now.setResizable (false);
        now.setVisible (true);

      }catch(Exception ex){

        //error if we Connont create container
        JOptionPane.showMessageDialog(null,"Connont create container: \n"
                                +ex.getMessage(),
                                "Runtime Exception",
                                JOptionPane.ERROR_MESSAGE);
        ex.printStackTrace ();
      }

    }


}
```

datat.class

```
//this class used as storage media for nedded info
public class datat implements java.io.Serializable{

    private String ContName;
    private String ContIP;
    private readings reads[]=new readings[3];
    private int rowcount;

    //constructor to specify container name and IP
    public datat(String ContName,String ContIP){
       this.ContName=ContName;
       this.ContIP=ContIP;
       rowcount=0;
    }

    //to get the name of container
    public String getConName(){
       return      ContName;
```

```
}

//to get the IP of container
public String getContIP(){
   return      ContIP;
}

//to add record of data to this node
public void addread(String processID,float value){
   reads[rowcount]=new readings(processID,value);
   rowcount++;
}

//get data stored in this node
public float getread(String processID){
   int i;
   for( i=0;i<rowcount;i++){
      if (processID.equalsIgnoreCase (reads[i].processID))
         return reads[i].value;
   }
   return -1;
}

//this inner class for data records
private class readings implements java.io.Serializable{

   public  String processID;
   public  float value;

   //constructor to create a record of data with value
   //ProcessID & value of reading
   readings(String processID,float value){
      this.processID=processID;
      this.value=value;
   }

  }
}
```

mobiagent.class

```
//important packages
import jade.tools.ToolAgent;
import jade.core.behaviours.Behaviour;
import jade.core.behaviours.OneShotBehaviour;
import jade.core.behaviours.SimpleBehaviour;
import jade.core.ContainerID;
import jade.core.AID;
import jade.core.Agent;
import jade.core.Location;

import jade.proto.SimpleAchieveREInitiator;

import jade.content.onto.basic.Action;

import jade.domain.mobility.MobilityOntology;

import jade.domain.FIPANames;
import jade.domain.FIPAAgentManagement.*;
import jade.domain.JADEAgentManagement.*;
import jade.domain.introspection.*;

import jade.domain.mobility.MoveAction;
import jade.domain.mobility.MobileAgentDescription;

import jade.content.lang.sl.SLCodec;

import jade.lang.acl.ACLMessage;

import javax.swing.JOptionPane;
import java.net.InetAddress;
import java.sql.*;
import java.util.Vector;
import java.util.Date;

public class mobiagent extends Agent {

    public String conname;
    public static int xx=0;
    public int counttmp;
    public transient boolean notstartup=false,failer;
    public Vector orderlist,count;
```

```
  private ACLMessage AMSSubscription = new
ACLMessage(ACLMessage.SUBSCRIBE);
  private ACLMessage AMSCancellation = new
ACLMessage(ACLMessage.CANCEL);
  private ACLMessage AMSRequest = new
ACLMessage(ACLMessage.REQUEST);

  //here the agent start after his state is ACTIVE
  public void setup(){

    failer=false;

    //Add registration, LoadTabel  and Moving behaviours
    Behaviour b1=new registerationBeh();
    Behaviour b2=new MovingBeh();
    Behaviour b0=new LoadTabelBeh();

    addBehaviour(b1);
    addBehaviour(b0);
    addBehaviour(b2);

  }

  //this metode called when doDelete() called
  public void takeDown() {
     float s1,s2,s3;
     String datas[]=new String[5];
     int i;

     //read the data from vector then store it in database
     for(i=0;i<orderlist.size ();i++){
        s1=((datat)orderlist.elementAt (i)).getread ("Process1");
        s2=((datat)orderlist.elementAt (i)).getread ("Process2");
        s3=((datat)orderlist.elementAt (i)).getread ("Process3");
        System.out.println("Process1: "+s1);
        System.out.println("Process2: "+s2);
        System.out.println("Process3: "+s3);
        datas[0]=((datat)orderlist.elementAt (i)).getContIP ();
        datas[1]=String.valueOf (s1);
        datas[2]=String.valueOf (s2);
        datas[3]=String.valueOf (s3);
        datas[4]=String.valueOf (s1+s2+s3);

        DBMS(datas);
```

```
    }
    System.out.println("*********************************************");
    System.out.println(getLocalName()+" is now shutting down.");

}

//this methode called when the agent is leaving the container
public void beforeMove(){
    System.out.println("*********************************************");
    System.out.println(getLocalName()+" is leaving "+here().getName ());
    System.out.println("*********************************************");

}

//this methode called when the agent successfully moved
public void afterMove(){
    counttmp=((Integer)count.remove (0)).intValue ();
    //check if the agent arrive to main container or happening fail
    if(here().getName ().equalsIgnoreCase ("Main-Container"))
        doDelete ();
    if(!failer){
        System.out.println("*********************************************");
        System.out.println(getLocalName()+" now in "+here().getName ());
        Behaviour b=new readdataBeh();
        addBehaviour(b);
    }
    count.add (0,(new Integer(counttmp+1)));
    setup ();
}

//return ACLMessages
protected ACLMessage getSubscribe() {
    return AMSSubscription;
}
protected ACLMessage getCancel() {
    return AMSCancellation;
}
protected ACLMessage getRequest() {
    return AMSRequest;
}

//behaviour of moving
class MovingBeh extends OneShotBehaviour{
```

```
public void action(){

    //check if there is more node
    if(counttmp<(orderlist.size ())){
       //get the next IP and set the location to it
       int ztmp=((Integer)count.elementAt (0)).intValue ();
       conname=((datat)orderlist.elementAt (ztmp)).getContIP ();
       System.out.println("count = "+ztmp+ " "+count.size ());
     }

     //set the location to Main-Container
     else {conname="Main-Container";

     }

     //call the move method with the AID of the agent and location
     moveAgent(mobiagent.this.getAID (), conname);
   }
}

//this behaviour for fill the agent with IP's
class LoadTabelBeh extends OneShotBehaviour{

   public void action(){

      try{

         if(count!=null&&!count.isEmpty ())
            counttmp=((Integer)count.elementAt (0)).intValue ();
         if(counttmp==0&&!notstartup){

            //Create Vector object
            count=new Vector();
            readInfoData();
            count.add (new Integer(0));
            notstartup=true;
          }


         }catch(Exception ex1){
            ex1.printStackTrace ();
         }
```

```
    //Fill the of ACL Message with specified concept
    ACLMessage requestMsg = getRequest();
    requestMsg.setOntology(MobilityOntology.NAME);

    //Validate the language and ontology
    getContentManager().fillContent(requestMsg, a);

    //Add SimpleAchieveREInitiator behaviour
    //The initiator sends a message (to performs  moving)
    addBehaviour(new AMSClientBehaviour("MoveAgent",requestMsg));

  } catch(Exception fe) {

    fe.printStackTrace();
  }
}

//this method is to store the data in power company database
public void DBMS(String[] infor){

   //Specify the name of DataSource
   String url = "jdbc:odbc:serverjdk";
   Connection con,con1;
   String s,Query,n;
   PreparedStatement updateV;
   try {
      Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
   }catch(java.lang.ClassNotFoundException e) {

      //error if ther is no jdbc:odbc bridge driver
      JOptionPane.showMessageDialog(null,"ClassNotFoundException: \n"
                              +e.getMessage (),
                              "IO Exception",
                              JOptionPane.ERROR_MESSAGE);
      System.err.print("ClassNotFoundException: ");
      System.err.println(e.getMessage());
    }

    try {
      con = DriverManager.getConnection(url);

      //determine the specific prepareStatement
      updateV = con.prepareStatement(
```

```
"INSERT INTO PMPR VALUES (?,?,?,?,?,?)");

//insert the data to database
updateV.setString(1,infor[0]);
updateV.setDate(2,new java.sql.Date(System.currentTimeMillis ()));
updateV.setString(3,infor[1]);
updateV.setString(4,infor[2]);
updateV.setString(5,infor[3]);
updateV.setString(6,infor[4]);

updateV.executeUpdate();

//close the connection to database
updateV.close ();
con.close();

} catch(SQLException ex) {

//error if any SQL error happen
JOptionPane.showMessageDialog(null,"SQLException: \n"
                    + ex.getMessage(),
                    "IO Exception",
                    JOptionPane.ERROR_MESSAGE);

System.err.println("SQLException: " + ex.getMessage());
}


}

//Read the database from host side to fill the vector
//with value of reading that carry the local ip
public void readClientData(){

//Specify the name of DataSource
String url = "jdbc:odbc:clientjdk";
Connection con;
String createString;
String ss,ss1,n,infor="",Hostname="",localad="";
String info[]=new String[8];

//determine the specific Query to get values and PID
createString = "select * from PMPT ORDER BY PID ASC";
Statement stmt;
```

```
try {
   Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
} catch(java.lang.ClassNotFoundException e) {

   //error if ther is no jdbc:odbc bridge driver
   System.err.print("ClassNotFoundException: ");
   JOptionPane.showMessageDialog(null,"ClassNotFoundException: \n"
                        +e.getMessage(),
                        "Class Not Found Exception",
                        JOptionPane.ERROR_MESSAGE);
   System.err.println(e.getMessage());
}

//connect to database and read the neaded data
try {

    con = DriverManager.getConnection(url);
    stmt = con.createStatement();
    ResultSet rs =stmt.executeQuery(createString);

    //add the values in the node
    while (rs.next()) {
       ss=rs.getString(1);
       ss1=rs.getString(2);
       ((datat)orderlist.elementAt (counttmp))
            .addread (ss,Float.parseFloat (ss1));
    }

    //close the connection to database
    stmt.close();
    con.close();
  } catch(SQLException ex) {

     //error if any SQL error happen
     JOptionPane.showMessageDialog(null,"SQLException: " +
                        ex.getMessage(),
                        "SQL Exception",
                        JOptionPane.ERROR_MESSAGE);

   }
}


//Read the database from server side to fill the vector
//with IP as nodes that will be filled with readings later
```

```
public void readInfoData(){

   //Specify the name of DataSource
   String url = "jdbc:odbc:serverjdk";
   Connection con;
   String createString;
   String ss;
   String info[]=new String[8];
   orderlist =new Vector();

   //determine the specific Query to get IP from DB at power company
   createString = "select IP from info ORDER BY IP ASC";
   Statement stmt;

   try {
      Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
   } catch(java.lang.ClassNotFoundException e) {

      //error if ther is no jdbc:odbc bridge driver
      JOptionPane.showMessageDialog(null,"ClassNotFoundException: \n"
                              +e.getMessage(),
                              "Class Not Found Exception",
                              JOptionPane.ERROR_MESSAGE);

   }

   //connect to database and read the neaded data
   try {

      con = DriverManager.getConnection(url);
      stmt = con.createStatement();
      ResultSet rs =stmt.executeQuery(createString);

      //create a datat node add the IP and hostname
      //and add it to vector
      while (rs.next()) {
         ss=rs.getString(1);
         orderlist.add (new datat("asd",ss));
      }

      //close the connection to database
      stmt.close();
      con.close();
   } catch(SQLException ex) {
```

```
        //error if any SQL error happen
        JOptionPane.showMessageDialog(null,"SQLException: " +
                                      ex.getMessage(),
                                      "SQL Exception",
                                      JOptionPane.ERROR_MESSAGE);


    }
}

class registerationBeh extends OneShotBehaviour{

    public void action(){

        //Register concepts, actions and predicates Ontology (JADEManagement)

        // Register the supported ontologies

        getContentManager().registerOntology(
        JADEManagementOntology.getInstance());

        //Register Introspection Ontology
        getContentManager().registerOntology(
        IntrospectionOntology.getInstance());

        //Register FIPA Agent Management specifications (FIPAManagement)
        getContentManager().registerOntology(
        FIPAManagementOntology.getInstance());

        //Register JADE mobility Ontology
        getContentManager().registerOntology(
        MobilityOntology.getInstance());


        //Register the supported languages (FIPANames)
        SLCodec codec = new SLCodec();
        getContentManager().
              registerLanguage(codec,FIPANames.ContentLanguage.FIPA_SL0);
        getContentManager().
              registerLanguage(codec,FIPANames.ContentLanguage.FIPA_SL1);
        getContentManager().
              registerLanguage(codec,FIPANames.ContentLanguage.FIPA_SL2);
        getContentManager().
```

```
            registerLanguage(codec,FIPANames.ContentLanguage.FIPA_SL);

        // Fill ACL messages fields
        AMSSubscription.setSender(getAID());
        AMSSubscription.clearAllReceiver();
        AMSSubscription.addReceiver(getAMS());
        AMSSubscription.setLanguage(FIPANames.ContentLanguage.FIPA_SL0);
        AMSSubscription.setOntology(IntrospectionOntology.NAME);
        AMSSubscription.setReplyWith(AMSSubscriber.AMS_SUBSCRIPTION);
        AMSSubscription.setConversationId(getLocalName());

        //Fill ACL messages fields with 3communicative performative to FIPA
        //(SUBSCRIBE, CANCEL and REQUEST)
        String content = AMSSubscriber.PLATFORM_EVENTS;
        AMSSubscription.setContent(content);

        AMSCancellation.setSender(getAID());
        AMSCancellation.clearAllReceiver();
        AMSCancellation.addReceiver(getAMS());
        AMSCancellation.setLanguage(FIPANames.ContentLanguage.FIPA_SL0);
        AMSCancellation.setOntology(IntrospectionOntology.NAME);
        AMSCancellation.setReplyWith(AMSSubscriber.AMS_CANCELLATION);
        AMSCancellation.setConversationId(getLocalName());
        // No content is needed (cfr. FIPA 97 Part 2 page 26)

        AMSRequest.setSender(getAID());
        AMSRequest.clearAllReceiver();
        AMSRequest.addReceiver(getAMS());

AMSRequest.setProtocol(FIPANames.InteractionProtocol.FIPA_REQUEST);
        AMSRequest.setLanguage(FIPANames.ContentLanguage.FIPA_SL0);

    }
  }


  //the initiator sends a single message
  //(i.e. it performs a single communicative act)
  private class AMSClientBehaviour extends SimpleAchieveREInitiator {

    private String actionName;


    public AMSClientBehaviour(String an, ACLMessage request) {
```

```
        super(mobiagent.this, request);
        actionName = an;
    }

    //this called when the responds with not understood by rma
    protected void handleNotUnderstood(ACLMessage reply) {
        System.out.println ("NOT-UNDERSTOOD received by RMA during "
                + actionName+ reply.getEnvelope ()
                .getComments ());
    }


    //this called when the responds with refused by rma
    protected void handleRefuse(ACLMessage reply) {
        System.out.println ("REFUSE received during "
                + actionName+ reply.getEnvelope ()
                .getComments ());
    }

    //this called when the responds with agree
    protected void handleAgree(ACLMessage reply) {
    }

    //this called when the responds with failure of action
    protected void handleFailure(ACLMessage reply) {
        System.out.println ("FAILURE received during " + actionName);
        failer=true;
        afterMove();
    }

    //this called when the responds with informed state
    protected void handleInform(ACLMessage reply) {
    }

  } // End of AMSClientBehaviour class

}
```