Development of Object-Oriented Application in the Internet Environment Using Java Language

การพัฒนาโปรแกรมประยุกต์วิธีเชิงวัตถุสำหรับระบบอินเตอร์เนต โดยใช้ภาษาจาวา

by

Miss Tiensawang Thamwanich

ID Number 38626076

Advisor

Dr. Dentcho N. Batanov

A PROJECT SUBMITTED IN PARTIAL FULFILLMENT
OF THE REQUIREMENT FOR THE SYSTEM DEVELOPMENT PROJECT
MASTER OF SCIENCE PROGRAM IN INFORMATION TECHNOLOGY
FIRST SEMESTER ACADEMIC YEAR 1997
FACULTY OF INFORMATION TECHNOLOGY
KING MONGKUT'S INSTITUTE OF TECHNOLOGY

ACKNOWLEDGEMENT

This project would not be put out successfully without the kind and sincere supports of many wonderful people. I wish to express my thanks and appreciation to all of them.

Two years of master degree gave me more than a second degree. I had new fellow classmates who had encouraged and helped each other throughout the program. One of them was Khun Zongwit Chareonrauywattana who had given me a great contribution during the programming process of my project. Without him my project were not be finished. Thank you very much.

Above all, the most profound gratitude is rightfully due to Prof. Dr. D.N. Batanov, my advisor, who has always helped enrich the object-oriented paradigm knowledge and intellectuality in his students. Being his advisee made me proud and gave me more confidence to defense my project.

My boss, Dr. Ammar Siamwalla, is greatly acknowledged for his kind support by giving me a big chance to continue my study without losing my job.

And thank you to my old friends who kept cheering me up when I was upset. Especially, KiKi, Khun Prisadee Jindahra who gave me a bright idea of an application for the project.

Last but not least, I would like to show my highest respect to my parents who always taught me how important education is. And I wish also to thank my sister who is always my last resort.

King Mongkut's Institute of Technology
Bangkok, Thailand

Tiensawang Thamwanich
December, 1997

CONTENTS

Page

บทคัดย่อ..... 1

Abstract..... 2

Acknowledgement..... 3

Contents..... 4

List of Figures..... 6

List of Table..... 7

CHAPTER

1. Introduction to Objects and Abstraction..... 9

2. Object-Oriented Concepts..... 11

2.1 Object-oriented software engineering..... 11

2.2 What is object-orientation?..... 12

2.3 Key concepts..... 13

2.3.1 Data abstraction..... 13

2.3.1.1 Objects..... 13

2.3.1.2 Classes and instances..... 15

2.3.1.3 Methods..... 18

2.3.2 Inheritance..... 18

2.3.3 Polymorphism..... 22

2.4 Garbage collector and multithreading..... 22

2.5 The disadvantages of an object-oriented approach..... 24

3. Java and the Internet..... 25

3.1 Java language..... 25

3.2 Client/Server Computing..... 28

3.2.1 Client-side..... 29

CONTENTS (cont.)

CHAPTER		Page
3. (cont.)	3.2.2 Server-side	29
	3.3 Three-Tiered Client/Server Architecture	30
	3.3.1 Advantages of Middleware	31
	3.3.2 Disadvantages of Middleware	32
	3.4 Java and Database connectivity	33
	3.5 Java and Common Gateway Interface (CGI)	34
4.	Comparing Java and C++	36
5.	Implementation: Electronic Alumni Directory	41
	5.1 Scope of an application	41
	5.2 System requirements	42
	5.3 Problem report	42
Bibliography		44
Author's Cirriculum vitae		45

LIST OF FIGURES

Page

Figure 1. Object..... 14
Figure 2. Class..... 16
Figure 3. Hierarchy of Classes..... 19
Figure 4. Single Inheritance..... 20
Figure 5. Multiple Inheritance..... 21
Figure 6. Java bytecode..... 26
Figure 7. Three-tiered Client/Server Architecture..... 30
Figure 8. Database Connectivity..... 33
Figure 9. Application: Alumni Directory..... 43

LIST OF TABLE

Page

Table 1. Limitations of Applets..... 27

CHAPTER 1

Introduction to Objects and Abstraction

Why has object-oriented programming had a significant impact on the software development? Object-oriented paradigm allows the real world to be represented more directly than the conventional ones do. Objects model the structure as well as the behavior of real-world domains or entities by means of a set of variables and a set of methods, respectively. We can say: "All things around us are objects and each of them belongs to some domain of interest."

The object-oriented approach uses the important abstraction principles for structuring objects by focusing on the similar properties of real-world objects and ignoring differences. There are three sub-processes of abstraction[1]: classification, aggregation, and generalization. Classification is used to define and group the similar objects into classes defining the structure and behavior of their instances. The aggregation is a process of forming a concept by describing the properties of the objects. And the last one, generalization is a process of forming a concept that covers a number of more special concepts, based on similarities of special concepts. The proper use of abstraction is one of the most significant characteristics of object orientation.

For more understanding, there are five basic characteristics of Smalltalk, the first successful object-oriented programming language and one of the languages on which Java is based. This represents a pure approach to object-oriented programming[2]

1. **Everything is an object.** Think of an object as a variable: it stores data but we can also ask it perform operations on itself by making requests. In theory, we can take any conceptual components in the problem we are trying to solve (dogs, building, services, etc.) and represent it as an object in our program.

---

[1] D.N. Batanov, Fundamentals of Object-Oriented Analysis, Design and Programming, (KMITL Bangkok: Lecture notes, 1996), page 2.

[2] Alan Kay, Object-Oriented Programming with Smalltalk, (McGraw-Hill, 1986), page 24.

2. **A program is a bunch of objects telling each other what to do by sending messages.** To make a request of an object, we "send a message" to that object. More concretely, we can think of a message as a request to call a function for a particular object.
3. **Each object has its own memory made up of other objects.** Or, we make a new kind of object by making a package contain existing objects. Thus, we can build the simplicity of objects.
4. **Every object has a type.** Using the parlance, each object is an instance of a class. Where "class" is synonymous with "type". The most important distinguishing characteristics of a class is "what messages can we send to it".
5. **All objects of a particular type can receive the same messages.** An object of type circle is also an object of type shape, a circle is guaranteed to received shape messages. This mean we can write code that talks to shapes, and automatically handle anything that fits the description of a shape. This is one of the most powerful concepts in the object-oriented programming.

Although it is too soon to understand all of the above concepts within this chapter but we will keep those as core of object-oriented programming to be analyzed within this paper.

CHAPTER 2

Object-Oriented Concepts

It is need not to say that object-oriented technology provides many benefits to software developers and their products. Many software companies claimed that their software products were object-oriented but, in fact, they were not. These may lead to misinformation and mistrust of object-oriented technology. In spite of the overuse and misuse of the term object-oriented, our understanding about object-oriented and its benefit will help us to decide whether the software product is object-oriented technology or not. This chapter introduces the key concepts behind object-oriented programming, design, and development in more details.

2.1 Object-oriented software engineering

For strong fundamental and easier to understand the following section, it is a good idea to start this chapter by paying attention on objet-oriented software engineering point of view.

One of experience end-user claims what happen to computer industry nowadays. In the past he can put a good software, for example a word processor program, in the several five-inches diskettes and bring with him anywhere. And he enjoyed with its work. But today he has to deliver a bunch of three-inches diskette and wait for almost half an hour to install in his machine. Moreover, the program has a lot of features which most of them he is rarely use. Moreover, it sometimes show him an unclue warning or error messages and after that the system is halted. Finally, the output is loosing what he has done.

Analyze from the above situation. We found that the significant reason of the size of the program which is bigger and bigger is their complexity. The more the developers put features in the program, the more complicated program is. And the more difficulty to manage. From the developers point of view, to minimize cost on software development is a goal of all software development manager. We usually hear

that 70 or even 80 percent of the overall development budget is consumed by maintenance costs. Object-orientation is a technology to solve or decrease this problem. Through classes, inheritance, and polymorphism, this technologies provide technical mechanisms for promoting and encouraging reusability and utilizing resource allocation as never before.

2.2 What is object-orientation?

Object-oriented is a way of thinking approach that allows to model and to analyze complex real-world domains and software systems to be developed in a easy and convenient way, using individual components or objects.[3] An object can be a tangible entity such as person, car or it can be intangible element such as an event. Therefore we can conclude that an object has both data (state) and functional (behavioral) components. Each object communicates with one another via 'messages'.

Finally, we will see what the following three major groups of computer users can expect from object orientation:[4]

- *End-users*: for this largest group of users, object orientation promises increasingly friendlier users interfaces. Object orientation can definitely help integrate multimedia data types in computing environments. For example, voice, image, and animation as well as text data. These can become a part of the computer's repertoire of stored and manipulated objects.
- *Application developers*: the most significant promise for this group is probably the object orientation provides tools that are easier and much more convenient to use. In addition, object orientation supports successfully the development of complex software systems.
- *System programmers*: this group includes developers of spread sheets, word processor, operating systems, database, etc. These are the power users of computing systems. For them, object-orientation enhances the

[3] D.N. Batanov, Fundamentals of Object-Oriented Analysis, Design and Programming, p. 5.
[4] Ibid., p. 8.

private information and methods that can be modified at anytime without affecting the other objects that depend on it. With our example, bicycle, we don't need to understand the gear mechanism in order to use it.

More about objects, each object communicates through its interface or we usually call as method. The interface establishes requests for a particular object. (the details of this process will be discussed in section 2.3.1.3.) One object sends a message (make a request) to another object, and the object figures out what to do with that message (execute program). In any relationship it is important to have boundaries. If all members of a class are available to everyone, then anyone can do anything they want. Though we should have a mechanism to prevent some objects to be accessed or manipulated directly from the outsiders. As we know from encapsulation, to get a service from one object we can ignore its internal mechanism.

For object-oriented programming language, Java, for example, uses four explicit keywords to set a boundary in a class: public, private, protected and friend. Their use and meaning are straight forward. These access specifiers determine the definition as follows:

Public means the following definition is available to everyone.

Private means no one can access that definition except the owner. Private is a wall between the owner and the outsiders. If someone tries to access a private member, they will get a compile-time error.

Friend has to deal with "package" which is Java way of making libraries. If something is friend it's available within the package, but not outside the package (sometimes this access is called 'package access').

Protected acts like private, with the exception that inherited classes have access to protected members, but not private members.

2.3.1.2 Classes and instances

In the real-world, we often have many objects of the same kind. For example, the bicycle is just one of many bicycles in this world. Using object-oriented approach, we can say that a bicycle object is an instance of the class of bicycle objects.

Or in the class of employee, there are Manager objects, Sales Person objects, and Engineer objects.

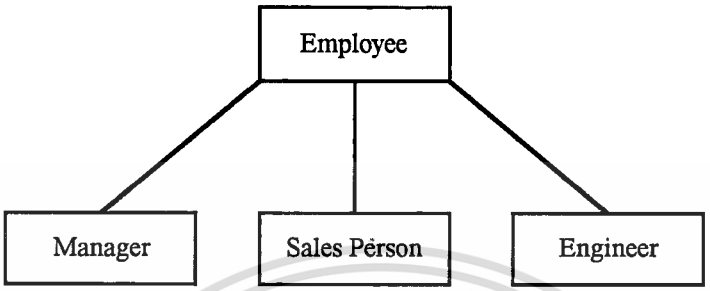

Figure 2. Class

We can say that a class is a template for multiple objects with something in common such as similar features. Every object is an instance of a class. The values for instance variables are provided by each instance of the class. Each instance contains the information that distinguishes it from other instances. After we have created the class, we must initiate (create an instance of it) before we can use it. When we create an instance of a class, we create an object of that type and the system allocates memory for that instance variables. Then we can use the object's instance methods to make it do something. In addition, classes can also define class variables and class methods. We can access class variables and methods either from an instance of a class or from a class. We don't have to instantiate a class to use its class variables and methods. Class methods can operate only on class variables. They do not have to access to instance variables or instance methods.

It is very important to note that the instances of the same class share the same instance methods which reside in the class itself. The system creates a single copy of all class variables for a class at the first time in a program. Then all instances of the class share its class variables. Let us get back to our example of bicycle, suppose that all bicycles have the same number of gear. To define an instance for number of gear is not necessary. Each instance has its own copy of the variable, but the value would be the same for every instance. In this situation, we can define a class variable that contains the number of gears. All instances share the class variables. If

one object changes the variable, it propagate to change for all other objects of that type. We can see that class provide a means of defining and managing the structure of objects. With this mechanism it is very much easier for developers to manage the state of variables.

When we have reached this point, notice that objects and classes look very similar to one another. So, what is the difference between classes and objects? In the real world, it is obvious that classes which we have defined as template (or blueprint in some books) are not the objects themselves. A template to build a bicycle is not a bicycle. However, we have problem about the confusion to differentiate classes and objects in program. This is because we use the term 'object' to both classes and instances. So we have to be very careful with this word.

Besides that objects provide the benefit of modularity and information hiding. Classes provide the benefit of reusability. Software developers use the same class and the same code to create many objects. Libraries or class libraries in programming language are good example of reusability.

Reusability is one of the greatest idea that object-oriented paradigm provide. But in fact, it turns out that this reusability is not easy to achieve when we do program. It takes experience and good design from the beginning of the programming process. Some experienced programmer suggests that the simplest way to reuse a class is to place an object inside a new class.[5] He calls this "creating a member object". Our new class can be made up of any number and type of other objects, whatever is necessary to achieve the functionality in our new class. This concept is call "composition" since we are composing a new class from existing classes. Sometimes composition is referred to as a *"has-a"* relationship as "a car *has a* trunk". He also adds that composition comes with a great deal of flexibility. The member objects of the new classes are usually private, making them inaccessible to the outsiders to use the classes. Thus we can change those members without affecting existing part of the program.

---

[5] Bruce Eckel, Thinking C++, (Osborne/McGraw-Hill, 1986), p. 52.

2.3.1.3 Methods

The functions that operate on an object are known as methods. They define the behavior that objects themselves can do or anything done to them. For example, here are some methods that we can do with a bicycle:

-Start pedaling

-Stop pedaling

-Change gear

-Break

Another common term of object-oriented software design originated in Smalltalk, is sending messages to an object.[6] In some books call this stage as 'message passing'. This is similar to a traditional function or procedure call. In some object-oriented programming languages do call methods as functions (C++ calls them member functions). But the term method is now common use in object-oriented technology, Java uses this word as well.

2.3.2 Inheritance

Once we have created a class it seems not so smart to create a brand new one that might have very similar functionalities. It would be better if we could take the existing data type, clone it and make additions and modification to the clone.

For example, when we see a bicycle, we would know that it had two wheels, handle bars, and pedals. So mountain bikes, racing bikes, and kid bikes are all different kind of bicycles. In object-oriented terminology, these bikes are all *subclasses or derived classes* of the bicycle class which is a *superclass or based class*.

[6] D.N. Batanov, Fundamentals of Object-Oriented Analysis, Design and Programming, p. 39.

Hierarchy of Classes

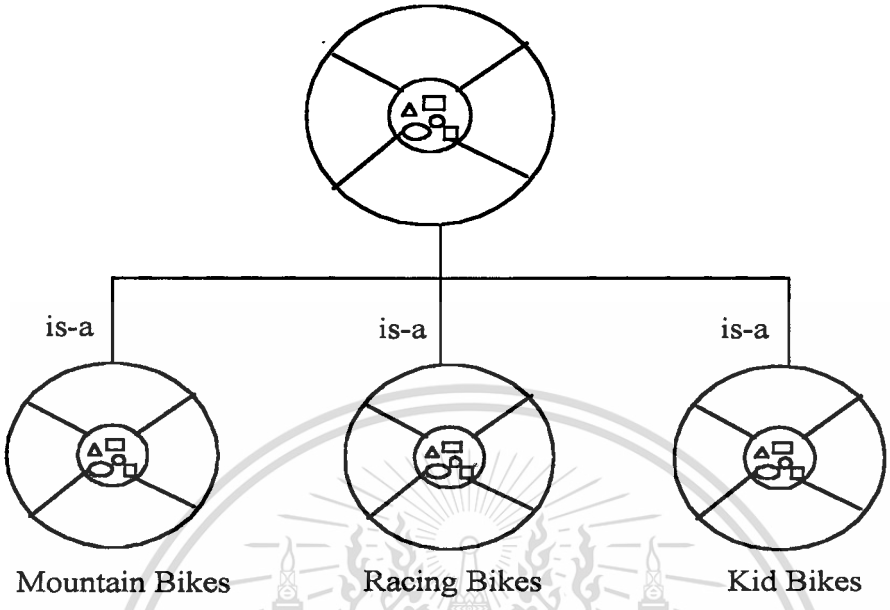

Figure 3. Hierarchy of Classes

Subclasses inherit all the methods and variables from their superclass. Inheritance enables modifications in a simple way. If we need to modify any property of all bicycles, we need only modify the root or superclass that will update every subclasses below. However, subclasses are not limited to the state and behaviors provided them by their superclass. Subclasses can add variables and methods to the ones they inherit from the superclass. Kid bike have extra wheel, some mountain bikes have an extra set of gears. Subclasses can also *override* inherited methods and provide specialized implementation for those methods. Or in the other case, if subclass defines a methods or variable that has the same name as a method defined in a superclass. The method definition that it found first (starting at the bottom to the top of the hierarchy) is the one that is actually executed. For example, if we have a mountain bike with an extra set of gears, we can override the superclass changeGears method so that we can use our new mountainGear method.

Inheritance is actually the reuse of class. So we are not limited to just one layer of inheritance. The inheritance tree, or class hierarchy, can be deep as we needed.

Moreover, we can define a method in a subclass with the same name as a method in superclass because overriding mechanism hides superclass's methods.

How about the relationship when we use overriding? In an overriding inheritance, the relationship between superclass and subclass is an *is-a* relationship. For example, mountain bike *is-a* bicycle. But when we apply overriding mechanism the new method will substitute or replace the method in the baseclass but not vice-versa, since it is not accessible from the baseclass. This can be describe as an *is-like-a* relationship.

Besides, compare with the real-world objects we found that every object is related to one or more other objects. Like ourselves, everyone must have parents. We can inherit some properties from more than one class of hierarchy. The object-oriented paradigm tries to model this real-world situation so that there are two forms of inheritance, namely *single inheritance and multiple inheritance*. We will use Employee as an example because it is more obvious to explain. In Figure 4

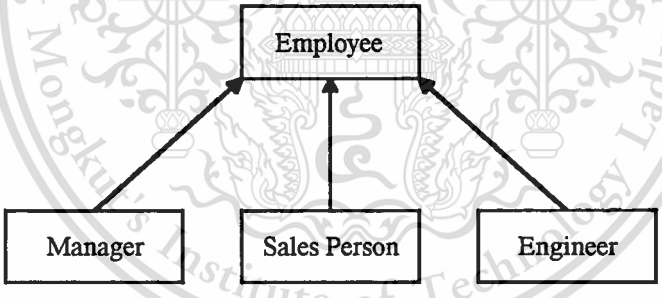

Figure 4. Single Inheritance

A company has employees. Employees can be managers, sales person or engineers. The company assigns properties to every employees such as employee number, salary, recruitment date, and so on. Therefore, managers, sales person, and engineers can all inherit common properties from a superclass of Employee. The arrows indicate that subclasses inherit methods and variable only from their superclass. This type of inheritance is called single inheritance because each of the subclasses inherits from no more than one supercalss.

For multiple inheritance, it allows a class to inherit from more than one superclasses as we can see in Figure 5.

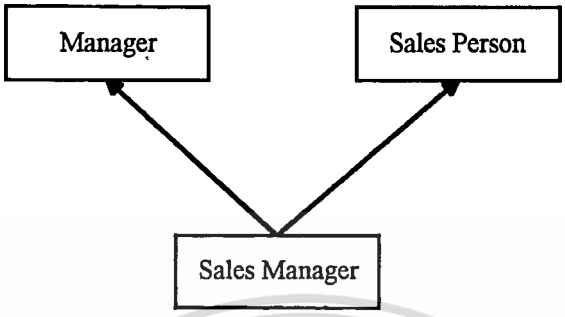

Figure 5. Multiple Inheritance

The rules for defining multiple inheritance must handle any conflict which may arise. For example, the same name may be used to represent different methods or variables which are inherited to the same subclasses. This increases the complexity of the program. A programming language must resolve this kind of conflict. Some object-oriented programming languages such as C++ and Smalltalk supports multiple inheritance but not Java.

Inheritance provides a powerful criteria of reusing classes by inheriting and creating specialization as necessary. We can conclude the benefits of inheritance which help developers achieving the goals of reusability and extendibility.

- Reusability: in the world of programming, we must face the problem of variation and repetition. Subclasses provide specialized behaviors from the common elements provided by the superclass. Through the use of inheritance, programmers can reuse the code in the superclass over and over again.
- Extendibility: The developers can benefit from tried and tested classes through use in previous developments. System are designed in a more modular and extensible way which are readily and easier to modify as user requirements change.

2.3.3 Polymorphism

Polymorphism is derived from the word 'polymorphic' means "having many forms".[7] In the sense of object-oriented programming, polymorphism refer to the fact that a single operation can have different behavior in different objects. In other words, different objects react differently to the same message. For example, an adding operation can be diffident as the process of adding two numbers and generate the summation. While in the sense of programming, this operation can be the concatenation of the string. Polymorphism facilitates the reuse of the same program in different instances and make for fewer changes to software. The classic example to illustrate polymorphism is drawing geometrical shapes such as Circle, Square, and Triangle. Drawing a Circle causes different code to be executed than drawing a square or a Triangle, but when the message of method or function draw is sent to an anonymous shape, the correct behavior occurs based on the actual type of the shape. With the mechanism of polymorphism, we can send a message to an object even though we don't know what specific type it is, and the right thing happens.

The process that object-oriented programming languages use to implement polymorphism is call *dynamic binding or late binding* where variables, procedures, types, functions or methods will be bound at the run-time of the program.

2.4. Garbage collector and multithreading

Technically, object-oriented paradigm is just about abstract data type, inheritance and polymorphism, but there are another issues which is more advance should be discussed briefly.

One of the most important factors concern the way objects are created and destroyed. Each object requires resources in order to exist, mostly kept in memory, which is very interesting to look at its mechanism. There are two approaches to be analyzed: the first one is placing the objects on the stack (there are some books called automatic or scope variables.) or in the static storage area. In this case, we must know the exact quantity, lifetime, and type of objects while we are writing the program

---

[7] George Wilkie, Object-Oriented Software Engineering: The Professional Developer's Guide, (Addison-Wesley, 1994.), p. 31.

which is difficult in some applications. The second approach is to create objects dynamically in a pool of memory called the *heap*. With this approach we don't know until run-time how many objects we need. Also the lifetime and object type are determined while the program is running. If we need a new object, simply make it on the heap memory at the point that we need it. Because the storage is managed dynamically, at the run-time. The dynamic approach provides the greater flexibility to solve programming problems. However, if we create a new object on the heap, the compiler has no information of its lifetime. So how does the object can be destroyed? There are two more options: programmer indicate in the program whenever he wants to destroy it , or the environment provide a process called *garbage collector* that automatically scan for the idle object and destroy it. Anyway, garbage collector requires the system that support *multithreading* and the other overhead.

Briefly about multithreading[8], a fundamental concept in computer programming is the idea of handling more than one task at a time. Many programming problem require that the program be able to stop what it's doing, deal with some other problem, and return to the main process. The solution has been approached in many ways: programmers with the low-level knowledge write a program to control interrupt service routines. Although it works well but it is difficult and non-portable. And most of the programming problem need only partitioning problem into separately-running pieces so the whole program can be more responsive. Within a program, these separately-running pieces are called *threads* and the general concept is called multithreading. A common example of multithreading is the user interface: by using threads, when a user presses a button they can get a quick response, rather than being forced to wait until the program finishes is current task.

Normally threads are just a way to allocate the time of a single processor, but if the operating system supports multiple processors, each thread can be assigned to a different processor and they can truly run in parallel. One of the very convenient features of multithreading at the language level is that the programmer does not need to worry about whether there are many processors or just one, the program is logically

---

[8] Bruce Eckel, Thinking C++, p. 36.

divided into threads and if the machine has more than one processor then the program just runs faster, without any special adjustment. This makes threading sound simple but if we have more than one thread running that's expecting to access the same resource we now have problem. To solve the problem, resource that can be shared (like the printer) must be locked while they are being used. So a thread locks a resource, completes its task, then release the lock so someone else can use the resource.

Java's threading is built into the language, which makes a complicated subject much simpler. The threading is supported on an object level, so one thread of execution is represented by one object. Moreover, Java also provides limited resource locking: it can lock the memory of any object (which is one kind of shared resource) so that only one thread can use it at a time.

2.5. The disadvantages of an object-oriented approach[9]

Every technologies have their problem. The better way to understand them is try to learn every facets both advantages and disadvantages. All of the above sections are about the advantages of this technology so how about the other side.

The single biggest disadvantage at present is the immaturity of the techniques and tools. The lack of a standard notation for expressing object-oriented design. This is a significant obstacle to promote this technology among developers and software engineers. There should be a set of symbols that anyone familiar with object concepts can understand.

Programming language development environments need more available classes to maximize the ability of reusability. Not only many useful concepts of object-oriented approach bring many benefits, but also bring problems associated with increased coupling (or linkage) between classes. Polymorphism and dynamic binding make program more complex and difficult to test.

---

[9] George Wilkie, Object-Oriented Software Engineering: The Professional Developer's Guide, p. 13.

CHAPTER 3

Java and the Internet

Java is, in fact, another computer programming language. But when it is combined with the Internet it is more than an ordinary programming language. Because Java not only solves traditional stand-alone programming problem, but also solve programming problems on the World-Wide-Web. This is the strong reason why is it being promoted as a revolutionary step in computer programming development.

In this chapter, it is not take more time on what Java language is and how to program Java because there are thousands of book about Java programming that we can find very easily in the bookstores. But this chapter will concentrate on Internet issue which is relevant to Java and the implementation level of this project. The working process of Java on the Internet is that users download Java bytecodes, which is actually a special piece of program, from the Internet or server side and run them on their own machines or client side.

3.1 Java Language

Although a scripting language like Perl can solve most of the client/server programming problems, but there still some problems which are more complicate. The most popular solution today is Java. Not only is it a very powerful programming language built to be secure, cross-platform and international, but Java is being continuously extended to provide language features and libraries that handle problems that are difficult in traditional programming languages, such as multithreading, database access, network programming and distributed computing. Java allows client-side programming via the *Applet*.

An Applet is a mini-program that will run only under a Web browser. The Applet is downloaded automatically as part of a Web pages and when the Applet is activated it executes a program. This is part of its powerful tools, it provides us with a way to automatically distribute the client software from the server, at the time the user

3. Processing is offloaded to the user's system, which presumably, is going to do it much faster than some host that is dealing of data needs to be computed, we do not have to worry about the speed of transmission from the host machine since the data is computed locally.

Although Java Applet has many advantages, it also has limitations. Most of the limitations of Applets (compared to Java applications) are related to the potential security risks of allowing an untrusted, and possibly hostile, `Applet` access to a client machine. The following table lists the differences between Applets and applications from a security-based perspective.

Table 1. Limitations of Applets

Criteria	Remote Applet	Local Applet	Applet Viewer	Application
Read local file	✗	✗	✓	✓
Write local file	✗	✗	✓	✓
Get file information	✗	✗	✓	✓
Delete file	✗	✗	✗	✓
Run another program	✗	✗	✓	✓
Read the `user.name` property	✗	✓	✓	✓
Connect to network port on server	✓	✓	✓	✓
Connect to network port on the other host	✗	✓	✓	✓
Load Java library	✗	✓	✓	✓
Call `exit`	✗	✗	✓	✓
Create a popup window	✓	✓	✓	✓

Source: Core Java, SunSoft.

3.2 Client/Server Computing

To understand the Web, we should start from the client/server system. The primary idea of client/server system is that we have an information center which keeps some kind of data, in a database. This information will be distributed to some group of people or machines. A key client/server concept is that the repository of information is located in the center so that it can be changed or updated and those changes will propagate out to the users. The software that distributes the information and the machine where the information and software reside is called the 'server'. The software that resides on the remote machine which communicate with the server, retrieve the information and processes and displays output on the remote machine is called 'client'.

The client/server computing is more complicated when a server is trying to serve many clients at once. An efficient database management is required to monitor and control information access. For example, the system should allow clients to insert new information into a server and making sure that the new data will be placed in the proper database.

The Web is actually one giant client/server system. But it is quite a complicated network system. The users don't need to know where are the server, since all they care about is connecting to and interacting with one server at a time. Things are more complicated when users what to do more than just deliver pages from a server; they want full client/server capability so that the client could feed information back to the server, for example to do database lookups on the server, to add new information to the server or to place an order (which require more security that the original systems offered). These are the changes we have been seeing in the development of the Web.

In the Java environment, the processing burden is mostly on the client's. The client requests a program from the server. The server sends that program in a platform-independent form (bytecode). The client then runs that code on the client machine and displays the results. Also, the server not only serves data, it also serves

the application itself, something that server did not do in the traditional client/server model.

3.2.1 Client-side

The Web's initial server-browser design provided for interactive content, but the interactivity was completely provided by the server. The server produced static pages for the client browser, which would simply interpret and display them. Basic HTML (HyperText Markup Language) contains simple mechanisms for data gathering: text-entry boxes, check boxes, lists and drop-down lists, as well as a button which can be programmed to do only two things: reset the data on the form or "submit" the data on the form back to the server. This submission passes through the Common Gateway Interface (CGI) provided on all Web servers. The text within the submission tells CGI what to do with it; the most common action is to run a program located in the *cgi-bin* directory of the server. This program can be written in most languages, but Perl is a common choice because it is designed for text manipulation and is interpreted, and so can be installed on any server regardless of processor of operating system.

For interactive pages, the process is start from the client asks the server to execute an external program. This is a program that is completely independent (so that the name external program), but it reside on the same machine as the HTTP server. The external program is either a native, executed program or a script that can be execute by the operating system. Also, the server has the appropriate permission to execute this program. the external program is executed and its output is sent directly to client. It is up to the external program to include the necessary headers in its output.

3.2.2 Server-side[11]

Before we can link a database to the WWW, we must first have a Web server. The Web server does all the thing the server must do in a client/server environment, such as managing files, responding to client's requests, and sending the clients the requested information.

---

[11] Piroz Mohseni, Web Database Primer Plus (Waite Group, 1996) p. 127.

There are numbers of servers on the market. What distinguishes each server from the others is the extra functionalities it offers. Some manage multiple requests better than others. Some manage their resources better. Some offers security features. Some even offer a direct connection to a database server by bypassing the CGI link (such as ORACLE Web server).

To choose a server, we must consider our application and look into a few years from now. Here are some questions to answer:

1. How many people (clients) will communicate with the server?
2. What platform and operating system will the server run on?
3. How important is security in the application we are developing?
4. How much do we plan to spend on the server?
5. Does the server support CGI? How efficient is its mechanism?
6. What logging mechanism does the server offer? How can we measure the server load?

3.3 Three-Tiered Client/Server Architecture

In the implementation level of this project, it was applied the three-tiered client/server architecture to access database. So we should pay attention upon this topic for more understanding.

Three-tiered application processing adds an additional layer to the standard client/server model which mentioned in section 3.2. By this approach, we can decrease workload on the server-side into two stages as shown in Figure 7.

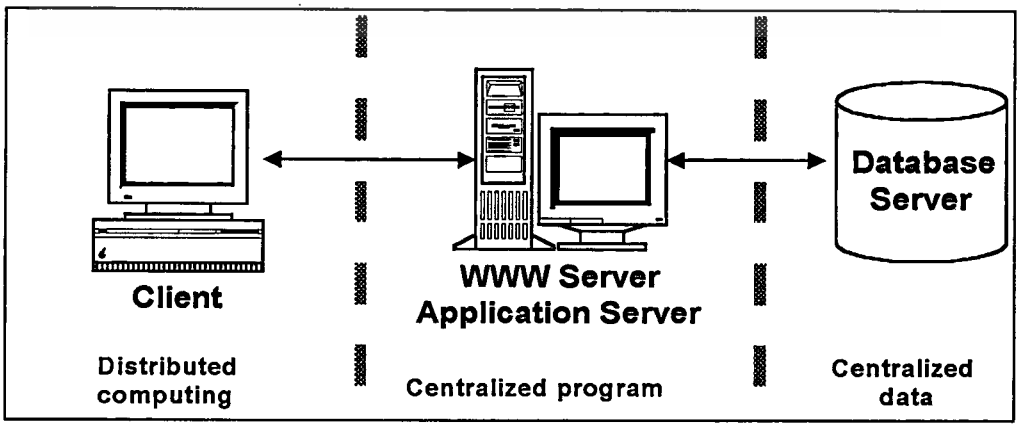

Figure 7. Three-tiered Client/Server Architecture

With three-tiered applications, a client-side application still performs the GUI computing. An additional server-side is a middleware. Program with this architecture starts from the parameters transmission from the client-side via the middleware which responsible for an application such as Web server to the database server. At the third tier, it process command, for example, SQL commands and sends results back to the client-side through the middleware again. We can see that middleware has important role so knowing advantages and disadvantages can help us whether we need it or not.

3.3.1 Advantages of Middleware[12]

- *Concurrency*: We can program the application server to handler concurrency issues, off-loading the task from the database server. We can implement concurrency checking entirely on the application server, if necessary. This process involves checking to see if a specific data object requested by a client has changed since the current request, asking the client to update the previously retrieved data, and alerting the user.
- *Legacy Databases*: Databases that operate on older network protocols can be piped through an application server running on a machine that can communicate with the database server, as well as with remote Internet clients. A middleware can tell a non-networked legacy database can be used to provide Internet access to its data. The application server can reside on the same machine as the non-networked database, and provide network access using a client that communicates to the application server.
- *Security*: We can program/obtain an application server that supports a secure connection to the remote clients. If we keep the local connection between the database server and the application server restricted to each other, we can create a fairly secure system. In this type of setup, our database server can only talk to the application server, so the threat of someone connection directly to the database server and causing damage is

[12] Pratik Patel and Karl Moss, JAVA Database Programming with JDBC(Arizona, Coriolis Group, 1996), p.223.

greatly limited. However, we must be sure that there are not no loopholes in our application server.

• *Simultaneous Connections*: The application server, in theory, can maintain only one active connection to the database server. On the other side, it can allow as many connections to itself from clients as it wants. In practice, however, significant speed problems will arise as more users attempt to use one connection. Managing a number of fixed connections to the database server is possible, though, so this speed degradation is not noticeable.

3.3.2 Disadvantages of Middleware[13]

• *Speed*: Speed is the main drawback to running an application server, especially if the application server is running on a slow machine. If the application server does not run on the same machine as the database server, there may be additional speed loss as the two communicate with each other.

• *Security*: If our application server is not properly secured, additional security holes could easily crop up. For example, a rogue user could break into the application server, then break into the database server using the application server's functions. Again, we must take great care to make sure that unauthorized access to the database server via the application server is not possible.

• *Reliability*: Adding an application server to the system introduces potential problems that may not be present in a two-tier system, where the clients are communicating directly with the database server.

[13] Ibid, p.224.

3.4 Java and Database Connectivity

There are three categories of database connectivity as shown in Figure 8.

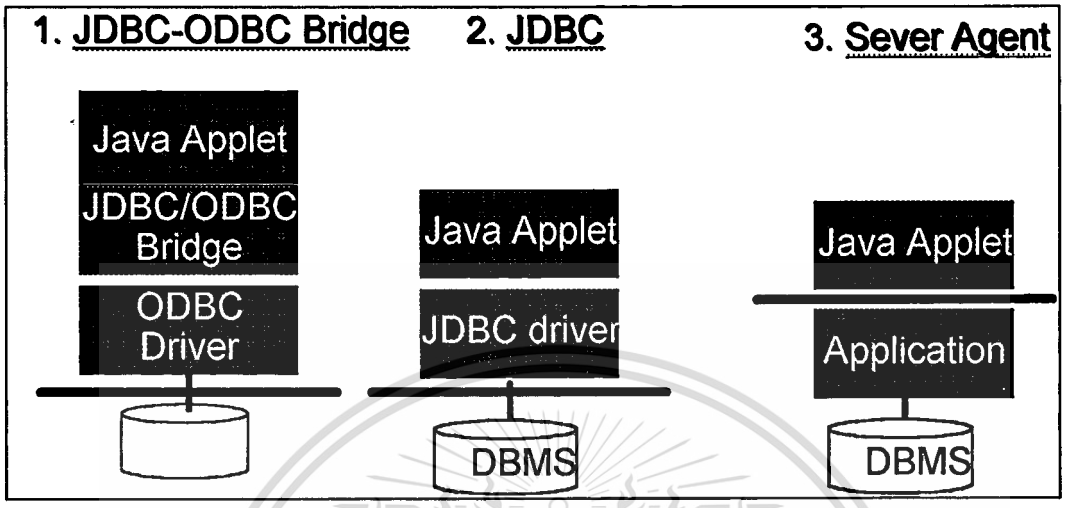

Figure 8. Database Connectivity

a). *JDBC-ODBC Bridge:* Because most of the database nowadays supported by ODBC driver so JavaSoft and Intersolv have produced this approach to simplify existed database connection. It converts JDBC method calls into ODBC function calls, allowing Java programs to communicate with any database with ODBC capabilities. Several vendors have produced JDBC drivers to allow connection to numerous databases like ORACLE, Sybase, Informix, MS-Access, CA-Ingres, Interbase and ADABAS, etc. But the drawback of this approach is that the client-side (the stacks above the line) needs to install ODBC driver or specific driver required by the DBMS at the database server-side. Client for this approach needs to be good enough because there are many parts to be resided on this side. Sometimes we call 'Fat-Client'.

b). *JDBC (Java DataBase Connection):* This is a package contains a set of classes, interfaces and exceptions for connecting to databases, executing SQL statements and retrieving the results from these statements. The JDBC API (Application Programming Interface) communicates with JDBC drivers which can be written in either Java or native methods; it is these drivers which handle the transfer of data between the API and a particular database. This approach is good because client-side need not to install any driver. (JDBC driver is already embedded in JDK 1.1.2, the

latest version.) But the problem is JDBC itself which does not easy to use. So there are some third-party products create the middleware to simplify this approach.

c). *Server Agent:* This approach is the most convenient to use nowadays because there are many third-party product available. This project also apply this approach by using the JetConnect, the special class library downloaded from the Internet. This approach is good for database center because the database server can be modified at anytime without worrying the client-side. It is centralized database because the application has to reside totally on the server side.

3.5 Java and Common Gateway Interface (CGI)

The Common Gateway Interface (CGI) which has been mentioned before is a standard for interfacing external applications with information servers. We have to write CGI program to access database server. It is the heart of our Web application, and in it we perform all the functions necessary in our application. Most important, we need to write the CGI code with the Web environment in mind. There are several points to concern for CGI programming. Since a CGI program is executable, it is basically the equivalent of letting the world run a program on our system, which causes security concerns for many system administrators. A second point of concern is the overhead involved with CGI programs. An application server might receive several requests in a short time period, each requiring the launch of a CGI program. In the same way that opening up several applications on our PC can slow it down or yield an "Out of memory" error, running multiple CGI programs can cause difficulties for our system.

Compare with Java, it is the Internet programming language that introduced new ways of linking database to the Web. With Java, the server can also serve an entire application or part of it. This application can establish a connection to a database using HTTP or its own independent communication channel such as ODBC or proprietary drivers. The unique features of Java make it greater than traditional way like CGI. A Java program can be executed on almost all platforms as explained in section 3.1. While the CGI program need to depend on the platform that we have

developed the original application. We need to recompile every time when we move to the new platform. Moreover, the application code resides on the client, no inquiries to the server are needed. Java applications are all event-driven, which means we can capture events associated with the objects in your application and perform appropriate code. We can tell immediately that the user performed an action and will not have to wait for the action to be sent to HTTP server and our CGI program.

Another unique feature of Java is its ability to access resources over the Internet. For example, in C we can open a file on the local file system and perform some operations on it. With Java, we can open a file residing on a different network as shamelessly as we would open a file on our own network. As a result, Java is a truly an Internet programming language.

CHAPTER 4

Comparing Java and C++

Since Java was derived from C++ which is one of the popular object-oriented programming languages. Java is a language in between Smalltalk and C++. It's also fully object-oriented - even more than C++ but less than Smalltalk. This chapter tries to gather the likeness and differences of these two languages from many sources. Surprisingly, there are number of differences which are intended to be significant improvements and to simplify the usage. To understand all of the comparisons is very difficult because some of them need to practice programming and need to know advance knowledge about the object-oriented programming which this paper could not cover. So it is highly recommend to have further advance topics. However, this chapter will take you through the important features that make Java distinct from C++.

1. The biggest problem is speed: interpreted Java runs something like 20 times slower than C++. Nothing prevent the Java language from being compiled and there are just-in-time compilers.
2. Java has both kinds of comments like C++ does.
3. Everything must be in a class. There are no global functions or global data. If we want the equivalent of global, make **static** methods and **static** data within a class. There are no structs or enumerations or unions. Only classes.
4. All method definitions are defined in the body of the class. Thus, in C++ it would look like all the functions are inlined, but they are not.
5. Class definitions are roughly the same form in Java as in C++, but there is no closing semicolon. There are no class declarations , only class definitions.

```
class Type {
     void Function( )  {  /* method body */ }
}
```

6. There is no scope resolution operator :: in Java. Java uses the dot for everything, but can get away with it since we can only define elements within a class. Even the method definitions must always occur within a class so there is no need for scope resolution there, either. One place where we will notice the difference is in the calling of **static** methods: we say **ClassName.methodName( );**. In addition, package names are established using the dot, and to perform the equivalent of a C++ # include we use the import keyword. For example: `import java.awt.*;`
7. Java, like C++, has primitive types for efficient access. In Java, these are **boolean, char, byte, short, int, long, float,** and **double**. All the primitive types have specified sizes that are machine-independent for portability (this must have some impact on performance, varying with the hardware). Type-checking is much tighter in Java. For example: Conditional expressions can only be **boolean**, not integer.
8. The **char** type uses the international 16-bit Unicode character set, so it can automatically represent most national characters.
9. Static quoted strings are automatically converted in **String** objects. There is no independent static character array string as there is in C/C++.
10. Arrays are quite different in Java. There is a read-only **length** member that tell us how big the array is, and run-time checking throws an exception if we go out of bounds. All arrays are created on the heap. Java does not support multidimensional arrays as in C/C++ but we can assign arrays that contain other arrays (the array handle is simply copied). The array identifier itself is a first-class object, with all the methods commonly available to all other objects.
11. All non-primitive data types can only be created using the keyword **new**. There is no equivalent to creating non-primitive objects "on the stack" as in C++. All primitive data types can only be created directly, without **new**. There are wrapper classes for all primitive classes so we can create equivalent heap-based objects with **new**. (Arrays of primitives are a

special case: they can be allocated via aggregate initialization as in C++, or by using **new**).

12. Java has no preprocessor. If we want to use classes in another library, we say **import** and the name of the library. There are no preprocessor-like macros.
13. There are Java pointers in the sense of C and C++. When we create an object with **new**, we get back a reference. For example:

    ```
    String s = new String (``Hello'');
    ```

    However, unlike C++ references that must be initialized when created and cannot be rebound to a different location, Java references do not have to be bound at the point of creation, and they can be rebound at will,which eliminates part of the need for pointers. The other reason for pointer is to select any place in memory (which makes them unsafe, which is why Java does not support them).
14. Java has constructors, similar to constructors in C++. We get a default constructor if we do not define one, and if we define a non-default constructor, there is no automatic default constructor defined for us, just like C++. There are no copy-constructors, since all arguments are passed by reference.
15. There are no destructors in Java. There is no "scope" of a variable, to indicate when the object's lifetime is ended - the lifetime of an object is determined instead by the garbage collector. There is a **finalize( )** method that's a member of each class, like destructor, but **finalize( )** is called by the garbage collector and is only supposed to be responsible for releasing resources. If we need something done at a specific point, we must create a special method and call it, not rely on **finalize( )**. All objects in C++ will be destroyed, but not all objects in Java are garbage collected. Because Java does not support destructors, we must be careful to create a cleanup method if necessary, and to explicitly call all the cleanup methods for the base class and member objects in our class.

16. There is no **goto** in Java. The one unconditional jump mechanism is the **break** label or **continue** label, which is used to jump out of the middle of multiply-nested loops.
17. Java uses a singly-rooted hierarchy, so all objects are inherited from the root class **Object**. In C++ we can start a new inheritance tree anywhere. This seems restrictive, but it gives a great deal of power since we know that every object is guaranteed to have at least the **Object** interface. C++ appears to be the only OO language that does not impose a singly-rooted hierarchy.
18. Java has built in multithreading support. There is a **Thread** class that we inherit to create a new thread by override the **run( )** method.
19. Instead of controlling blocks of declarations like C++ does, access specifiers **(public, private, and protected)** are placed on each definition for each member of a class. Without an explicit access specifier, the element defaults to "friend", which means it is accessible to other elements in the same package but inaccessible outside the package. The class, and each method within the class, has an access specifier to determine whether it is visible outside the file. Sometimes the **private** keyword is used less in Java because "friend" access is often more useful than excluding access from other classes in the same package (however, with multithreading the proper use of **private** is essential). The Java **proteced** keyword means "accessible to inheritors and to others in this package." There is no equivalent to the C++ **protected** keyword which means "accessible only to inheritors" **(private protected** used to do this, but it was removed.).
20. Inheritance in Java has the same effect as in C++ , but the syntax is different. Java uses the **extends** keyword to indicate inheritance from a base class, and the super keyword to specify methods to be called in the base class that have the same name as the method we are in (however, the super keyword in Java only allows us to access methods in the parent

class, one level up in the hierarchy. Base-class scoping in C++ allows us to access methods that are deeper in the hierarchy).

21. All functions are implemented as methods. There are no functions that are not tied to classes.
22. It is far easier to turn out bug-free code using Java than using C++. Some estimates say that roughly every 50 lines of C++ code has at least one bug.

CHAPTER 5

Implementation: An Electronic Alumni Directory

To understand all of the theoritical frameworks from chapter one through chapter four is sometimes not easy to accomplish by only reading. Especially for the object-oriented paradigm, which composed of three main stages: objec-oriented analysis (OOA), object-oriented design (OOD), and object-oriented programming (OOP). And also because one of the objectives of this project is to program so that making this project completely and learning how to apply object-oriented features by programming is the best way to make understanding.

5.1 Scope of an Application

An application which has been developed for this project was an electronic alumni directory for the faculty of Information Technology, KMITL. This application program intended to serve all of the former students of the faculty for both graduate level and undergraduate level. Anyway the program was based on the existed database at the development time, it can be modified easily to fit with any changes in the future.

Because this application was related to the database issue and to maintain our database to be up-to-date, it needed to do the fundamental database functions as followed:

1). To search for desired information with appropriate search criteria

2). To add new records to the database

3). To update or modify existed database and

4). To delete obsolete information in the database

The first task was widely used by the alumni students and it was the main service of this program. So it was opened for everyone. But the other three tasks needed to access to the database directly so that it should have some security mechanism to protect the database from unfriendly accesses. This program provided

password mechanism for authorized user only, for example in this case, the database manager.

5.2 System Requirements

To use this program it needs to complete the following requirements.

Server-side:

1. Any Web (HTTP) server
2. ODBC driver to connect database (e.g. MS-Access, Microsoft SQL)
3. Installing the special class library: JetConnect
4. Running Java application, a server daemon processes

This server-side application is responsible for

- controlling connections from the remote clients (because this program is multi-user program)
- receiving SQL commands from the remote clients and communicate with database server (SQL: e.g. query, insert, update, delete)

5. Installing Applet for calling from the remote clients. This is responsible for
   - controlling GUI at the client-side
   - creating and sending SQL commands to the server

Client-side:

It is no need to installing additional driver just running an appropriate URL path from the Internet browser program which support Java Applet such as Netscape Gold, Netscape communicator, or Internet Explorer.

5.3 Problem Report

Because JDK 1.0.2, which was used to develop this program, does not support international language so it could not search by Thai fonts.

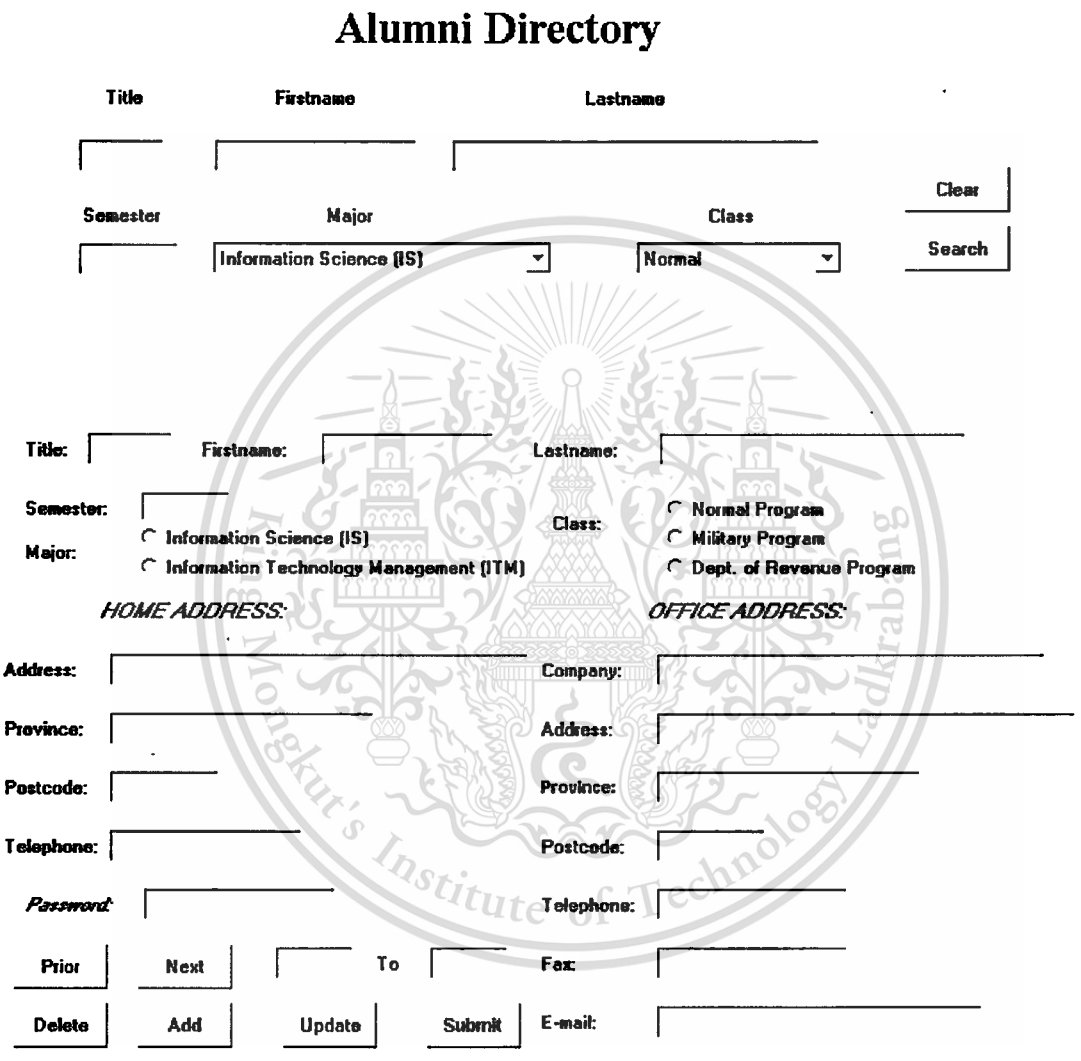

Figure 9. Alumni Directory

BIBLIOGRAPHY

Barkakati, Nabajyoti. Object-Oriented Programming in C++. Indiana: SAMS, 1991.

Batanov, D.N. Fundamentals of Object-Oriented Analysis, Design and Programming. KMITL, Bangkok: Lecture Notes, 1996.

Campione, Mary and Kathy Walrath. The Java™ Tutorial: Object-Oriented Programming for theInternet. Massachusetts: Addison-Wesley, 1996.

Cornell, Gary and Cay S. Horstmann. Core JAVA. Second Edition. New Jersey: Prentice Hall, 1997.

Deitel, Harvey M. and Paul J. Deitel. Java How to Program. New Jersey: Prentice Hall, 1997.

Eckel, Bruce. Thinking C++. San Francisco: McGraw-Hill, 1986.

Gottlob, Georg., Michael Schrefl and Brigitte Röck. "Extending Object-Oriented Systems with Roles." Journal of ACM Transaction on Information Systems, Vol. 14, No. 3 (July 1996) pp. 268-296.

Jacobson, Ivar and others. Object-Oriented Software Engineering: A Use Case Driven Approach. Fourth Edition. Massachusetts: Addison-Wesley, 1992.

Jebson, Brian. JAVA™ Database Programming. New York: John Wiley & Sons, 1997.

Kay, Alan. Object-Oriented Programming with Smalltalk. New York: Addison-Wesley, 1985.

Lemay, Laura and Charles L. Perkins. Teach Yourself Java in 21 Days. Indiana: Sams.net, 1996.

Mohseni, Piroz. Web Database Primer Plus. California: Waite Group, 1996.

Patel, Pratik and Karl Moss. JAVA Database Programming with JDBC. Arizona: Coriolis Group, 1996.

Simon, Errol. Distributed Information System: From Client/Sever to Distributed Multimedia. London: McGraw-Hill, 1996.

Wilkie, George. Object-Oriented Software Engineering: The Professional Developer's Guide. Cambridge: Addison-Wesley, 1994.

CIRRICULUM VITAE

Author's Name	Miss Tiensawang Thamwanich
Date of Birth	11 February 1969
Place of Birth	Bangkok
Bachelor Degree	Bachelor of Arts (Economics)
Graduate Institute	Thammasat University
Graduate Year	1991 (2534)
Present Carrier	Researcher Thailand Development Research Institute Foundation (TDRI)