

2001

# CORBA: A Quantitative and Qualitative Comparison of Industrial Strength, Commercial CORBA ORBs for the JAVA Platform

Michelle Leigh McKeller

*University of North Florida*



# CORBA:
# A QUANTITATIVE AND QUALITATIVE COMPARISON OF INDUSTRIAL STRENGTH, COMMERCIAL CORBA ORBS FOR THE JAVA PLATFORM

by

Michelle Leigh McKeller

A graduate project submitted to the
Department of Computer and Information Sciences
in partial fulfillment of the requirements for the degree of

Master of Science in Computer and Information Sciences

UNIVERSITY OF NORTH FLORIDA
DEPARTMENT OF COMPUTER AND INFORMATION SCIENCES

Dec, 2001

The graduate project "CORBA: A Quantitative and Qualitative Comparison of Industrial Strength, Commercial CORBA ORBs for the Java Platform" submitted by Michelle Leigh McKeller in partial fulfillment of the requirements for the degree of Master of Science in Computer and Information Sciences has been

Approved by the graduate project committee: Date

**Signature Deleted** 12/13/01

Dr. Sanjay P. Ahuja
Graduate Project Advisor and Committee Chairperson

**Signature Deleted** 12/13/01

Dr. Charles N. Winton
Graduate Director

Accepted for the Department of Computer and Information Sciences:

**Signature Deleted** 12/13/01

Dr. Judith L. Solano
Chairperson of the Department

ACKNOWLEDGEMENT

I would like to especially thank Dr. Sanjay Ahuja for his instruction and support for the project.

I would also like to thank my friends and family for their support and understanding during the many hours I dedicated to achieving this milestone in my life and career.

Finally, I would also like to thank my employer, Homeside Lending, for their tuition reimbursement program and flexible work hours, which made pursuing this degree possible.

CONTENTS

FIGURES

TABLES

# ABSTRACT

In distributed systems design, middleware is a key component. Middleware establishes the communication between a client and server in a multi-tiered architecture. One approach to middleware is implementing the OMG's CORBA standard, through the use of ORBs. Three of the more popular commercially available ORBs are Sun's Java 2 ORB, Borland's VisiBroker for Java, and IONA's Orbix 2000 for Java. The purpose of this graduate project was to compare the three ORBs both quantitatively and qualitatively. The project compares the ORBs quantitatively by measuring the performance of each ORB, in terms of response time. The comparison was done qualitatively by looking at the services each ORB provides, the level of ease of implementing a simple, client-server application in each ORBs' syntax, the time taken to develop each application, difficulties encountered, and the stability of each ORB when tested. The results of the project should prove to be useful for distributed systems designers, and for researchers studying middleware products. In addition, each of the applications created for the project can be re-used for any future performance or load testing of the ORBs one might want to conduct.

# Chapter 1

# INTRODUCTION

Due to the increasing popularity of the Internet and E-Commerce, distributed systems are in widespread use today. Distributed systems provide the speed and reliability that are needed in today's high-volume, fast-paced computing world. When designing a distributed system, the question is asked, which middleware product should be used? The answer is not a simple one. With so many middleware products available, it is difficult to pick the best one to meet all of your systems needs. Purchasing a product based on an article or anecdotal experience is a poor substitute for an in-depth analysis of a product prior to purchase. However, most distributed systems designers do not have the time or materials available for an in-depth analysis of middleware products. Due to this, to aid in choosing the right middleware, a comparison was made of three of the more popular middleware products that implement the OMG's CORBA standard, through the use of ORBs. For the comparison of the middleware products each product was compared both quantitatively and qualitatively by a graduate student, who is also a full-time senior systems programmer. The quantitative comparison was done by measuring the performance, the response time for the server to respond to a client request, for each ORB. The qualitative comparison involved implementing a separate client-server application for each product. The applications were then compared by how easy each one was to implement, the amount of time it took to create each application to where it

was fully functional, the types of difficulties encountered with each product, and the stability of each product. Lastly, for the qualitative comparison, each ORB was evaluated in terms of services it provides.

The usefulness of the project should be great for any distributed systems designer, or researcher in the field of middleware. Each application created for the project lends itself to further use outside the scope of this project, for further testing of the middleware products, or with some minor revisions, for testing of another ORB product.

# Chapter 2

## DISTRIBUTED SYSTEMS

To understand the project and its goal, one must first begin with a clear understanding of distributed systems, what they are, and how they are used. The definition of a distributed system is one in which components located at networked computers communicate and coordinate their actions only by passing messages [Coulouris01]. Distributed systems have the following characteristics: concurrency of components, lack of a global clock, and independent failures of components. Concurrency of components means that multiple components in the system can run concurrently or at the same time. An example of this would be multiple servers servicing multiple client requests concurrently. Lack of a global clock means that the time of day at each server can vary. If the application you are implementing involves dates and times then your application must handle the continuity of date and time across each server or client the application is running on. By independent failures of components, we mean that if your application is running on multiple servers, one of the servers can be taken down, and the remaining servers can still function as if all servers were fully functioning. Examples of popular distributed systems are the Internet and any intranet.

The main purpose or benefit of a distributed system is the ability to share resources among many systems. The resources that can be shared are objects, hardware, software,

printers, database records, files. The World Wide Web is an example of resource sharing. With distributed systems, many challenges arise, some of which are the heterogeneity of the systems components, openness, scalability, failure handling, concurrency of components, security, and transparency [Coulouris01]. Heterogeneity of the systems components means that your distributed system must be able to handle differences among each component in your system. The differences can include hardware and software. Openness means a new component can be added to the system and it can be made available for use by a variety of clients. Scalability means the ability to continue the same level of performance and reliability under an increasing number of users. Failure handling is the ability to handle any failures that occur in the distributed system, including software and hardware failures. Transparency means that to the client or user, the distributed system is one mechanism, they are not aware of how many servers are in the system, or where they are located. With transparency, the user only knows, in order to connect to the system, the following needs to be done. The underlying architecture of the system the user is connecting to is invisible to them.

When dealing with heterogeneity in your distributed system, a component to consider including in your design is middleware.

# Chapter 3

# MIDDLEWARE

Middleware refers to a software layer that provides a programming abstraction as well as masking the heterogeneity of the underlying networks, hardware, operating systems, and programming languages [Coulouris01]. An example of a popularly used middleware is CORBA. Java RMI is another example of a middleware product, however RMI differs from CORBA in the fact that it can only be implemented in the Java programming language. The majority of middleware is implemented over IP, the Internet protocol. IP itself hides the heterogeneity of the underlying networks. However, all middleware handles the heterogeneity of operating systems and hardware.

Masking the differences of the lower layers of a distributed system's architecture is just one job of middleware. The other job of middleware is to provide a uniform programming model for system programmers to use. Middleware is represented by objects or processes in a set of computers that interact with each other to implement resource sharing and communication support for distributed applications [Coulouris01]. Middleware provides the foundation for building software components that can work together in a distributed environment. Middleware also provides a mechanism for applications to communicate with each other through the use of remote method

invocation, notification of events, replication of shared data, and communication between a group of processes.

Middleware also provides applications with services that are bound to the distributed programming model that the middleware provides [Coulouris01]. An example of this would be the many services that CORBA provides. Figure 1 shows the software and hardware service layers of a distributed system.

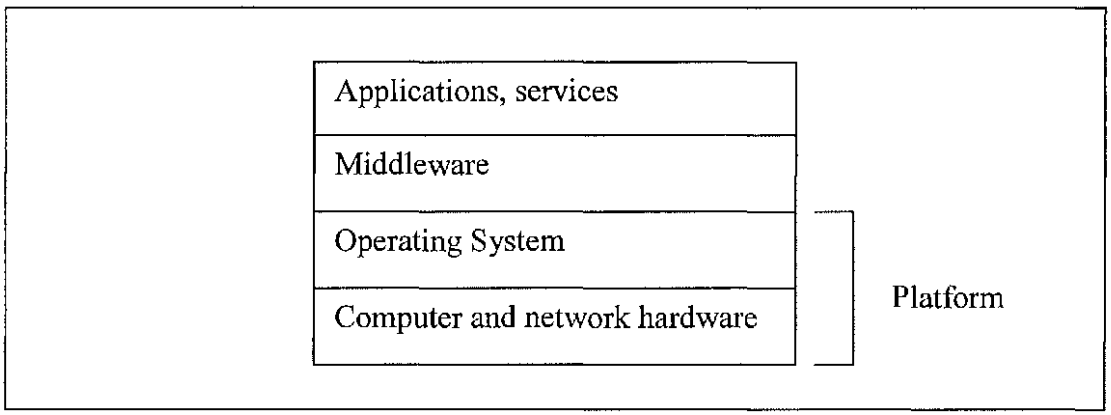



Figure 1: Software and hardware service layers in distributed systems [Coulouris01]

# Chapter 4

# CORBA

In 1989 the Object Management Group or OMG was formed with a goal in mind of creating a means of allowing distributed objects that were programmed in any language to communicate with each other [Coulouris01]. The goal was achieved when the OMG introduced a metaphor, the object request broker, commonly known as an ORB [Coulouris01]. The ORB helps an object to invoke a method on another object. The ORB handles locating the object, activating the object if needed, and communicating the object's request to the other object. The called upon method is carried out, and a response is sent back to the requesting object. Later, the OMG created a specification for an ORB architecture, this specification is CORBA.

CORBA – Common Object Request Broker Architecture is the most commonly used middleware component used today. The basics of CORBA are portability and interoperability. Portability meaning an application written to access a particular CORBA implementation or ORB, could be minimally changed to access another ORB implementation. In reality this takes more than minimal work with today's ORB products. Interoperability means that anyone can take the specification, implement their ORB, and that ORB can communicate with any ORB implementation. This concept is

referred to as IIOP or Internet Inter-ORB Protocol. IIOP is based on TCP/IP and is widely used on the Internet.

## 4.1 CORBA Services

CORBA not only provides portability and interoperability, but it also provides a wealth of distributed services to support them. The following table lists the more popular CORBA services [jGuru01]:

| Service | Description |
|---|---|
| Object Life Cycle | Defines how CORBA objects are created, removed, moved, and copied |
| Naming | Defines how CORBA objects can have friendly symbolic names |
| Events | Decouples the communication between distributed objects |
| Relationships | Provides arbitrary typed n-ary relationships between CORBA objects |
| Externalization | Coordinates the transformation of CORBA objects to and from external media |
| Transactions | Coordinates atomic access to CORBA objects |
| Concurrency Control | Provides a locking service for CORBA objects in order to ensure serializable access |
| Property | Supports the association of name-value pairs with CORBA objects |
| Trader | Supports the finding of CORBA objects based on properties describing the service offered by the object |
| Query | Supports queries on objects |

Table 1: CORBA Services

Some of the services are used more often than others and are worth a more detailed explanation.

The Naming Service allows names to be bound to remote object references of CORBA objects within naming contexts [Coulouris01]. The scope in which a set of names applies is called a naming context. Within a naming context, each name must be unique. A client calls the Naming Service to get a reference to a remote object. The client passes the name of the object to the Naming Service and the service then looks up the object reference by name, and returns the reference to the client. The client can then invoke methods on the remote object using that reference.

The Event Service is a notification service. The event service allows suppliers, or objects of interest, to communicate notifications to subscribers. The notifications are sent to the subscribers as results of a CORBA remote method invocation. Notifications can be pulled by the subscriber or pushed by the supplier. The Event Service would be useful, for example, in situations where a client received updates of new client application releases from a server. The client could automatically be notified when a new release was made available, the pushed technique, or could invoke a method on the server to see if notifications were available, the pulled technique.

The Notification Service is basically the same as the Event Service with some improvements. With the Notification Service, filters are used to either keep notifications

from being sent to all recipients of the notification, or if a client, used to keep from receiving all notifications, and only pick which ones they would like to receive.

The Transaction Service provides the capabilities to send remote method invocations as one transaction using begin, commit, and rollback commands. The client creates the transaction, the ORB then processes the transaction based on the commands at the beginning and end of the transaction.

## 4.2 CORBA Architecture

The CORBA architecture consists of several components which are the ORB core, the object adapter, skeletons, client stubs/proxies, implementation repository, interface repository, dynamic invocation interface, and dynamic skeleton interface.

The ORB Core provides an interface that includes the following [Coulouris01]:

- operations enabling it to be started and stopped
- operations to convert between remote object references and strings
- operations to provide argument lists for requests using dynamic invocation

The object adapter is the connector of the programming language interface of the corresponding servant classes and the CORBA objects with IDL interfaces. The object adapter creates the remote object references for the CORBA objects, activates objects,

and sends remote method invocations through a skeleton to the correct servant. The object adapter has two implementations the BOA, basic object adapter, and the POA, portable object adapter. The main reason for the POA is because the BOA specification was not complete, and therefore affected server portability. By using the POA, any application code that accesses one type of ORB should be able to be modified slightly and access another vendor's ORB.

Skeletons are generated when the IDL interface is compiled. Remote method invocations are invoked through a skeleton to get to a servant. The skeleton unmarshals request arguments being sent to the servant, and marshals the results of replies from the servant. Skeletons are created in the server's programming language.

Stubs/proxies are used by the client, and are created in the client's programming language. The stubs marshal the arguments in remote requests and unmarshal the results from a reply. The stubs perform the exact opposite job as the skeletons.

The implementation repository activates objects on demand. The interface repository provides registered IDL interface information to an application. The interface repository is used when, for whatever reason, a proxy is not available. The interface repository can provide the client or server with information about the methods and parameters of an object.

A dynamic invocation interface is used when for whatever reason it is not practical to allow proxies on a client machine. The dynamic invocation interface is used in combination with the interface repository to invoke a remote method on an object. If the type of an object interface is not known at compile time, then the dynamic skeleton interface can be used for remote method invocations.

## 4.3 Interface Definition Language

The IDL or Interface Definition Language, facilitates defining modules, interfaces, types, attributes and method signatures for an application [Coulouris01]. The IDL is created at the beginning of programming an application. Once completed, the IDL is compiled with the compiler included with your ORB's implementation. The commands to invoke the compiler differ among vendor products. The IDL is programming language neutral, by providing language bindings for most of the commonly used programming languages. By this, a client/server can be written in different languages, but still communicate with each other.

# Chapter 5

# ORBS

The OMG created the CORBA specification to allow for communication across any platform and any programming language. The specification when implemented is referred to as an ORB. The specification can be implemented by anyone willing to try. However the time it would take to implement your own ORB seems wasteful, when so many good ORB implementations are available to the public. Three of the more popular ORBs available are Sun's Java 2 ORB, Borland's VisiBroker for Java, and IONA's Orbix 2000. Although the CORBA specification's purpose was to provide uniformity and hence portability among applications, each of these ORBs are very different from the other. The following discussion will highlight those differences.

## 5.1 Java 2 ORB

The Java 2 ORB is one of the more appealing implementations of the CORBA specification. The ease of use of the Java 2 ORB is outstanding compared to other orbs. Sun has provided several tutorials, articles, and an extensive knowledge base on how to use the ORB. With the current version of the JDK, 1.3, the Java 2 ORB provides a Naming Service only. The Java 2 ORB differs from other ORBs in the fact that it does

not include a distinct basic object adapter (BOA) or a portable object adapter (POA). The object adapter manages the creation and lifecycle of objects in the CORBA distributed space [DEV01]. Since the Java 2 ORB is without an object adapter, the ORB handles this task itself. Also, the object adapter provides an API that object implementations use for various low level services [jGuru01]. Other services not implemented in the current version of the Java 2 ORB are the interface repository, policies and methods for getting them, domain managers and methods for getting them, and ORB methods for supporting single-threading [SUN].

The Java 2 ORB is only partially compliant with the CORBA 2.x specification. A note worth mentioning, the newest beta version of the JDK, 1.4, includes support for the POA through the Java 2 ORB and is more compatible with the CORBA specification.

## 5.2 VisiBroker

Since the Java 2 ORB's list of available services and features is so short, the next two sections will seem extremely long in comparison.

One of the main features of VisiBroker is the extensive amount of services it provides. One of those features is the Smart Agent Service, which is a directory service that provides the ability for multiple smart agents on a network to cooperate to provide high availability and load balancing for client access to the server objects [VISI].

An additional feature of VisiBroker is the Object Activation Daemon (OAD). The OAD is used to automatically start object implementations when a client needs to use them, and can defer activation of an object until a client request [VISI]. The OAD is the interface to the implementation repository. With the current version of VisiBroker, activation is done through POAs, previous versions used BOAs, and the current version is backward compatible.

VisiBroker provides a Location Service, which enables you to access information for an object from multiple Smart Agents. While working with the Smart Agents, the Location Service can tell you all of the available instances of an object.

VisiBroker provides for single and multithreaded thread management. VisiBroker's connection management minimizes the number of client connections to the server by multiplexing all requests for the same object over the same connection.

One nice feature of VisiBroker is that it comes equipped with two IDL compilers, one functions as any normal IDL compiler generating skeletons and stubs, the other skips generating skeletons and stubs and simply populates the interface repository with the contents of the IDL file being compiled. Given this, the dynamic skeleton interface and the dynamic invocation interface are also standard features of VisiBroker.

An additional feature are interceptors that can be used to extend the ORB with customized client and server code that enables load balancing, monitoring, or security to meet specialized needs of distributed applications [VISI].

One point worth mentioning again is the fact that the 4.1 release of VisiBroker is backward compatible with previous versions. By this, a previous implementation of VisiBroker that implemented the basic object adapter (BOA), can still be used. One ORB that does not provide support for the BOA is Orbix 2000. The fact that VisiBroker does still support the BOA is a positive fact for VisiBroker. Finally, VisiBroker is easily used, with a nice amount of user-friendly documentation available at Borland's web-site.

## 5.3 Orbix 2000

VisiBroker provides a lot of services. Orbix 2000 provides all of these services as well. For example, Orbix 2000 has services similar to the Smart Agent and Object Activation Daemon. They are all bundled under a section called Location Domains, which includes the Location Service. Orbix also has the capability to insert the contents of an IDL interface into the interface repository, but instead of creating a separate IDL compiler to accomplish this, Orbix 2000 simply has the functionality as a flag setting for the standard IDL compiler. Therefore, to simplify matters, the following discussion will only mention what Orbix 2000 has that VisiBroker does not.

Orbix comes bundled with code generators referred to as genies. The genies allow a developer to create an entire client-server, multithreaded, fully functional application in minutes. The documentation for Orbix 2000 is geared towards the application developer using the genies. Therefore, implementation of an application using Orbix 2000 is faster when using the genies instead of venturing on your own.

Orbix 2000 comes equipped with an object transaction service, which is one of the CORBA services mentioned in the specification. The transaction service allows for remote method invocations to be processed in bundles, called transactions. The transaction is controlled through commit and rollback commands. If any portion of the transaction fails, a rollback command is issued, and none of the transaction is committed.

One note to make is that both VisiBroker and Orbix 2000 come standard with the more common CORBA services, such as the naming and event services.

# Chapter 6

# PROJECT DESCRIPTION

Since the purpose of the project was a comparison of the products, emphasis was placed on the data gathered, rather than the applications created for the project. However, the applications created can be re-used for further testing of the ORB's or with some minor revisions, to test another ORB product.

The project was to compare three of the more popular, commercially available ORB products both quantitatively and qualitatively using the Java programming language. The quantitative aspect focused on the performance of each ORB. The qualitative aspect focused on the ease of implementing each ORB, the time involved, any difficulties that occurred, the stability of each ORB, and the services each ORB provides. To compare each of the ORBs, a fully functional client-server application was created for each one of the products. The application was a three-tiered client-server application. The first tier was the client, the second tier was the server, and the third tier was an Oracle database. Each tier ran on a separate server machine. The servers were named DSP, Neptune, and Manatee. DSP is a single processor Pentium 200 MHz machine with 64 megabytes of RAM, running Red Hat Linux 7 with kernel version 2.2. Neptune is a dual Pentium III 550 MHz processor machine with 512 megabytes of RAM. Manatee is running Oracle 8 Enterprise Edition with the Partitioning and Objects Options, Release 8.0.5.0.0.

## 6.1 Scenario

The original proposal for the project consisted of a GUI for a client with web capabilities. Since the focus of the project was the end result, which was the data, not the application's usefulness, the GUI was discarded. Another stipulation on the project was to automate the test cases, in hopes of reusability of the applications created. Due to these stipulations some improvisation was used.

Assume you are a clerk for a local part's store. Everyday you sell parts to customers and record this information into a spreadsheet, like Microsoft Excel. At the end of the day, it is your responsibility to convert the spreadsheets into text files, and drop them into the correct directory on your networked Linux machine. During the day, you basically have two types of transactions, an account transaction, which consists of adding new accounts, and updating current accounts. Also, you might need to find out what information is currently stored for an existing account. As the clerk, you create three spreadsheets, called Read.txt, Update.txt, and InsertAcct.txt. Also, you have customers that make payments on their accounts during the day, referred to as transactions. Transactions can be added, updated, or read for whatever reason. At the end of the day, you create three more text files from spreadsheets calling them Read.txt, UpdateTrans.txt, and Insert.txt. The Read.txt is the same for both the account and transaction because we can only read the last transaction made for an account. Therefore, all that is needed is the account number to read an account, or read the last transaction made for an account.

## 6.2 Database and Connectivity

An Oracle database was used to store the account and transaction information that is kept in the text files the store clerk submits at the end of each business day. The database schema for the project consists of six tables named: Accounts, Transactions, Transaction_Types, Account_Statuses, Account_Types and States. Refer to Appendix A: Database Tables Creation Script for further details of each table. The Transaction_Types, Account_Statuses, Account_Types, and States tables are code lookup tables for the Accounts and Transactions tables. The Accounts table consists of 25 fields, that when their lengths are summed equals 840. The Transactions table consists of 5 fields, that when their lengths are summed equals 58. A record being either read, inserted, or updated on the Accounts table is supposed to mimic a heavy load being sent across a network. A record being either read, inserted, or updated on the Transactions table is supposed to mimic a light load being sent across a network. To increase speed at the database, for the transaction id field (trans_id) a sequence was created to automatically create the next available transaction id number.

JDBC – Java Database Connectivity was used for the database connectivity. JDBC was a perfect match since each application was being implemented in Java. To minimize time at the server, only one database connection was created. The database connection was located in the server class. The server class was the main program on the server side, and was only invoked at start up by the developer. The server itself has no methods called on by the client.

## 6.3 Services

Each ORB's implementation had a different way of invoking services that have to be running prior to starting your server and client. The commands are listed in Appendix B. For the Java 2 ORB and VisiBroker, a simple naming service was the only service that had to be manually started. For Orbix 2000, a series of services are all ran as background daemons on the Linux machine. The Orbix 2000 services are all started and stopped with a script that is automatically created for you when configuring the Orbix 2000 installation. The Orbix 2000 services must be started prior to running the server. However, these services are configured to where only one user-id has control over these services. The default user-id is 'root', however you can override this during the installation configuration. The current user-id with access to start and stop these services is 'mmckeller'. The services are currently running on Neptune and DSP.
One final note, files were created that can be sourced in order to run the services for the Java 2 ORB and VisiBroker.

## 6.4 Servers

Each application's server required different command-line arguments and different syntax for them to run. Appendix C includes these commands. Recall that a separate application was required for each ORB product's implementation, according to the OMG CORBA specification this should really not have been required. According to the standard, portability allows for a few simple lines of code to be changed and the same

application should be able to invoke any of the three ORB implementations. If each of the ORBs supported the same services, and implemented the same object adapter, then this might be possible. However, the Java 2 ORB does not implement a BOA or POA, VisiBroker implements both, and Orbix 2000 only implements the POA. Since Java's implementation of their ORB most closely resembles the BOA implementation of VisiBroker, the decision was made to implement VisiBroker in the BOA style rather than the POA. The only difference between the BOA and POA are a few low-level additional services that the POA offers that were not needed in any of the applications.

If all three of the ORBs supported the POA, then ideally the only two things that would need to change between each application, are two settings, the ORBClass properties and the ORBSingletonClass properties. The properties can be set four different ways, refer to Appendix D for further details. In order not to invoke the incorrect ORB, the only way to set the properties, for use in this project, was to set the properties on the java command line. For visibroker, the properties were wrapped in the 'vbj' command that is used to invoke a server, instead of using the java command. One final note, files were created that can be sourced in order to run the server for the Java 2 ORB and VisiBroker.

## 6.5 Factory Classes

In order to limit the number of activated objects at the server, a factory class was used, that could be called by the client. A factory class is a wrapper type object that

encapsulates other objects within it. By doing this, the only object that is registered through the naming service is the factory class, which for the project was called "Manager", and this is the only object that has to be looked up in the naming service.

## 6.6 Clients

Due to each ORBs individualized syntax, each client was invoked with different command line arguments. The command to invoke each client is listed in Appendix E. One of the command line arguments, for each client, was the name of the test file you wanted to process. Another is the number of clients you wanted to run concurrently, and also what type of transaction you wished to run, either an insert, update, or read on an account or transaction. The type of transaction and name of file were separated so that the filenames could change without changing the application code.

To mimic several clients running concurrently, each test file consisted of 200 records. A for..loop was used in the client to read in one record per client to be ran. A new thread was spawned by the client for each client to be run. The record that was to be sent to the database was broken down into an array and the array was passed to the client thread. The client thread then took the array passed it to the remote object, the remote object performed the database update, returning any result sets or error messages to the client. Due to the design of the applications, greater numbers of clients could be tested by simply adding more test data to the test files, and changing the number of clients on the

command line. The only limit is the ORB itself. One final note, files were created that can be sourced in order to run the client for the Java 2 ORB and VisiBroker. Instructions on how to compile and run the client-server for each application are included in Appendix F.

# Chapter 7

# TESTING

Performance testing was implemented by recording the response time of each server to a client's request. The start time was recorded when the client called the remote method for an object. The stop time was recorded at the next line in the code following the remote method call. The difference of the start and stop times gives the response time.

Increasing numbers of clients were run concurrently, and separate tests ran for each increment. The number of clients began with 1 going to 200 with increments of 25. Also, a separate test was ran for an insert, update, and read of an account record, which mimicked a heavy load, and for an insert, update, and read of a transaction record, which mimicked a light load. The sum of all tests equals 54 separate tests for each ORB product. The total number of tests ran were 162 tests.

The response times were recorded in log files, and the mean, max, min, and variance for each orb under the conditions listed above were calculated and recorded in tabular and graphical forms. The graphs are Appendix G, H, I, J.

During testing, if a request failed for whatever reason, that request was omitted from the response time calculations.

# Chapter 8

# RESULTS AND COMPARISONS

Each ORB performed well and was fairly easy to implement. No one ORB clearly stood out from the rest as being the better ORB in every category. Each ORB performed better or worse depending on the test ran. The results are in the following sections.

## 8.1 Java 2 ORB

### 8.1.1 Quantitative Comparison

The Java 2 ORB had the slowest response times under heavy conditions, and tied with Orbix 2000 under light conditions.

The Java 2 ORB had a higher variance than Orbix 2000, but it's variance almost equaled VisiBroker's variance. The Java 2 ORB had a higher maximum response time than the other ORBs, a minimum response time higher than VisiBroker, but lower than Orbix, an average response time the same as Orbix 2000 for light loads, and greater than the other two ORBs under heavy loads.

The reason for this is unknown, basically because we are not aware of how the ORB is built by Java. An educated guess would be that when this version of the JDK was created, CORBA was just beginning to become popular, and not a whole lot was known about it. The VisiBroker and Orbix 2000 ORBs are fairly new products, thereby having access to the latest CORBA specifics.

### 8.1.2 Qualitative Comparison

The Java 2 ORB is a great ORB product. The ease of use of the Java 2 ORB outweighs its lack of services. Java provides nice tutorials at their web-site. Several books have been published with examples of using the Java 2 ORB, also sometimes referred to as the Java IDL. For a simple, client-server application that needs to implement CORBA, the Java 2 ORB is definitely the right choice. For a large, multi-user, heavy data, production environment, the ORB is not ideal. The ORB lacks too many of the common CORBA services. The only real service of use it provides is the Naming Service. Another downfall is the IDL compiler that comes with the implementation can only generate Java code. VisiBroker and Orbix 2000 both come with C++ versions of their ORB product. Another disadvantage to using the Java ORB is that it offers no support for a single-threaded model, only multi-threaded. The disadvantage is small, but could become an issue when implementing an application using it.

## 8.2 VisiBroker

### 8.2.1 Quantitative Comparison

Under light load conditions, VisiBroker outperformed the Java 2 ORB and Orbix 2000. However under heavy conditions, VisiBroker's response times were so similar to Orbix 2000's that it was really a tie between the two on who was the fastest.

VisiBroker's variance almost equaled the Java 2 ORB's, but was higher than Orbix 2000's. VisiBroker had a minimum response time lower than Orbix 2000, a maximum response time lower than Orbix 2000 for light loads, a higher maximum response time than Orbix 2000 for heavy loads, average response times lower than Orbix 2000 for light loads, and average response times almost equal to Orbix 2000 for heavy loads.

Overall VisiBroker seemed to have the better times. The only reason that could be found for this, is the response time seemed to be quicker for a client request when it followed a failed client request call. When a request failed, it was not included in the calculations for the response times.

### 8.2.2 Qualitative Comparison

VisiBroker is an easy ORB product to use when creating client-server applications. VisiBroker provides a wealth of services, almost all of the CORBA services that are available, with the exception of the Transaction service. One advantage that VisiBroker clearly has over ORBIX 2000 is the fact that it supports the BOA still. Backward compatibility is always a key decision when choosing any software product. The fact that VisiBroker is backward compatible is definitely a feather in its cap.

## 8.3 Orbix 2000

### 8.3.1 Quantitative Comparison

Under heavy load conditions, Orbix 2000 tied with VisiBroker for quickest response times. Under light loads, Orbix 2000 tied with the Java 2 ORB for the slowest reponse times.

Orbix 2000 had the lowest variance in response times over the other two ORBs. Orbix 2000 had the highest minimum response times over the other two ORBs, tied with the Java 2 ORB for the highest maximum response time under light loads, the lowest

maximum response time for heavy loads, an average response time higher than VisiBroker, but the same as Java for light loads, and an average response time the same as VisiBroker, but lower than Java for heavy loads.

Orbix 2000's response times were evenly distributed, meaning the first client request serviced had almost the same response time as the last client request serviced, which is good. Even distribution among response times means the load balancing in Orbix is working properly, and as the number of clients continues to increase to numbers greater than those used for this project, the response times should stay even across multiple clients.

### 8.3.2 Qualitative Comparison

Orbix 2000 provides more services than the other two ORBs used in the comparison. Orbix 2000 provides a transaction service that neither ORB has. Orbix 2000 also provides code generators, which can substantially speed up the learning curve and development time for an application. The lack of support for the BOA is a major downfall with Orbix 2000. Due to this, there is no backward compatibility with previous Orbix versions, and there is no portability with code that implements the BOA. Another downfall of the Orbix 2000 is its lack of good documentation. Orbix has the standard manuals that all software products have. However, the manuals lack good examples for the beginning Orbix 2000 programmer. The examples are either too simplistic, by not

implementing a naming service, or are too advanced, by including every service, or feature available. Administration with Orbix is a challenge itself, but is outside the realm of this comparison. Refer to Appendix K for further details on how to properly configure Orbix for your installation. Refer to Appendix L for the source code of the Java 2 ORB application. Refer to Appendix M for the source code of the VisiBroker ORB application. Refer to Appendix N for the source code of the Orbix 2000 application. Refer to Appendix O for a directory structure of the project CD, which includes a copy of this paper.

8.4 Additional Comparisons

The following table lists the remaining components of the comparison of the three ORBs:

| Feature | Java 2 ORB | VisiBroker | Orbix 2000 |
|---|---|---|---|
| Supports single and multi-threading | Multi-thread support only | Single and Multi | Single and Multi |
| Naming Service | Yes | Yes | Yes |
| Event Service | No | Yes | Yes |
| IDL supports code generation for multiple programming languages | No | Yes | Yes |
| BOA Support | No | Yes | No |
| POA Support | Not in current version | Yes | Yes |
| Policy Support | No | Yes | Yes |
| Location Service | No | Yes | Yes |
| Activator Daemon | No | Yes | Yes |
| Interface Repository | No | Yes | Yes |
| Transaction Service | No | No | Yes |
| Dyn-Any | No | Yes | Yes |
| Code Generator | No | No | Yes |
| Time to develop | 8 hrs | 10 hrs | 14 hrs |
| Time to configure | 0 | 0 | 40 hrs |
| Stability of Code | Stable | Stable | Stable |
| Ease of Implementation | Easy | Not as Easy | Difficult |
| Difficulties Encountered | Conflict with other ORBs, fixed when ORBClass property set to use Java 2 ORB. | Difficulty generating BOA support classes using the IDL compiler. Fixed when set flag for compiler to generate the classes. | Difficulty following code examples, using code generating genies, installing and configuring ORBIX, and configuring the repository to work across a network |

Table 2: Additional Comparison Results

# Chapter 9

# CONCLUSIONS

In conclusion the best ORB to use in your distributed system is the ORB that meets your applications needs the best, however here are a few recommendations.

For simple, client-server applications, with a low number of concurrent clients running, the Java 2 ORB works best.

For large, heavy data applications where no transaction processing is necessary, VisiBroker works best.

For large, heavy data applications where transaction processing is needed, and the time for development is short, then the Orbix 2000 ORB works best.

For the migrating of existing, older CORBA implementations to a new ORB, VisiBroker would be best due to its support for both the BOA and POA.

# REFERENCES


[Coulouris01]
Coulouris, George, J. Dollimore, and T. Kindberg, Distributed Systems Concepts and Design, Addison-Wesley, 2001

[DEV01]
Sun – Advanced Programming for the Java 2 Platform, http://developer.java.sun.com/developer/onlineTraining/Programming/JDCBook/corba.html

[jGuru01]
Sun – Introduction to CORBA Short Course, http://developer.java.sun.com/developer/onlineTraining/corba/corba.html

[SUN]
Package org.omg.CORBA, http://java.sun.com/products/jdk/1.2/docs/api/org/omg/CORBA/package-summary.html#unimpl

[VISI]
VisiBroker documentation version 4.0, http://www.borland.com/techpubs/books/vbj/vbj40/framesetindex.html

APPENDIX A

Database Tables Creation Script

```
REM
REM Michelle McKeller
REM Tables for Account Database
REM
REM TABLE
REM     ACCOUNTS
REM     TRANSACTIONS
REM     TRANSACTION_TYPES
REM     STATES
REM     ACCOUNT_STATUSES
REM     ACCOUNT_TYPES
REM
PROMPT
PROMPT Creating Table ACCOUNTS
CREATE TABLE accounts(
 acct_num                VARCHAR2(9)        NOT NULL,
 pin_num                 VARCHAR2(4)             NOT NULL,
 acct_status_cd          VARCHAR2(4)             NOT NULL,
 acct_type_cd            VARCHAR2(4)             NOT NULL,
 ssn                     VARCHAR2(9)             NOT NULL,
 last_name               VARCHAR2(30)            NOT NULL,
 first_name              VARCHAR2(25)            NOT NULL,
 middle_initial          VARCHAR2(1)             NULL,
 prim_address            VARCHAR2(30)            NOT NULL,
 prim_city               VARCHAR2(30)            NOT NULL,
 prim_state_cd           VARCHAR2(2)             NOT NULL,
 prim_zip_cd             VARCHAR2(5)             NOT NULL,
 prim_zip_cd_ext         VARCHAR2(4)             NULL,
 sec_address             VARCHAR2(30)            NULL,
 sec_city                VARCHAR2(30)            NULL,
 sec_state_cd            VARCHAR2(2)             NULL,
 sec_zip_cd              VARCHAR2(5)             NULL,
 sec_zip_cd_ext          VARCHAR2(4)             NULL,
 prim_phone_num          VARCHAR2(20)            NOT NULL,
 sec_phone_num           VARCHAR2(20)            NULL,
 fax_num                 VARCHAR2(20)            NULL,
 create_dt               VARCHAR2(20)            NOT NULL,
 modified_dt             VARCHAR2(20)            NOT NULL,
```

```
 comments                VARCHAR2(256)          NULL,
 addtl_comments          VARCHAR2(256)          NULL
)
;
REM   TRANSACTIONS
PROMPT
PROMPT Creating Table TRANSACTIONS
CREATE TABLE transactions(
 trans_id                VARCHAR2(9)            NOT NULL,
 acct_num                VARCHAR2(9)            NOT NULL,
 trans_type_cd           VARCHAR2(4)            NOT NULL,
 trans_amt               NUMBER(14,2)           NOT NULL,
 create_dt               VARCHAR2(20)           NOT NULL
)
;
REM    TRANSACTION_TYPES
PROMPT
PROMPT Creating Table TRANSACTION_TYPES
CREATE TABLE transaction_types(
 trans_type_cd           VARCHAR2(4)            NOT NULL,
 trans_type_name         VARCHAR2(20)           NOT NULL
)
;
REM   STATES
PROMPT
PROMPT Creating Table STATES
CREATE TABLE states(
 state_cd                VARCHAR2(2)            NOT NULL,
 state_name              VARCHAR2(20)           NOT NULL
)
;
REM   ACCOUNT_STATUSES
PROMPT
PROMPT Creating Table ACCOUNT_STATUSES
CREATE TABLE account_statuses(
 acct_status_cd          VARCHAR2(4)            NOT NULL,
 acct_status_name        VARCHAR2(20)           NOT NULL
)
;
REM   ACCOUNT_TYPES
PROMPT
PROMPT Creating Table ACCOUNT_TYPES
CREATE TABLE account_types(
 acct_type_cd            VARCHAR2(4)            NOT NULL,
 acct_type_name          VARCHAR2(20)           NOT NULL
);
```

# APPENDIX B

## Commands to Start the Services

For the Java 2 ORB:
tnameserv -ORBInitialPort 2400

For VisiBroker:
osagent

For Orbix 2000:
source start_orbix2000_services

# APPENDIX C

## Commands to Start the Servers

For the Java 2 ORB:
java CBServer -ORBClass com.sun.CORBA.iiop.ORB -ORBSingletonClass com.sun.CORBA.iiop.ORB -ORBInitialPort 2400 http://neptune.cocse.unf.edu/mmckeller/Corba/CBServer oracle.jdbc.driver.OracleDriver jdbc:oracle:thin:@manatee.unf.edu:1521:sid1 youruseridhere yourpasswordhere

For VisiBroker:
vbj VBServer http://neptune.cocse.unf.edu:2400/mmckeller/VisiBroker/VBServer oracle.jdbc.driver.OracleDriver jdbc:oracle:thin:@manatee.unf.edu:1521:sid1 youruseridhere yourpasswordhere

For Orbix 2000:

```
<project name="generated_app" default="build_all" basedir=".">

 <property name="idl_flags" value="-I/opt/iona/orbix_art/1.2/idl -jbase=-POBBankObjects:-Ojava_output -
jpoa=-POBBankObjects:-Ojava_output "/>
 <property name="clpath"
value="/usr/local/jdk1.3/lib/classes111.zip:$ORACLE_HOME/jdbc/lib/classes111.zip:/etc/opt/iona/domain
s:/opt/iona/orbix_art/1.2/classes/orbix2000.jar:/opt/iona/orbix_art/1.2/classes/omg.jar:/etc/opt/iona:/etc/opt/
iona/domains:/opt/iona/orbix_art/1.2/classes/orbix2000.jar:/opt/iona/orbix_art/1.2/classes/omg.jar:/etc/opt/i
ona:/etc/opt/iona:/etc/opt/iona/domains:export:/opt/iona/orbix_art/1.2/demos/classes:/etc/opt/iona:/etc/opt/i
ona/domains"/>
 <target name="init">
  <tstamp/>
  <property name="classes" value="classes"/>
  <mkdir dir="${classes}"/>
 </target>

 <target name="idl_compile" depends="init">
  <exec dir="./" command="/opt/iona/orbix_art/1.2/bin/idl ${idl_flags} OBBank.idl"
output="idl_compiler.out"/>
  <exec dir="./" command="cat idl_compiler.out"/>
 </target>

 <target name="build_all" depends="idl_compile">
 <javac classpath="./classes:${clpath}"
       srcdir="./" destdir="./classes"/>
 </target>

 <target name="runserver" depends="">
  <java classpath="./classes:${clpath}"
```

```
        jvmargs="-Dorg.omg.CORBA.ORBClass=com.iona.corba.art.artimpl.ORBImpl -
Dorg.omg.CORBA.ORBSingletonClass=com.iona.corba.art.artimpl.ORBSingleton"
     args="-ORBdomain_name default-domain
http://neptune.cocse.unf.edu:2400/mmckeller/Orbix/OBBankObjects/OBBankObjects/server
oracle.jdbc.driver.OracleDriver jdbc:oracle:thin:@manatee.unf.edu:1521:sid1 youruseridhere
yourpasswordhere"
        fork="yes"
        classname="OBBankObjects.server"/>
 </target>

 <target name="runclient" depends="">
  <java classpath="./classes:${clpath}"
        jvmargs="-Dorg.omg.CORBA.ORBClass=com.iona.corba.art.artimpl.ORBImpl -
Dorg.omg.CORBA.ORBSingletonClass=com.iona.corba.art.artimpl.ORBSingleton"
     args="-ORBdomain_name default-domain READ 25 Read.txt
http://neptune.cocse.unf.edu:2400/mmckeller/Orbix/OBBankObjects/OBBankObjects/server"
        fork="yes"
        classname="OBBankObjects.client"/>
 </target>

 <target name="info" depends="">
  <echo message=" help:" />
  <echo message="build.xml options:"/>
  <echo message=""/>
  <echo message="info      Prints out this message." />
  <echo message="build_all  Deletes class files, IDL compiler generated files"/>
  <echo message="and rebuilds everything."/>
  <echo message="clean     Removes all class files."/>
  <echo message="clean_all  Removes all generated files."/>
  <echo message="runserver  Run the server.   "/>
  <echo message="runclient  Run the client."/>
 </target>



 <target name="clean" depends="">
  <deltree dir="classes"/>
  <delete dir="./" includes="idl_compiler.out,ant_env.csh ant_env.sh,*.ref"/>
 </target>
 <target name="clean_all" depends="">
  <deltree dir="OBBankObjects"/>
  <deltree dir="classes"/>
  <deltree dir="java_output"/>
  <delete dir="./" includes="idl_compiler.out,ant_env.csh ant_env.sh,*.ref"/>
  <deltree dir="idlgen"/>
  <delete dir="./" includes="./*.ref"/>
  <delete dir="./" includes="build.xml"/>
  <delete dir="./" includes="idl_compiler.out"/>
 </target>

</project>
```

APPENDIX D

Setting the properties

1. Set them in the orb.properties file in the jdk_installation_dir\jre\lib directory. Note: if there are multiple ORB vendors products running on the same machine, this file will cause a conflict among the ORBs. You could be using a different ORB than you intended to with this property files existence.

2. If there is another ORB installation already using the orb.properties file, the properties may be overridden on the command line as follows:

   Java –Dorg.omg.CORBA.ORBClass=name of your orb class – Dorg.omg.CORBA.ORBSingletonClass=name of your orb's singleton class

3. Use the wrapper that comes with your ORB's implementation, for VisiBroker it's the vbj and vbjc commands, for Orbix 2000 it's the it_java and it_javac commands.

4. Set programmatically: this only applies to the org.omg.CORBA.ORBClass as the search for the org.omg.CORBA.ORBSingletonClass is implemented in a static initializer on the ORB class. This is executed once in the JVM's lifetime when the ORB class is loaded. Therefore, this value cannot be set programmatically.

The following is the order in which the ORB class searches for these properties:

For org.omg.CORBA.ORBClass:
1. Check in Applet parameter or application string array, if any.
2. Check in the properties parameter, if any.
3. Check in the System properties.
4. Check in the orb.properties file located in the jdk_installation_dir\jre\lib directory.
5. Fall back on a hard-coded default behavior.

For org.omg.CORBA.ORBSingletonClass:
1. Check in Applet parameter or application string array, if any.
2. Check in the properties parameter, if any.
3. Check in the System properties.
4. Check in the orb.properties file located in the jdk_installation_dir\jre\lib directory.

# APPENDIX E

## Commands to Start the Clients

For the Java 2 ORB:
java CBClient -ORBClass com.sun.CORBA.iiop.ORB -ORBSingletonClass com.sun.CORBA.iiop.ORB -ORBInitialHost neptune.cocse.unf.edu -ORBInitialPort 2400 READ 25 Read.txt http://neptune.cocse.unf.edu/mmckeller/Corba/CBServer

For VisiBroker:
vbj VBClient READ 25 Read.txt
http://neptune.cocse.unf.edu:2400/mmckeller/VisiBroker/VBServer

For Orbix 2000:
Refer to Appendix C: For Orbix 2000

APPENDIX F

Compiling and Running the Applications

To begin verify that the .bash_profile file included on the project CD is set as your .bash_profile on the server. The settings in it point to the correct JDK directory to use (1.3) and to the correct JDBC driver to use.

For the Java 2 ORB:

1. Run the idl compiler: **idlj –fall CBBank.idl**
2. Run in both the CBBankObjects directory and the regular directory: **javac *.java**
3. To start the naming service, in a separate secure shell session on the server machine, **source svrcbnamesvr**
4. To start the server, **source svrcb**
5. To start the client, **source clcbclient**
6. Make sure the file name your testing with is the same as the file name in clcbreadacct, also modify the number of clients to be what you want to test with in the same file. Separate scripts were created for each type of test file.

For VisiBroker:

1. To run the idl compiler, first the .module class must be removed, then run the makefile by typing, **make**
2. To compile the code, all class files must first be removed, then run the makefile by typing, **make**
3. To start the naming service, in a separate secure shell session on the server machine, **source svrvbnamesvr**
4. To start the server, **source svrvb**
5. To start the client, **source clvbclient**
6. Make sure the file name your testing with is the same as the file name in clvbreadacct, also modify the number of clients to be what you want to test with in the same file. Separate scripts were created for each type of test file.

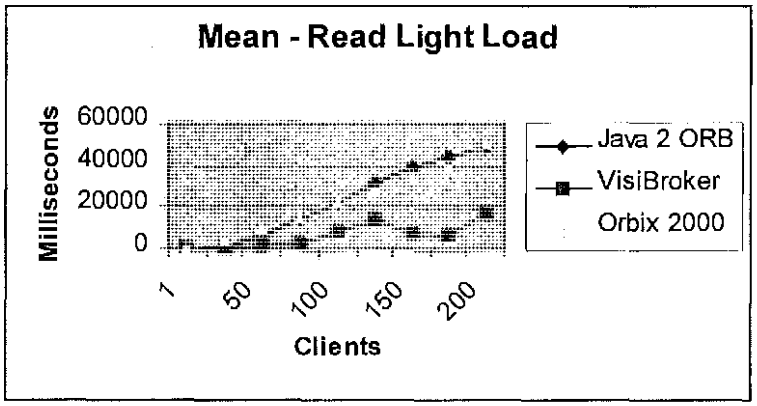
For Orbix 2000:

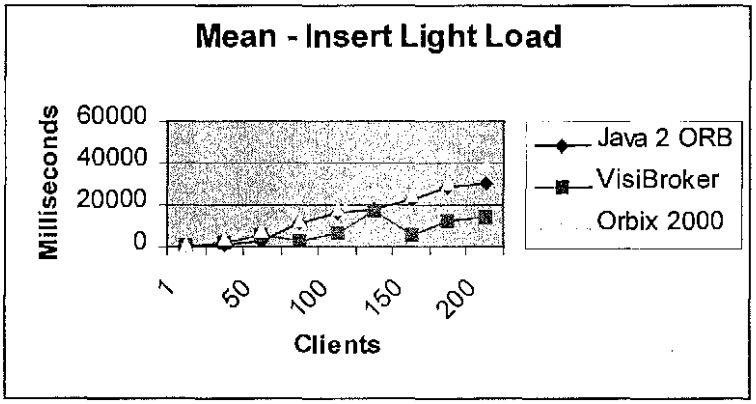
1. Keep the directory structure the same as it is on the project CD.
2. **source O2KEnv.sh**
3. **source /opt/iona/bin/orbix2000_env**
4. **source /opt/iona/bin/orbix2000_java_env**
5. **source ant_env.sh**
6. **start_orbix2000_services**

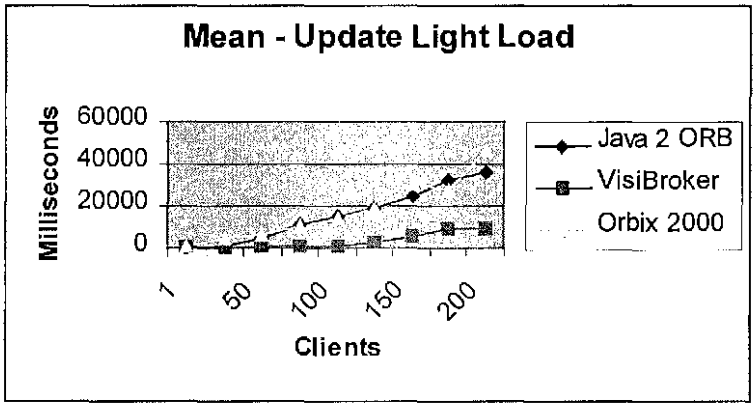
7. To compile the code completely, **ant clean** then **ant build_all**
8. To start the server, **ant runserver**
9. To start the client, **ant runclient**
10. To test different test files, separate .xml documents were created, you need only rename the .xml document to build.xml to use it.

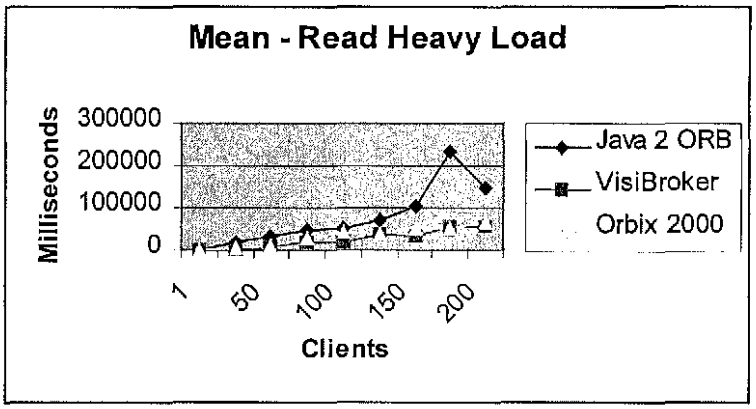
APPENDIX G

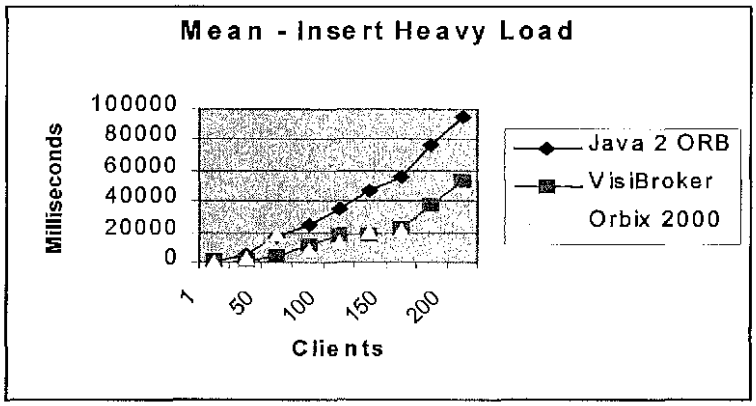
Mean of Response Times

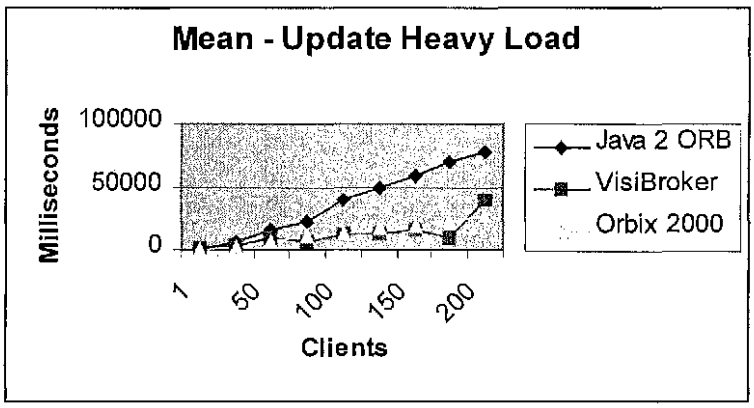


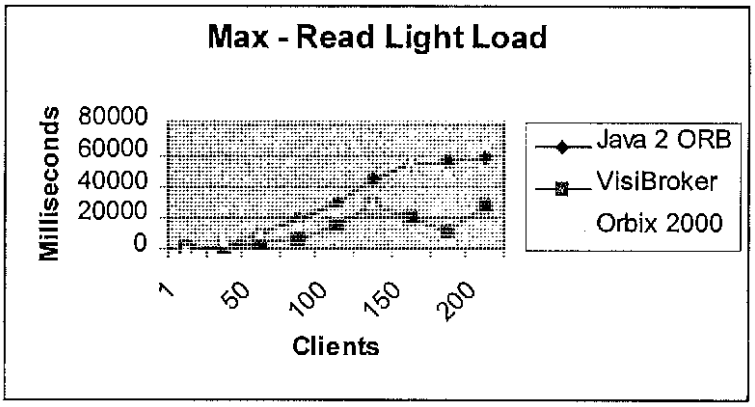

Mean - Read Heavy Load
Milliseconds
300000
200000
100000
0
1
50
100
150
200
Clients
Java 2 ORB
VisiBroker
Orbix 2000

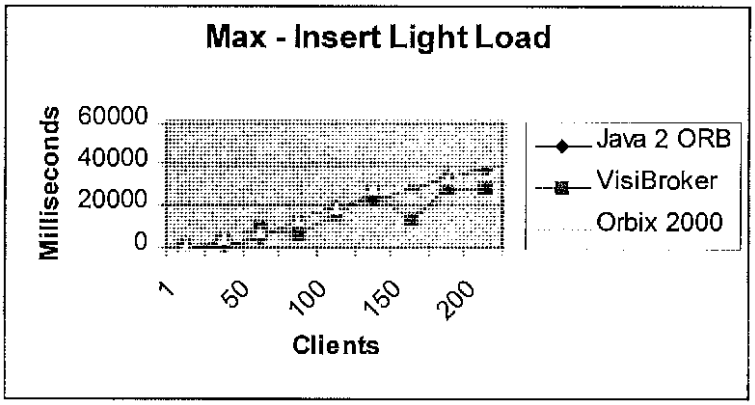

Mean - Insert Heavy Load
Milliseconds
100000
80000
60000
40000
20000
0
1
50
100
150
200
Clients
Java 2 ORB
VisiBroker
Orbix 2000

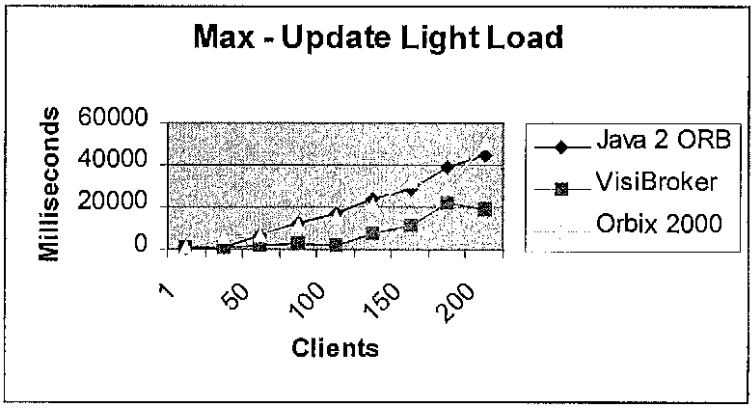

Mean - Update Heavy Load
Milliseconds
100000
50000
0
1
50
100
150
200
Clients
Java 2 ORB
VisiBroker
Orbix 2000

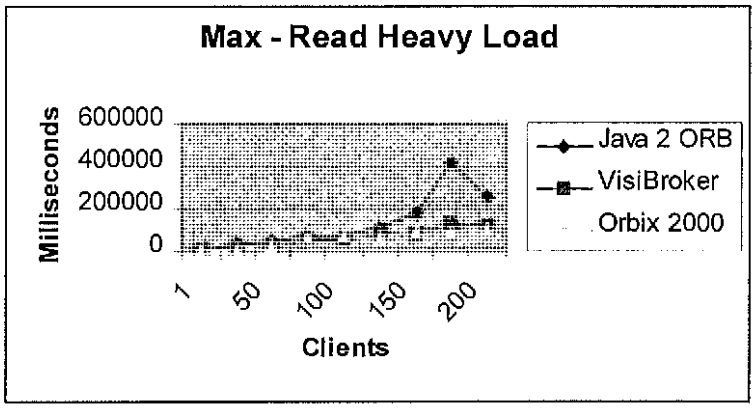
APPENDIX H

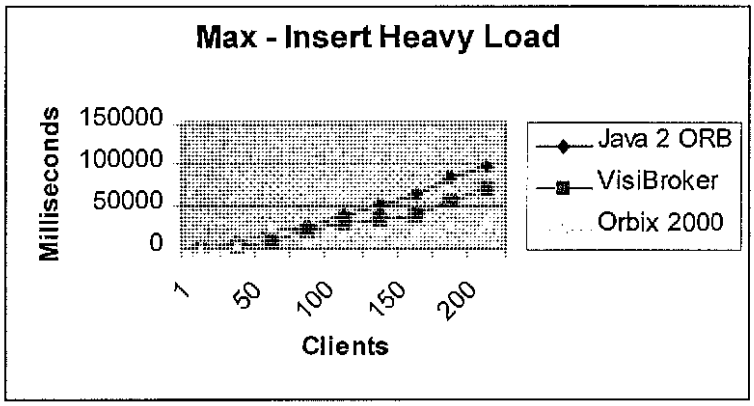
Maximum of Response Times

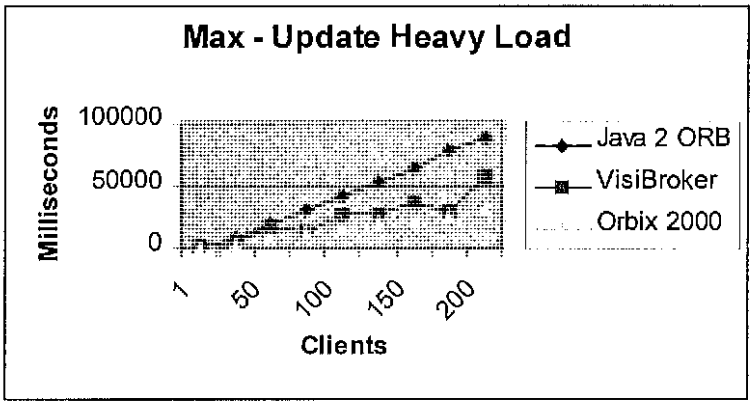


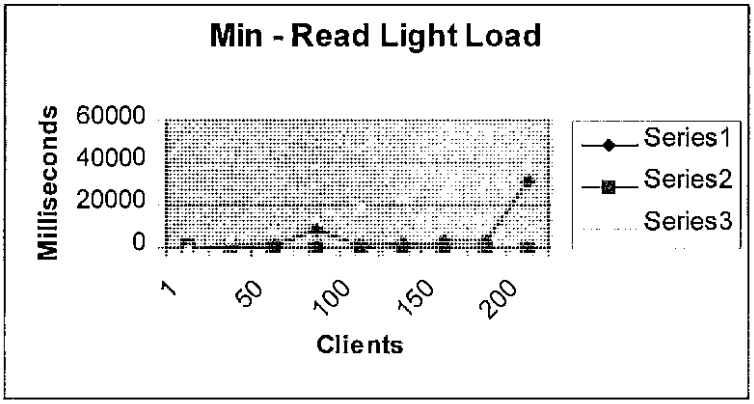

Max - Read Heavy Load
Milliseconds
600000
400000
200000
0
1
50
100
150
200
Clients
Java 2 ORB
VisiBroker
Orbix 2000

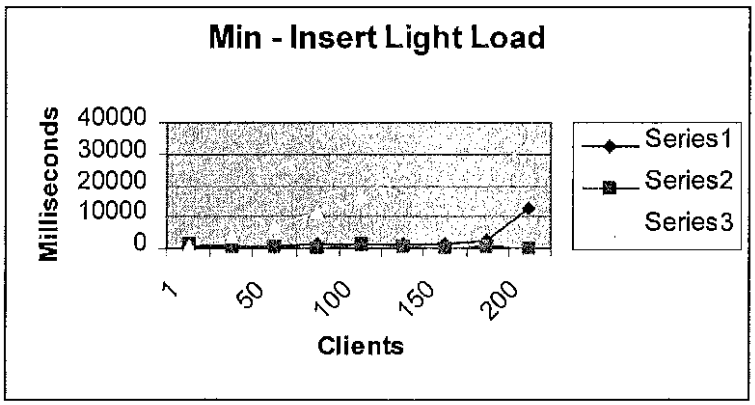

Max - Insert Heavy Load
Milliseconds
150000
100000
50000
0
1
50
100
150
200
Clients
Java 2 ORB
VisiBroker
Orbix 2000

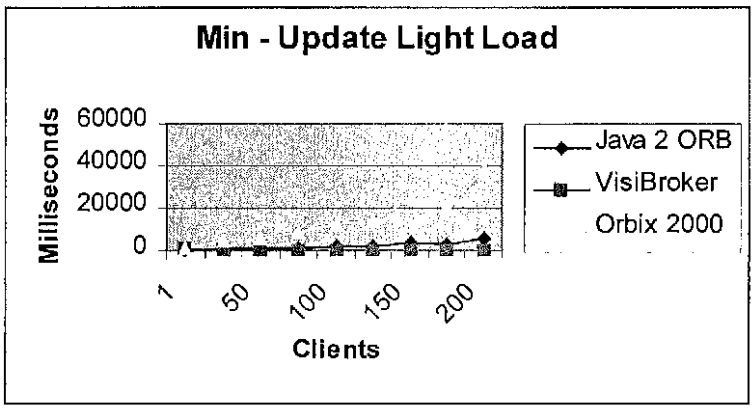

Max - Update Heavy Load
Milliseconds
100000
50000
0
1
50
100
150
200
Clients
Java 2 ORB
VisiBroker
Orbix 2000

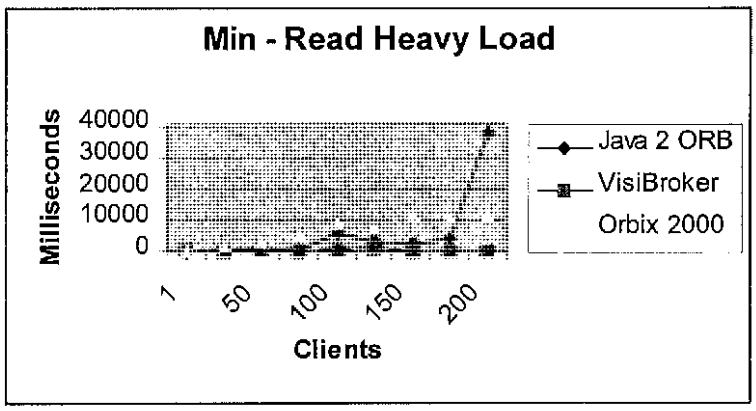
APPENDIX I

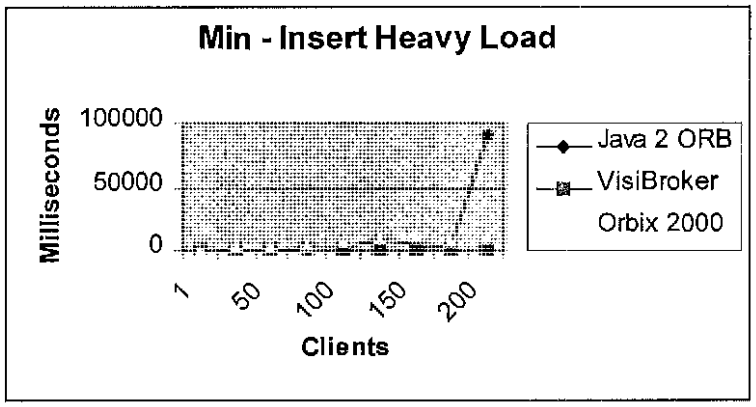
Minimum of Response Times

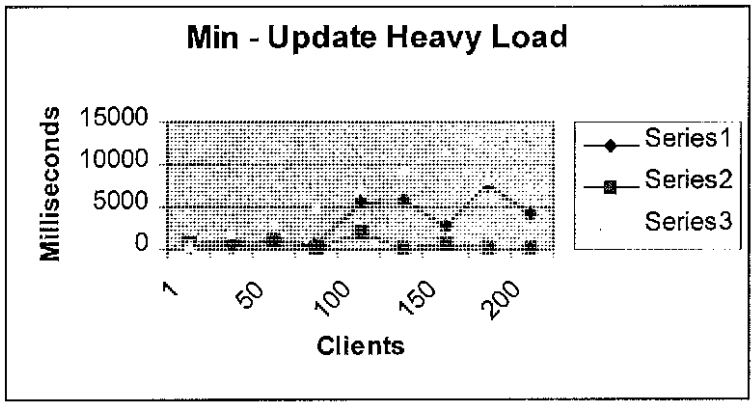


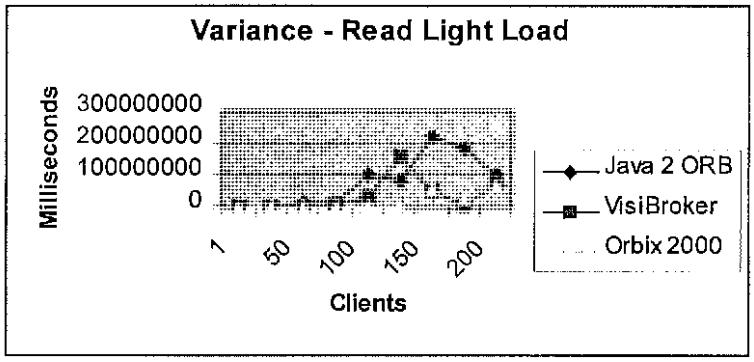

Min - Read Heavy Load
Milliseconds
40000
30000
20000
10000
0
1
50
100
150
200
Clients
Java 2 ORB
VisiBroker
Orbix 2000

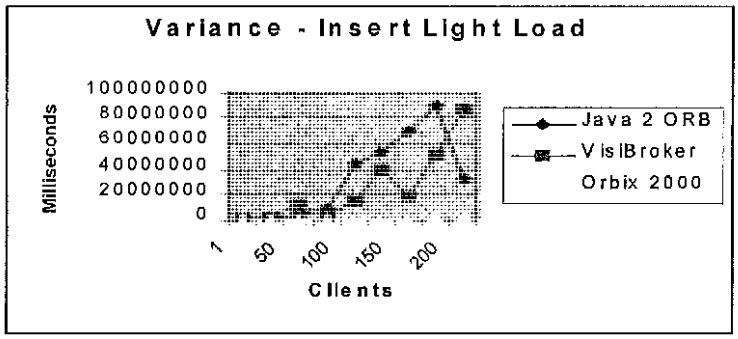

Min - Insert Heavy Load
Milliseconds
100000
50000
0
1
50
100
150
200
Clients
Java 2 ORB
VisiBroker
Orbix 2000

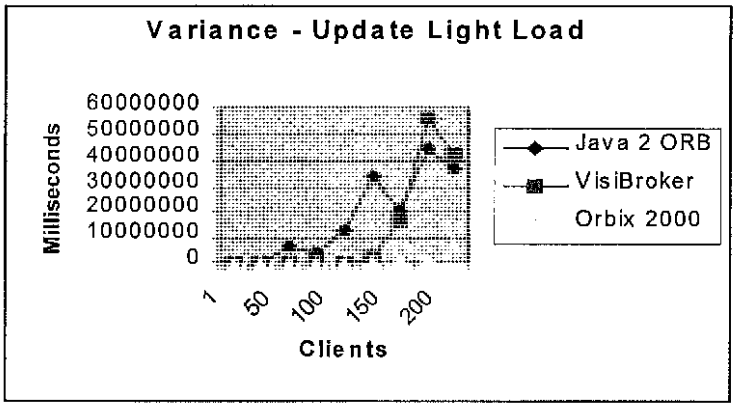

Min - Update Heavy Load
Milliseconds
15000
10000
5000
0
1
50
100
150
200
Clients
Series1
Series2
Series3

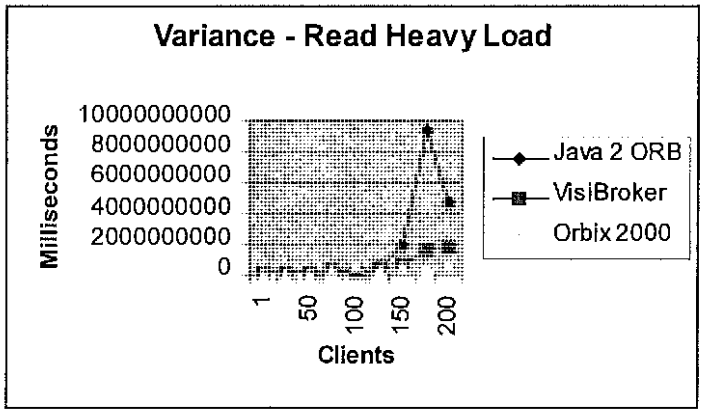
APPENDIX J

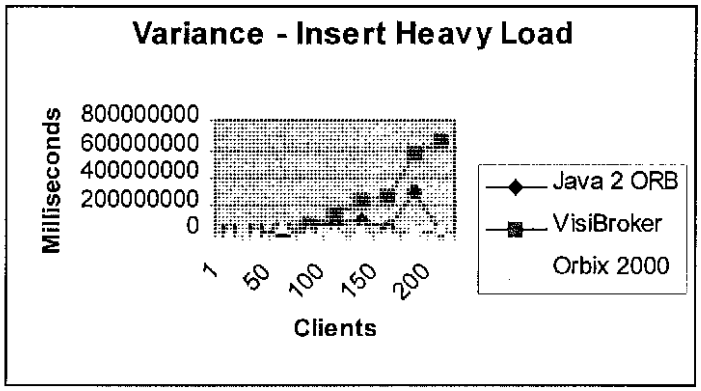
Variance of Response Times

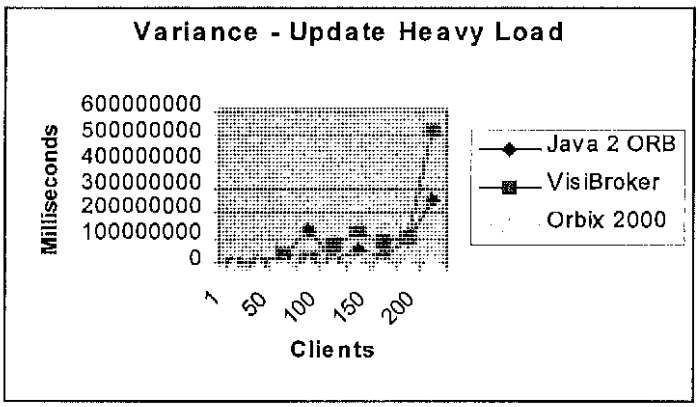



Variance - Read Heavy Load
Milliseconds
10000000000
8000000000
6000000000
4000000000
2000000000
0
1
50
100
150
200
Clients
Java 2 ORB
VisiBroker
Orbix 2000


Variance - Insert Heavy Load
Milliseconds
800000000
600000000
400000000
200000000
0
1
50
100
150
200
Clients
Java 2 ORB
VisiBroker
Orbix 2000


Variance - Update Heavy Load
Milliseconds
600000000
500000000
400000000
300000000
200000000
100000000
0
1
50
100
150
200
Clients
Java 2 ORB
VisiBroker
Orbix 2000

# APPENDIX K

## Administration of Orbix 2000

### Configuration

After Orbix 2000 has been installed, it must be configured in order to use any of the CORBA services. When Orbix 2000 is installed, the installer can choose a default local host configuration or to run a manual configuration. The manual configuration must be chosen, otherwise the services cannot be ran. The configuration must be ran on both the client and server, with the server being the first to be configured. Also, the user running the configuration must have root access to the server.

### Running a configuration repository across a network

When configuring Orbix 2000, when asked if the configuration will be file-based or configuration repository based, select configuration repository based, this will allow the naming service to run so that the client-server can communicate. When finished with configuring the server, save the client.prep file to a place where you have access to retrieve it. This file will be used to run the configuration on the client side. When you are ready to configure the client, the following command is used: **configure -useprep client.prep** this will automatically configure the client. The only thing that should need to be entered during the configuration is the host name your running the client on. One thing to be careful of is prior to running the configuration on the client, start the orbix 2000 services on the server side. During the client configuration, Orbix 2000 attempts to create a connection to the naming services running on the other machine, the server, if they are down it cannot complete the configuration. Also, one thing to make certain of is to give the capability to start and stop the Orbix 2000 services to someone else besides root. If you do not change the default on this, then only the system administrator can start and stop the services. The purpose of running the configuration in this manner, is to be able to have two machines communicate across a network. If the configuration repository is not set up in this manner, then only a client running on the same machine as the server can communicate with it.

# APPENDIX L

## Java 2 ORB Application

```
// Michelle McKeller - Graduate Project
// Corba - Java 2 Orb
// CBClient

import java.io.*;
import java.util.*;

public class CBClient
{
        public static void main(String[] args)
        {
                try
                {
                        String typeTest = "";
                        String numClients = "";
                        String Url = "";
                        String fileName = "";
                        String inLine = "";
                        File name;
                        boolean bToken = false;
                        int iNum = 0;
                        int iIndex = 0;
                        int iTokenCt = 0;

                        //Retrieving command line arguments
                        if ( args.length < 12 )
                        {
                                System.out.println("At least 12 command-line arguments are needed.");
                                System.exit(0);
                        }

                        // Type of test we're running, read, insert, or update
                        typeTest = args[8].trim().toUpperCase();

                        // Number of Clients running
                        numClients = args[9].trim();

                        // Convert from string to int we'll need this value for, for loops
                        iNum = Integer.valueOf(numClients).intValue();
                        System.out.println("Number of Clients running will be: " + iNum);

                        // Name of text file we'll be using for the test
                        fileName = args[10].trim();

                        // URL should be in the format of "http://neptune.cocsc.unf.edu:2400/mmckeller/Corba/CBServer"
                        Url = args[11].trim();

                        // Reading in data from file given on command line
                        // However many clients are supposed to run, that is how
                        // many lines we will read, one for each client, then
                        // spawn a client thread.
                        System.out.println( "Searching for file entered on command line..." );

                        name = new File( fileName );
```

```
                        if( name.exists() )
                        {
                                System.out.println( "File found, reading in 1 line per client thread.");
                                FileReader inFile = new FileReader ( name );
                                BufferedReader inBuff = new BufferedReader( inFile );

                                // Reading in one line per client requested, then spawning the client thread
                                for ( iIndex = 0; iIndex < iNum; iIndex++ )
                                {
                                        if ( (inLine = inBuff.readLine()) != null )
                                        {
                                                StringTokenizer st = new StringTokenizer( inLine, "\t\n\r", bToken
);

                                                String [] items = new String [ ( st.countTokens() ) ] ;

                                                try
                                                {
                                                  iTokenCt = 0;
                                                        while ( st.hasMoreTokens() )
                                                        {
                                                                String word = st.nextToken();

                                                                items[ iTokenCt ] = word;


                                                                iTokenCt = iTokenCt + 1;

                                                        }
                                                }catch( NoSuchElementException e )
                                                {
                                                        // If any of the fields were empty, this is going
                                                        // to get thrown, but thats acceptable
                                                }
                                                // Spawning a client thread and passing it the string array of data
                                                // that we read in

                                                new CBClientThread( args, typeTest, String.valueOf( iIndex + 1),
Url, items ).start();
                                        }else
                                        {
                                                // The file has to have as many rows in it as there are clients listed
                                                // on the command line if not, this error occurs
                                                System.out.println( "File does not support number of clients on
command line." );
                                                System.exit(0);
                                        }
                                }
                        }else
                        {
                                System.out.println( "File Name entered on command line does not exist.");
                                System.exit(0);
                        }
                }catch(IOException e)
                {
                        System.out.println("An error occurred processing request.");
                }
        }
}
// End of CBClient
```

```
// Michelle McKeller - Graduate Project
// Corba - Java 2 Orb
// CBClientThread

import java.lang.*;
import java.io.*;

// All CORBA applications need these classes
import java.net.MalformedURLException;
import java.util.Locale;
import org.omg.CosNaming.*;
import org.omg.CORBA.*;
import CBBankObjects.*;


public class CBClientThread extends Thread
{
        String aTypeTest, aUrl, aClientNum;
        String [] aArgs;
        String [] aDataItems;

  public CBClientThread(String [] args, String typeTest, String clientNum, String Url, String [] dataItems)
        {
                aArgs = args;
          aTypeTest = typeTest;
                aClientNum = clientNum;
                aUrl = Url;
                aDataItems = dataItems;
  }

  public void run()
        {
                long time1 = 0;
                long time2 = 0;

                try
                {
                        FileWriter Log = new FileWriter( "dbDataLog.txt", true );
                        BufferedWriter w = new BufferedWriter(Log);

                        // Create and initialize the ORB
                        ORB orb = ORB.init( aArgs, null );
                        // Get the root naming context
                        org.omg.CORBA.Object objRef = orb.resolve_initial_references("NameService");
                        NamingContext ncRef = NamingContextHelper.narrow(objRef);
                        // Resolve the object reference in naming
                        NameComponent nc = new NameComponent(aUrl, "");
                        NameComponent path[] = {nc};
                        CBManager mgr = CBManagerHelper.narrow(ncRef.resolve(path));

                        if ( mgr == null )
                        {
                                System.out.println( "Error: Manager object equals null.");
                                System.exit(0);
                        }

                        CBAccount acct = null;
                        CBTransaction trans = null;
                        String [] acctData = null;
                        String [] transData = null;
                        String rowsAffected = "";
                        int i = 0;

                        // Deciding whether to read, insert, or update
                        if ( aTypeTest.equals( "READ" ) )
                        {
                                // Reads an account record
                                // mimics a heavy load

                                // Start timer
```

```
                    time1 = System.currentTimeMillis();

                    // Reading the main account record
                    acct = mgr.getAccount( aDataItems[0] );

                    if ( acct == null )
                    {
                            // Stop timer
                            time2 = System.currentTimeMillis();

                            System.out.println( "Read account failed constructing account object." );
                    }else
                    {
                            acctData = acct.readAccount();

                            // Stop timer
                            time2 = System.currentTimeMillis();

                            if ( acctData[0].trim() == "" )
                            {
                                    System.out.println( "Read account failed reading record from the
database." );
                            }else
                            {
                                    w.write( "Returned data was: " );

                                    for ( i = 0; i < 25; i++ )
                                    {
                                            w.write( acctData[i] );
                                            w.newLine();

                                            System.out.println( "Account data returned from server
was: " + acctData[i] );
                                    }
                            }
                    }

            }else if ( aTypeTest.equals( "INSERT" ) )
            {
                    // Inserts a new transaction record
                    // mimics a light load

                    // Start timer
                    time1 = System.currentTimeMillis();

                    // Inserting a transaction record
                    acct = mgr.getAccount( aDataItems[0] );

                    if ( acct == null )
                    {
                            // Stop timer
                            time2 = System.currentTimeMillis();

                            System.out.println( "Insert transaction failed constructing account object." );

                    }else
                    {
                            trans = acct.putTransaction( aDataItems[0], aDataItems );
                            rowsAffected = trans.insertTransaction();

                            // Stop timer
                            time2 = System.currentTimeMillis();

                            if ( rowsAffected.trim() == "" )
                            {
                                    System.out.println( "Transaction insert failed when trying to insert
record into the database." );
                            }else
                            {
                                    w.write( "Rows affected: " + rowsAffected );
```

```
                    w.newLine();

                    System.out.println( "Rows affected: " + rowsAffected );
                }
            }


        }else if ( aTypeTest.equals( "UPDATE" ) )
        {
            // Updates an existing account record
            // mimics a heavy load

            // Start timer
            time1 = System.currentTimeMillis();

            acct = mgr.putAccount( aDataItems[0], aDataItems );

            if ( acct == null )
            {
                // Stop timer
                time2 = System.currentTimeMillis();

                System.out.println( "Update account failed constructing account object.");

            }else
            {
                rowsAffected = acct.updateAccount();

                // Stop timer
                time2 = System.currentTimeMillis();

                if ( rowsAffected == "" )
                {
                    System.out.println( "Update account failed when trying to update
the record in the database." );

                }else
                {
                    w.write( "Rows affected: " + rowsAffected );
                    w.newLine();

                    System.out.println( "Rows affected: " + rowsAffected );
                }
            }

        }else if ( aTypeTest.equals( "INSERTACCT" ) )
        {
            // Inserts a new account record
            // mimics a heavy load

            // Start timer
            time1 = System.currentTimeMillis();

            // Inserting a new account
            acct = mgr.putAccount( aDataItems[0], aDataItems );

            if ( acct == null )
            {
                // Stop timer
                time2 = System.currentTimeMillis();

                System.out.println( "Insert account failed constructing account object.");

            }else
            {
                rowsAffected = acct.insertAccount();

                // Stop timer
                time2 = System.currentTimeMillis();

                if ( rowsAffected == "" )
```

```
                                        {
                                                System.out.println( "Insert Account failed when trying to insert
record into the database." );
                                        }else
                                        {
                                                w.write( "Rows affected: " + rowsAffected );
                                                w.newLine();

                                                System.out.println( "Rows affected: " + rowsAffected );
                                        }
                                }

                        }else if ( aTypeTest.equals( "READTRANS" ) )
                        {
                                // This retrieves the last transaction record for the account
                                // mimics a light load

                                // Start timer
                                time1 = System.currentTimeMillis();

                                acct = mgr.getAccount( aDataItems[0] );

                                if ( acct == null )
                                {
                                        // Stop timer
                                        time2 = System.currentTimeMillis();

                                        System.out.println( "Read transaction failed constructing the account object."
);
                                }else
                                {
                                        trans = acct.getTransaction( aDataItems[0] );
                                        if ( trans == null )
                                        {
                                                // Stop timer
                                                time2 = System.currentTimeMillis();

                                                System.out.println( "Read transaction failed constructing
transaction object." );
                                        }else
                                        {
                                                transData = trans.readTransaction();

                                                // Stop timer
                                                time2 = System.currentTimeMillis();

                                                if ( transData[0].trim() == "" )
                                                {
                                                        System.out.println( "Read transaction failed reading the
record from the database." );
                                                }else
                                                {
                                                        w.write( "Returned data was: " );

                                                        for ( i = 0; i < 4; i++ )
                                                        {
                                                                w.write( transData[i] );
                                                                w.newLine();

                                                                System.out.println( "Transaction data
returned from server was: " + transData[i] );
                                                        }
                                                }
                                        }
                                }
                        }else if ( aTypeTest.equals( "UPDATETRANS" ) )
                        {
                                // Start timer
                                time1 = System.currentTimeMillis();
```

```
                                        acct = mgr.getAccount( aDataItems[0] );

                                        if ( acct == null )
                                        {
                                                // Stop timer
                                                time2 = System.currentTimeMillis();

                                                System.out.println( "Update transaction failed constructing the account
object." );
                                        }else
                                        {
                                                trans = acct.putTransaction( aDataItems[0], aDataItems );

                                                if ( trans == null )
                                                {
                                                        // Stop timer
                                                        time2 = System.currentTimeMillis();

                                                        System.out.println( "Update transaction failed constructing the
transaction object." );
                                                }else
                                                {
                                                        rowsAffected = trans.updateTransaction();

                                                        // Stop timer
                                                        time2 = System.currentTimeMillis();

                                                        if ( rowsAffected.trim() == "" )
                                                        {
                                                                System.out.println( "Update transaction failed when
updating the record in the database." );
                                                        }else
                                                        {
                                                                w.write( "Rows affected: " + rowsAffected );
                                                                w.newLine();

                                                                System.out.println( "Rows Affected: " + rowsAffected
);
                                                        }
                                                }
                                        }
                                }
                                w.close();
                        }catch( IOException e )
                        {
                                System.out.println( "An error occurred writing to the dbDataLog file." );
                                e.printStackTrace();
                        }
                        catch( Exception e )
                        {
                                System.out.println( "An error has occurred in the Client." );
                                e.printStackTrace();
                        }

                        try
                        {
                                FileWriter Log = new FileWriter( "timelog.txt", true );
                                BufferedWriter w = new BufferedWriter(Log);
                                w.write( "Time spent on Client " + aClientNum + ": " + ( time2 - time1 ) + " Milliseconds" );
                                w.newLine();
                                w.close();
                        }catch(IOException e)
                        {
                                System.out.println( "Problem occurred creating log file: " + e );
                        }

                        System.out.println( "Time spent on Client " + aClientNum + ": " + (time2 - time1) + " Milliseconds" );
        }
}
// End of CBClientThread
```

```
// Michelle McKeller - Graduate Project
// Corba - Java 2 Orb
// CBServer

import java.io.IOException;
import java.net.MalformedURLException;
import org.omg.CosNaming.*;
import org.omg.CosNaming.NamingContextPackage.*;
import org.omg.CORBA.*;
import CBBankObjects.*;

public class CBServer
{
        // public No-argument constructor
 public CBServer()
 {
 }

 public static void main(String args[])
 {
        new CBServer();

        CBDBServer dbServer = null;
        CBManagerImpl ManImpl = null;

        String ServerUrl = "";
        String Driver = "";
        String dbURL = "";
        String User = "";
        String Password = "";

        try
        {
                if ( args.length < 11 )
                {
                        System.out.println("At least 11 command-line arguments are needed.");
                        System.exit(0);
                }

                ServerUrl = args[6].trim();
                Driver = args[7].trim();
                dbURL = args[8].trim();
                User = args[9].trim();
                Password = args[10].trim();

                System.out.println( "Server running on: " + ServerUrl );

    // Initialize the ORB.
    org.omg.CORBA.ORB orb = org.omg.CORBA.ORB.init(args,null);
                System.out.println( "ORB Initialized.");

                // Create a Database Server Object
                dbServer = new CBDBServer( Driver, dbURL, User, Password );

                // Create a Manager Object
                ManImpl = new CBManagerImpl( dbServer );

                orb.connect(ManImpl);

    // print stringified object reference
    System.out.println( "Created manager\n" + orb.object_to_string(ManImpl) );

    // Get the root naming context
    org.omg.CORBA.Object objRef = orb.resolve_initial_references("NameService");
    NamingContext ncRef = NamingContextHelper.narrow(objRef);
                System.out.println( "Name Service Initialized." );

                // Bind the object reference in naming
                NameComponent nc = new NameComponent(ServerUrl, "");
                NameComponent path[] = {nc};
```

```
                    ncRef.rebind(path, ManImpl);
                    System.out.println( "Name Component bound." );

                    // Wait for invocations from clients
                    java.lang.Object sync = new java.lang.Object();
                    synchronized(sync)
                    {
                            sync.wait();
            }

            }catch (Exception e)
            {
             System.err.println("Failure during object export to CORBA: " + e);
             e.printStackTrace(System.out);
             System.exit(-1);
            }
 }
}

// End of CBServer
```

```
// Michelle McKeller - Graduate Project
// Corba - Java 2 Orb
// CBDBServer

import java.net.*;
import java.sql.*;
import java.io.*;

public class CBDBServer {

        String sDriver = "";
        String sdbURL = "";
        String sUser = "";
        String sPassword = "";
        String dbaseURL = "";
        Connection dbConnection;
        Statement query;

        // Constructor
        public CBDBServer( String Driver, String dbURL, String User, String Password )
        {
                sDriver = Driver; //oracle.jdbc.driver.OracleDriver
                sdbURL = dbURL; // "jdbc:oracle:thin:@manatee.cocse.unf.edu:1521:sid1
                sUser = User;
                sPassword = Password;

                initialize();
        }

        public void initialize()
        {
                try
                {
                        Class.forName( sDriver );
                        dbaseURL = sdbURL;
                        dbConnection = DriverManager.getConnection(dbaseURL, sUser, sPassword);
                        query = dbConnection.createStatement();
    }
    catch(SQLException e)
                {
      System.out.println("failed");
                        e.printStackTrace();
    }
                catch(ClassNotFoundException e)
                {
                        System.out.println("failed cl");
                }
        }

        public int runDBInOrUpQuery( String sSqlQuery )
        {
                int rows = 0;

                try
                {
                        rows = query.executeUpdate( sSqlQuery );

    }
    catch(SQLException e)
                {
      System.out.println("failed");
                        e.printStackTrace();
    }

                return rows;
        }

  public ResultSet runDBReadQuery( String sSqlQuery )
        {
                ResultSet results = null;
```

```
        try
                        {
                                results = query.executeQuery( sSqlQuery );
        }
        catch(SQLException e)
                        {
          System.out.println("failed");
                                e.printStackTrace();
        }

                        return results;
                }
}

// End of CBDBServer
```

```
// Michelle McKeller - Graduate Project
// Corba - Java 2 Orb
// CBManagerImpl

import CBBankObjects.*;

public class CBManagerImpl extends CBBankObjects._CBManagerImplBase
{
        CBDBServer dbSvr;

        // public No-argument constructor
        public CBManagerImpl( CBDBServer mydbServer)
        {
                this.dbSvr = mydbServer;
        }

        // Returns an Account object.
        public CBAccount getAccount( String Id ) throws Unknown
        {
    CBAccountImpl Acct = new CBAccountImpl( Id, dbSvr );
          System.out.println("Returning Account Object for the given account number.");
          return Acct;
        }

        // Adds an Account Object
        public CBAccount putAccount( String Id, String [] Data) throws Unknown
        {
                CBAccountImpl Account = new CBAccountImpl( Id, Data, dbSvr );
          System.out.println("Returning Account Object for the given account number.");
    return Account;
        }
}

// End of CBManagerImpl
```

```
// Michelle McKeller - Graduate Project
// Corba - Java 2 Orb
// CBAccountImpl

import java.sql.*;

import CBBankObjects.*;

public class CBAccountImpl extends CBBankObjects._CBAccountImplBase
{
   private String Id;
   private String Data[];
            private CBDBServer dbSvr;

   // public constructor
   public CBAccountImpl( String sId, CBDBServer mydbSvr )
   {
      this.Id = sId;
                        this.dbSvr = mydbSvr;
   }

   // overloaded constructor
   public CBAccountImpl( String sId, String [] sData, CBDBServer mydbSvr )
   {
      this.Id = sId;
      this.Data = sData;
                        this.dbSvr = mydbSvr;
   }

   // Methods for accessing a Transaction Object
   public CBTransaction getTransaction( String sId ) throws Unknown
   {
      CBTransactionImpl Trans = new CBTransactionImpl( sId, dbSvr );
      System.out.println("Returning Transaction Object for the given account number.");
      return Trans;
   }

   public CBTransaction putTransaction( String sId , String [] sData) throws Unknown
   {
      CBTransactionImpl Transact = new CBTransactionImpl( sId, sData, dbSvr );
      System.out.println("Returning Transaction Object for the given account number.");
      return Transact;
   }

   // Gets data for the Account object
   public synchronized String [] readAccount( )
   {
        String [] InfoData = new String [ 25 ];

        String sqlQuery = "SELECT acct_num, pin_num, acct_status_cd, acct_type_cd, ssn, last_name, first_name, middle_initial,
prim_address, prim_city, prim_state_cd, prim_zip_cd, prim_zip_cd_ext, sec_address, sec_city, sec_state_cd, sec_zip_cd,
sec_zip_cd_ext, prim_phone_num, sec_phone_num, fax_num, create_dt, modified_dt, comments, addtl_comments FROM accounts
WHERE acct_num = '" + Id + "'";

                                System.out.println( "Sql query being sent to the db is: " + sqlQuery );

        ResultSet results = dbSvr.runDBReadQuery( sqlQuery );

                                try
                                {
                                        results.next();

                                        InfoData[0] = results.getString( "acct_num" );
                                        System.out.println( "result: " + InfoData[0] );

                                        InfoData[1] = results.getString( "pin_num" );
                                        System.out.println( "result: " + InfoData[1] );

                                        InfoData[2] = results.getString( "acct_status_cd" );
                                        System.out.println( "result: " + InfoData[2] );
```

```
                InfoData[3] = results.getString( "acct_type_cd" );
                System.out.println( "result: " + InfoData[3] );

                InfoData[4] = results.getString( "ssn" );
                System.out.println( "result: " + InfoData[4] );

                InfoData[5] = results.getString( "last_name" );
                System.out.println( "result: " + InfoData[5] );

                InfoData[6] = results.getString( "first_name" );
                System.out.println( "result: " + InfoData[6] );

                InfoData[7] = results.getString( "middle_initial" );
                System.out.println( "result: " + InfoData[7] );

                InfoData[8] = results.getString( "prim_address" );
                System.out.println( "result: " + InfoData[8] );

                InfoData[9] = results.getString( "prim_city" );
                System.out.println( "result: " + InfoData[9] );

                InfoData[10] = results.getString( "prim_state_cd" );
                System.out.println( "result: " + InfoData[10] );

                InfoData[11] = results.getString( "prim_zip_cd" );
                System.out.println( "result: " + InfoData[11] );

                InfoData[12] = results.getString( "prim_zip_cd_ext" );
                System.out.println( "result: " + InfoData[12] );

                InfoData[13] = results.getString( "sec_address" );
                System.out.println( "result: " + InfoData[13] );

                InfoData[14] = results.getString( "sec_city" );
                System.out.println( "result: " + InfoData[14] );

                InfoData[15] = results.getString( "sec_state_cd" );
                System.out.println( "result: " + InfoData[15] );

                InfoData[16] = results.getString( "sec_zip_cd" );
                System.out.println( "result: " + InfoData[16] );

                InfoData[17] = results.getString( "sec_zip_cd_ext" );
                System.out.println( "result: " + InfoData[17] );

                InfoData[18] = results.getString( "prim_phone_num" );
                System.out.println( "result: " + InfoData[18] );

                InfoData[19] = results.getString( "sec_phone_num" );
                System.out.println( "result: " + InfoData[19] );

                InfoData[20] = results.getString( "fax_num" );
                System.out.println( "result: " + InfoData[20] );

                InfoData[21] = results.getString( "create_dt" );
                System.out.println( "result: " + InfoData[21] );

                InfoData[22] = results.getString( "modified_dt" );
                System.out.println( "result: " + InfoData[22] );

                InfoData[23] = results.getString( "comments" );
                System.out.println( "result: " + InfoData[23] );

                InfoData[24] = results.getString( "addtl_comments" );
                System.out.println( "result: " + InfoData[24] );

}
catch(SQLException e)
{
```

```
                                        System.out.println("Read transaction failed: " + e );
                                    }

        return InfoData;
    }


    // Inserts the data for the account object
    public synchronized String insertAccount()
    {
        String sMessage = "";

                                        String sqlQuery = "INSERT INTO accounts( acct_num, pin_num, acct_status_cd, acct_type_cd, ssn,
last_name, first_name, middle_initial, prim_address, prim_city, prim_state_cd, prim_zip_cd, prim_zip_cd_ext, sec_address, sec_city,
sec_state_cd, sec_zip_cd, sec_zip_cd_ext, prim_phone_num, sec_phone_num, fax_num, create_dt, modified_dt, comments,
addtl_comments ) ";
                                        sqlQuery = sqlQuery + "VALUES( '" + Data[0] + "', '" + Data[1] + "', '" + Data[2] + "', '" + Data[3] +
"', '" + Data[4] + "', '" + Data[5] + "', '" + Data[6] + "', '" + Data[7] + "', '" + Data[8] + "', '" + Data[9] + "', '" + Data[10] + "', '" +
Data[11] + "', '" + Data[12] + "', '" + Data[13] + "', '" + Data[14] + "', '" + Data[15] + "', '";
                                        sqlQuery = sqlQuery + Data[16] + "', '" + Data[17] + "', '" + Data[18] + "', '" + Data[19] + "', '" +
Data[20] + "', '" + Data[21] + "', '" + Data[22] + "', '" + Data[23] + "', '" + Data[24] + "')";

                                        System.out.println( "Sql query being sent to the db is: " + sqlQuery );

        int rows = dbSvr.runDBInOrUpQuery( sqlQuery );

        sMessage = "Number of rows affected: " + String.valueOf( rows );

        return sMessage;
    }

    // Updates the data for the account object
    public synchronized String updateAccount()
    {
        String sMessage = "";

        String sqlQuery = "UPDATE accounts SET pin_num = '" + Data[1] + "', acct_status_cd = '" + Data[2] + "', acct_type_cd = '" +
Data[3] + "', ssn = '" + Data[4] + "', last_name = '" + Data[5] + "', first_name = '" + Data[6] + "', middle_initial = '" + Data[7] + "',
prim_address = '" + Data[8] + "', prim_city = '" + Data[9] + "', ";
                                        sqlQuery = sqlQuery + "prim_state_cd = '" + Data[10] + "', prim_zip_cd = '" + Data[11] + "',
prim_zip_cd_ext = '" + Data[12] + "', sec_address = '" + Data[13] + "', sec_city = '" + Data[14] + "', sec_state_cd = '" + Data[15] + "',
sec_zip_cd = '" + Data[16];
                                        sqlQuery = sqlQuery + "', sec_zip_cd_ext = '" + Data[17] + "', prim_phone_num = '" + Data[18] + "',
sec_phone_num = '" + Data[19] + "', fax_num = '" + Data[20] + "', create_dt = '" + Data[21] + "', modified_dt = '" + Data[22] + "',
comments = '" + Data[23] + "', addtl_comments = '" + Data[24] + "' WHERE acct_num = '" + Id + "'";

                                        System.out.println( "Sql query being sent to the db is: " + sqlQuery );

        int rows = dbSvr.runDBInOrUpQuery( sqlQuery );

        sMessage = "Number of rows affected: " + String.valueOf( rows );

        return sMessage;
    }

}
// End of CBAccountImpl
```

```
// Michelle McKeller - Graduate Project
// Corba - Java 2 Orb
// CBTransactionImpl

import java.sql.*;

import CBBankObjects.*;

public class CBTransactionImpl extends CBBankObjects._CBTransactionImplBase {
   private String Id;
   private String Data[];
            private CBDBServer dbSvr;

   // public constructor
   public CBTransactionImpl( String sId, CBDBServer mydbSvr )
   {
      this.Id = sId;
                        this.dbSvr = mydbSvr;
   }

   // overloaded constructor
   public CBTransactionImpl( String sId, String [] sData, CBDBServer mydbSvr )
   {
      this.Id = sId;
      this.Data = sData;
                        this.dbSvr = mydbSvr;
   }

   // Gets data for the transaction object
   public synchronized String [] readTransaction( )
   {
      String [] InfoData = new String [ 4 ];

      String sqlQuery = "SELECT trans_id, acct_num, trans_type_cd, trans_amt, create_dt FROM transactions WHERE acct_num = '"
+ Id + "' AND trans_id = (SELECT MAX(trans_id) FROM transactions WHERE acct_num = '" + Id + "')";

                        System.out.println( "Sql query being sent to the db is: " + sqlQuery );

      ResultSet results = dbSvr.runDBReadQuery( sqlQuery );

                        if ( results == null )
                        {
                                 InfoData[0] = "";
                        }else
                        {
                                 try
                                 {
                                          results.next();
                                          InfoData[0] = results.getString( "acct_num" );
                                          System.out.println( "results: " + InfoData[0]);

                                          InfoData[1] = results.getString( "trans_type_cd" );
                                          System.out.println( "results: " + InfoData[1]);

                                          InfoData[2] = results.getString( "trans_amt" );
                                          System.out.println( "results: " + InfoData[2]);

                                          InfoData[3] = results.getString( "create_dt" );
                                          System.out.println( "results: " + InfoData[3]);

                                 }
                                 catch(SQLException e)
                                 {
                                          System.out.println("Read transaction failed: " + e );
                                 }
                        }

      return InfoData;
   }
```

```
        // Inserts the data for the transaction object
        public synchronized String insertTransaction()
        {
                String sMessage = "";

                String sqlQuery = "INSERT INTO transactions(trans_id, acct_num, trans_type_cd, trans_amt, create_dt)
VALUES (trans_id_sequence.NEXTVAL, '" + Data[0] + "', '" + Data[1] + "', '" + Data[2] + "', '" + Data[3] + "')";

                System.out.println( "Sql query being sent to the db is: " + sqlQuery );

                int rows = dbSvr.runDBInOrUpQuery( sqlQuery );

                sMessage = "Number of rows affected: " + String.valueOf( rows );

                return sMessage;
        }

  // Updates the data for the transaction object
        public synchronized String updateTransaction()
        {
                String sMessage = "";

                String sqlQuery = "UPDATE transactions SET trans_type_cd = '" + Data[1] + "', trans_amt = '" + Data[2] + "',
create_dt = '" + Data[3] + "' WHERE acct_num = '" + Data[0] + "' AND trans_id = (SELECT MAX(trans_id) FROM transactions
WHERE acct_num = '" + Data[0] + "')";

                System.out.println( "Sql query being sent to the db is: " + sqlQuery );

                int rows = dbSvr.runDBInOrUpQuery( sqlQuery );

                sMessage = "Number of rows affected: " + String.valueOf( rows );

                return sMessage;
        }


}

// End of CBTransactionImpl
```

```
// Michelle McKeller - Graduate Project
// Corba - Java 2 Orb
// CBBank.idl

module CBBankObjects {
  exception Unknown{};

        typedef string CBAcct[25];
        typedef string CBTrans[4];

        interface CBTransaction {
                // Reads the transaction data for a given account number
                CBTrans readTransaction() raises(Unknown);

                // Inserts the transaction data into the database
    // for a given account number
                string insertTransaction() raises(Unknown);

                // Updates the transaction data in the database
    // for a given account number and trans id
                string updateTransaction() raises(Unknown);
        };

        interface CBAccount {

                //Gets Last made transaction for an Account
                CBTransaction getTransaction(in string sId) raises(Unknown);

                //Puts Last made transaction for an Account
                CBTransaction putTransaction(in string sId, in CBTrans objItem) raises(Unknown);

                // Reads the account data for a given account number
                CBAcct readAccount() raises(Unknown);

                // Inserts the account data into the database
    // for a given account number
                string insertAccount() raises(Unknown);

                // Updates the account data in the database
    // for a given account number
                string updateAccount() raises(Unknown);
        };

        interface CBManager {
                // Gets an account object
                CBAccount getAccount(in string Id) raises(Unknown);

                // Puts an account object
                CBAccount putAccount(in string Id, in CBAcct objItem) raises(Unknown);
        };


};

// End of CBBank.idl
```

# APPENDIX M

## VisiBroker Application

```
// Michelle McKeller - Graduate Project
// Corba - VisiBroker
// VBClient

import java.io.*;
import java.util.*;

public class VBClient
{
        public static void main(String[] args)
        {
                try
                {
                        String typeTest = "";
                        String numClients = "";
                        String Url = "";
                        String fileName = "";
                        String inLine = "";
                        File name;
                        boolean bToken = false;
                        int iNum = 0;
                        int iIndex = 0;
                        int iTokenCt = 0;

                        //Retrieving command line arguments
                        if ( args.length < 4 )
                        {
                                System.out.println("At least 4 command-line arguments are needed.");
                                System.exit(0);
                        }

                        // Type of test we're running, read, insert, or update
                        typeTest = args[0].trim().toUpperCase();

                        // Number of Clients running
                        numClients = args[1].trim();

                        // Convert from string to int we'll need this value for, for loops
                        iNum = Integer.valueOf(numClients).intValue();
                        System.out.println("Number of Clients running will be: " + iNum);

                        // Name of text file we'll be using for the test
                        fileName = args[2].trim();

                        // URL should be in the format of
"http://neptune.cocse.unf.edu:2400/mmckeller/VisiBroker/VBServer"
                        Url = args[3].trim();

                        // Reading in data from file given on command line
                        // However many clients are supposed to run, that is how
                        // many lines we will read, one for each client, then
                        // spawn a client thread.
                        System.out.println( "Searching for file entered on command line..." );

                        name = new File( fileName );
```

```
                        if( name.exists() )
                        {
                                System.out.println( "File found, reading in 1 line per client thread.");
                                FileReader inFile = new FileReader ( name );
                                BufferedReader inBuff = new BufferedReader( inFile );

                                // Reading in one line per client requested, then spawning the client thread
                                for ( iIndex = 0; iIndex < iNum; iIndex++ )
                                {
                                        if ( (inLine = inBuff.readLine()) != null )
                                        {
                                                StringTokenizer st = new StringTokenizer( inLine, "\t\n\r", bToken
);

                                                String [] items = new String [ ( st.countTokens() ) ] ;

                                                try
                                                {
                                                  iTokenCt = 0;
                                                        while ( st.hasMoreTokens() )
                                                        {
                                                                String word = st.nextToken();

                                                                items[ iTokenCt ] = word;


                                                                iTokenCt = iTokenCt + 1;

                                                        }
                                                }catch( NoSuchElementException e )
                                                {
                                                        // If any of the fields were empty, this is going
                                                        // to get thrown, but thats acceptable
                                                }
                                                // Spawning a client thread and passing it the string array of data
                                                // that we read in

                                                new VBClientThread( args, typeTest, String.valueOf( iIndex + 1),
Url, items ).start();
                                        }else
                                        {
                                                // The file has to have as many rows in it as there are clients listed
                                                // on the command line if not, this error occurs
                                                System.out.println( "File does not support number of clients on
command line." );
                                                System.exit(0);
                                        }
                                }
                        }else
                        {
                                System.out.println( "File Name entered on command line does not exist.");
                                System.exit(0);
                        }
                }catch(IOException e)
                {
                        System.out.println("An error occurred processing request.");
                }
        }
}
// End of VBClient
```

```
// Michelle McKeller - Graduate Project
// Corba - VisiBroker
// VBClientThread

import java.lang.*;
import java.io.*;
import java.net.MalformedURLException;
import java.util.Locale;
import VBBankObjects.*;

public class VBClientThread extends Thread
{
        String aTypeTest, aUrl, aClientNum;
        String [] aArgs;
        String [] aDataItems;

  public VBClientThread(String [] args, String typeTest, String clientNum, String Url, String [] dataItems)
        {
                aArgs = args;
          aTypeTest = typeTest;
                aClientNum = clientNum;
                aUrl = Url;
                aDataItems = dataItems;
  }

  public void run()
        {
                long time1 = 0;
                long time2 = 0;

                try
                {
                        FileWriter Log = new FileWriter( "dbDataLog.txt", true );
                        BufferedWriter w = new BufferedWriter(Log);

                        // Create and initialize the ORB
                        org.omg.CORBA.ORB orb = org.omg.CORBA.ORB.init( aArgs, null );

                        VBManager mgr = VBManagerHelper.bind(orb, aUrl);

                        if ( mgr == null )
                        {
                                System.out.println( "Error: Manager object equals null.");
                                System.exit(0);
                        }

                        VBAccount acct = null;
                        VBTransaction trans = null;
                        String [] acctData = null;
                        String [] transData = null;
                        String rowsAffected = "";
                        int i = 0;

                        // Deciding whether to read, insert, or update
                        if ( aTypeTest.equals( "READ" ) )
                        {
                                // Reads an account record
                                // mimics a heavy load

                                // Start timer
                                time1 = System.currentTimeMillis();

                                // Reading the main account record
                                acct = mgr.getAccount( aDataItems[0] );

                                if ( acct == null )
                                {
                                        // Stop timer
                                        time2 = System.currentTimeMillis();
```

```
                    System.out.println( "Read account failed constructing account object." );
                }else
                {
                    acctData = acct.readAccount();

                    // Stop timer
                    time2 = System.currentTimeMillis();

                    if ( acctData[0].trim() == "" )
                    {
                        System.out.println( "Read account failed reading record from the
database." );
                    }else
                    {
                        w.write( "Returned data was: " );

                        for ( i = 0; i < 25; i++ )
                        {
                            w.write( acctData[i] );
                            w.newLine();

                            System.out.println( "Account data returned from server
was: " + acctData[i] );
                        }
                    }
                }

            }else if ( aTypeTest.equals( "INSERT" ) )
            {
                // Inserts a new transaction record
                // mimics a light load

                // Start timer
                time1 = System.currentTimeMillis();

                // Inserting a transaction record
                acct = mgr.getAccount( aDataItems[0] );

                if ( acct == null )
                {
                    // Stop timer
                    time2 = System.currentTimeMillis();

                    System.out.println( "Insert transaction failed constructing account object." );

                }else
                {
                    trans = acct.putTransaction( aDataItems[0], aDataItems );
                    rowsAffected = trans.insertTransaction();

                    // Stop timer
                    time2 = System.currentTimeMillis();

                    if ( rowsAffected.trim() == "" )
                    {
                        System.out.println( "Transaction insert failed when trying to insert
record into the database." );
                    }else
                    {
                        w.write( "Rows affected: " + rowsAffected );
                        w.newLine();

                        System.out.println( "Rows affected: " + rowsAffected );
                    }
                }


            }else if ( aTypeTest.equals( "UPDATE" ) )
            {
                // Updates an existing account record
```

```
                    // mimics a heavy load

                    // Start timer
                    time1 = System.currentTimeMillis();

                    acct = mgr.putAccount( aDataItems[0], aDataItems );

                    if ( acct == null )
                    {
                            // Stop timer
                            time2 = System.currentTimeMillis();

                            System.out.println( "Update account failed constructing account object.");

                    }else
                    {
                            rowsAffected = acct.updateAccount();

                            // Stop timer
                            time2 = System.currentTimeMillis();

                            if ( rowsAffected == "" )
                            {
                                    System.out.println( "Update account failed when trying to update
the record in the database." );
                            }else
                            {
                                    w.write( "Rows affected: " + rowsAffected );
                                    w.newLine();

                                    System.out.println( "Rows affected: " + rowsAffected );
                            }
                    }

            }else if ( aTypeTest.equals( "INSERTACCT" ) )
            {
                    // Inserts a new account record
                    // mimics a heavy load

                    // Start timer
                    time1 = System.currentTimeMillis();

                    // Inserting a new account
                    acct = mgr.putAccount( aDataItems[0], aDataItems );

                    if ( acct == null )
                    {
                            // Stop timer
                            time2 = System.currentTimeMillis();

                            System.out.println( "Insert account failed constructing account object.");

                    }else
                    {
                            rowsAffected = acct.insertAccount();

                            // Stop timer
                            time2 = System.currentTimeMillis();

                            if ( rowsAffected == "" )
                            {
                                    System.out.println( "Insert Account failed when trying to insert
record into the database." );
                            }else
                            {
                                    w.write( "Rows affected: " + rowsAffected );
                                    w.newLine();

                                    System.out.println( "Rows affected: " + rowsAffected );
                            }
```

```
                }

            }else if ( aTypeTest.equals( "READTRANS" ) )
            {
                // This retrieves the last transaction record for the account
                // mimics a light load

                // Start timer
                time1 = System.currentTimeMillis();

                acct = mgr.getAccount( aDataItems[0] );

                if ( acct == null )
                {
                    // Stop timer
                    time2 = System.currentTimeMillis();

                    System.out.println( "Read transaction failed constructing the account object."
);
                }else
                {
                    trans = acct.getTransaction( aDataItems[0] );
                    if ( trans == null )
                    {
                        // Stop timer
                        time2 = System.currentTimeMillis();

                        System.out.println( "Read transaction failed constructing
transaction object." );
                    }else
                    {
                        transData = trans.readTransaction();

                        // Stop timer
                        time2 = System.currentTimeMillis();

                        if ( transData[0].trim() == "" )
                        {
                            System.out.println( "Read transaction failed reading the
record from the database." );
                        }else
                        {
                            w.write( "Returned data was: " );

                            for ( i = 0; i < 4; i++ )
                            {
                                w.write( transData[i] );
                                w.newLine();

                                System.out.println( "Transaction data
returned from server was: " + transData[i] );
                            }
                        }
                    }
                }
            }else if ( aTypeTest.equals( "UPDATETRANS" ) )
            {
                // Start timer
                time1 = System.currentTimeMillis();

                acct = mgr.getAccount( aDataItems[0] );

                if ( acct == null )
                {
                    // Stop timer
                    time2 = System.currentTimeMillis();

                    System.out.println( "Update transaction failed constructing the account
object." );
                }else
```

```
                                        {
                                                trans = acct.putTransaction( aDataItems[0], aDataItems );

                                                if ( trans == null )
                                                {
                                                        // Stop timer
                                                        time2 = System.currentTimeMillis();

                                                        System.out.println( "Update transaction failed constructing the
transaction object." );
                                                }else
                                                {
                                                        rowsAffected = trans.updateTransaction();

                                                        // Stop timer
                                                        time2 = System.currentTimeMillis();

                                                        if ( rowsAffected.trim() == "" )
                                                        {
                                                                System.out.println( "Update transaction failed when
updating the record in the database." );
                                                        }else
                                                        {
                                                                w.write( "Rows affected: " + rowsAffected );
                                                                w.newLine();

                                                                System.out.println( "Rows Affected: " + rowsAffected
);
                                                        }
                                                }
                                        }
                                }
                                w.close();
                        }catch( IOException e )
                        {
                                System.out.println( "An error occurred writing to the dbDataLog file." );
                                e.printStackTrace();
                        }
                        catch( Exception e )
                        {
                                System.out.println( "An error has occurred in the Client." );
                                e.printStackTrace();
                        }

                        try
                        {
                                FileWriter Log = new FileWriter( "timelog.txt", true );
                                BufferedWriter w = new BufferedWriter(Log);
                                w.write( "Time spent on Client " + aClientNum + ": " + ( time2 - time1 ) + " Milliseconds" );
                                w.newLine();
                                w.close();
                        }catch(IOException e)
                        {
                                System.out.println( "Problem occurred creating log file: " + e );
                        }

                        System.out.println( "Time spent on Client " + aClientNum + ": " + (time2 - time1) + " Milliseconds" );
        }
}
// End of VBClientThread
```

```
// Michelle McKeller - Graduate Project
// Corba - VisiBroker
// VBServer

import java.io.IOException;
import java.net.MalformedURLException;
import VBBankObjects.*;

public class VBServer
{
        // public No-argument constructor
 public VBServer()
 {
 }

 public static void main(String args[])
 {
        new VBServer();

        VBDBServer dbServer = null;

        String ServerUrl = "";
        String Driver = "";
        String dbURL = "";
        String User = "";
        String Password = "";

        try
        {
                if ( args.length < 5 )
                {
                        System.out.println("At least 5 command-line arguments are needed.");
                        System.exit(0);
                }

                ServerUrl = args[0].trim();
                Driver = args[1].trim();
                dbURL = args[2].trim();
                User = args[3].trim();
                Password = args[4].trim();

    // Initialize the ORB.
    org.omg.CORBA.ORB orb = org.omg.CORBA.ORB.init(args,null);
                System.out.println( "ORB Initialized.");

                // Initialize the BOA.
                com.inprise.vbroker.CORBA.BOA boa = ((com.inprise.vbroker.CORBA.ORB)orb).BOA_init();

                // Create a Database Server Object
                dbServer = new VBDBServer( Driver, dbURL, User, Password );

                // Create a Manager Object
                VBManager Man = new VBManagerImpl( ServerUrl, dbServer );

                // Export the newly created object.
                boa.obj_is_ready(Man);

                System.out.println("Created manager\n" + orb.object_to_string(Man) );

                // Wait for incoming requests
                boa.impl_is_ready();

        }catch (Exception e)
        {
         System.err.println("Failure during object export to CORBA: " + e);
         e.printStackTrace(System.out);
         System.exit(-1);
        }
 }
}
```

```
// End of VBServer
```

```
// Michelle McKeller - Graduate Project
// Corba - VisiBroker
// VBDBServer

import java.net.*;
import java.sql.*;
import java.io.*;

public class VBDBServer {

        String sDriver = "";
        String sdbURL = "";
        String sUser = "";
        String sPassword = "";
        String dbaseURL = "";
        Connection dbConnection;
        Statement query;

        // Constructor
        public VBDBServer( String Driver, String dbURL, String User, String Password )
        {
                sDriver = Driver; //oracle.jdbc.driver.OracleDriver
                sdbURL = dbURL; // "jdbc:oracle:thin:@manatee.cocse.unf.edu:1521:sid1
                sUser = User;
                sPassword = Password;

                initialize();
        }

        public void initialize()
        {
                try
                {
                        Class.forName( sDriver );
                        dbaseURL = sdbURL;
                        dbConnection = DriverManager.getConnection(dbaseURL, sUser, sPassword);
                        query = dbConnection.createStatement();
    }
    catch(SQLException e)
                {
      System.out.println("failed");
                        e.printStackTrace();
    }
                catch(ClassNotFoundException e)
                {
                        System.out.println("failed cl");
                }
        }

        public int runDBInOrUpQuery( String sSqlQuery )
        {
                int rows = 0;

                try
                {
                        rows = query.executeUpdate( sSqlQuery );

    }
    catch(SQLException e)
                {
      System.out.println("failed");
                        e.printStackTrace();
    }

                return rows;
        }

  public ResultSet runDBReadQuery( String sSqlQuery )
        {
                ResultSet results = null;
```

```
    try
                    {
                            results = query.executeQuery( sSqlQuery );
    }
    catch(SQLException e)
                    {
      System.out.println("failed");
                            e.printStackTrace();
    }

                    return results;
          }
}

// End of VBDBServer
```

```
// Michelle McKeller - Graduate Project
// Corba - VisiBroker
// VBManagerImpl

import VBBankObjects.*;

public class VBManagerImpl extends VBBankObjects._VBManagerImplBase
{
        VBDBServer dbSvr;

        // public No-argument constructor
        public VBManagerImpl( String Url, VBDBServer mydbServer)
        {
                super( Url );
                this.dbSvr = mydbServer;
        }

        // Returns an Account object.
        public VBAccount getAccount( String Id ) throws Unknown
        {
    VBAccount Acct = new VBAccountImpl( Id, dbSvr );

          System.out.println("Returning Account Object for the given account number.");

          return Acct;
        }

        // Adds an Account Object
        public VBAccount putAccount( String Id, String [] Data) throws Unknown
        {
                VBAccount Account = new VBAccountImpl( Id, Data, dbSvr );

          System.out.println("Returning Account Object for the given account number.");

    return Account;
        }
}

// End of VBManagerImpl
```

```
// Michelle McKeller - Graduate Project
// Corba - VisiBroker
// VBAccountImpl

import java.sql.*;
import VBBankObjects.*;

public class VBAccountImpl extends VBBankObjects._VBAccountImplBase
{
  private String Id;
  private String Data[];
        private VBDBServer dbSvr;

  // public constructor
  public VBAccountImpl( String sId, VBDBServer mydbSvr )
  {
    this.Id = sId;
                this.dbSvr = mydbSvr;
  }

  // overloaded constructor
  public VBAccountImpl( String sId, String [] sData, VBDBServer mydbSvr )
  {
    this.Id = sId;
    this.Data = sData;
                this.dbSvr = mydbSvr;
  }

  // Methods for accessing a Transaction Object
  public VBTransaction getTransaction( String sId ) throws Unknown
  {
    VBTransaction Trans = new VBTransactionImpl( sId, dbSvr );
    System.out.println("Returning Transaction Object for the given account number.");
    return Trans;
  }

  public VBTransaction putTransaction( String sId , String [] sData) throws Unknown
  {
    VBTransaction Transact = new VBTransactionImpl( sId, sData, dbSvr );
    System.out.println("Returning Transaction Object for the given account number.");
    return Transact;
  }

  // Gets data for the Account object
  public synchronized String [] readAccount( )
  {
      String [] InfoData = new String [ 25 ];

      String sqlQuery = "SELECT acct_num, pin_num, acct_status_cd, acct_type_cd, ssn, last_name, first_name, middle_initial,
prim_address, prim_city, prim_state_cd, prim_zip_cd, prim_zip_cd_ext, sec_address, sec_city, sec_state_cd, sec_zip_cd,
sec_zip_cd_ext, prim_phone_num, sec_phone_num, fax_num, create_dt, modified_dt, comments, addtl_comments FROM accounts
WHERE acct_num = '" + Id + "'";

                System.out.println( "Sql query being sent to the db is: " + sqlQuery );

      ResultSet results = dbSvr.runDBReadQuery( sqlQuery );

                try
                {
                        results.next();

                        InfoData[0] = results.getString( "acct_num" );
                        System.out.println( "result: " + InfoData[0] );

                        InfoData[1] = results.getString( "pin_num" );
                        System.out.println( "result: " + InfoData[1] );

                        InfoData[2] = results.getString( "acct_status_cd" );
                        System.out.println( "result: " + InfoData[2] );
```

```
                InfoData[3] = results.getString( "acct_type_cd" );
                System.out.println( "result: " + InfoData[3] );

                InfoData[4] = results.getString( "ssn" );
                System.out.println( "result: " + InfoData[4] );

                InfoData[5] = results.getString( "last_name" );
                System.out.println( "result: " + InfoData[5] );

                InfoData[6] = results.getString( "first_name" );
                System.out.println( "result: " + InfoData[6] );

                InfoData[7] = results.getString( "middle_initial" );
                System.out.println( "result: " + InfoData[7] );

                InfoData[8] = results.getString( "prim_address" );
                System.out.println( "result: " + InfoData[8] );

                InfoData[9] = results.getString( "prim_city" );
                System.out.println( "result: " + InfoData[9] );

                InfoData[10] = results.getString( "prim_state_cd" );
                System.out.println( "result: " + InfoData[10] );

                InfoData[11] = results.getString( "prim_zip_cd" );
                System.out.println( "result: " + InfoData[11] );

                InfoData[12] = results.getString( "prim_zip_cd_ext" );
                System.out.println( "result: " + InfoData[12] );

                InfoData[13] = results.getString( "sec_address" );
                System.out.println( "result: " + InfoData[13] );

                InfoData[14] = results.getString( "sec_city" );
                System.out.println( "result: " + InfoData[14] );

                InfoData[15] = results.getString( "sec_state_cd" );
                System.out.println( "result: " + InfoData[15] );

                InfoData[16] = results.getString( "sec_zip_cd" );
                System.out.println( "result: " + InfoData[16] );

                InfoData[17] = results.getString( "sec_zip_cd_ext" );
                System.out.println( "result: " + InfoData[17] );

                InfoData[18] = results.getString( "prim_phone_num" );
                System.out.println( "result: " + InfoData[18] );

                InfoData[19] = results.getString( "sec_phone_num" );
                System.out.println( "result: " + InfoData[19] );

                InfoData[20] = results.getString( "fax_num" );
                System.out.println( "result: " + InfoData[20] );

                InfoData[21] = results.getString( "create_dt" );
                System.out.println( "result: " + InfoData[21] );

                InfoData[22] = results.getString( "modified_dt" );
                System.out.println( "result: " + InfoData[22] );

                InfoData[23] = results.getString( "comments" );
                System.out.println( "result: " + InfoData[23] );

                InfoData[24] = results.getString( "addtl_comments" );
                System.out.println( "result: " + InfoData[24] );

}
catch(SQLException e)
{
                System.out.println("Read transaction failed: " + e );
```

```
                                }

        return InfoData;
    }


    // Inserts the data for the account object
    public synchronized String insertAccount()
    {
        String sMessage = "";

                                String sqlQuery = "INSERT INTO accounts( acct_num, pin_num, acct_status_cd, acct_type_cd, ssn,
last_name, first_name, middle_initial, prim_address, prim_city, prim_state_cd, prim_zip_cd, prim_zip_cd_ext, sec_address, sec_city,
sec_state_cd, sec_zip_cd, sec_zip_cd_ext, prim_phone_num, sec_phone_num, fax_num, create_dt, modified_dt, comments,
addtl_comments ) ";
                                sqlQuery = sqlQuery + "VALUES( '" + Data[0] + "', '" + Data[1] + "', '" + Data[2] + "', '" + Data[3] +
"', '" + Data[4] + "', '" + Data[5] + "', '" + Data[6] + "', '" + Data[7] + "', '" + Data[8] + "', '" + Data[9] + "', '" + Data[10] + "', '" +
Data[11] + "', '" + Data[12] + "', '" + Data[13] + "', '" + Data[14] + "', '" + Data[15] + "', ";
                                sqlQuery = sqlQuery + Data[16] + "', '" + Data[17] + "', '" + Data[18] + "', '" + Data[19] + "', '" +
Data[20] + "', '" + Data[21] + "', '" + Data[22] + "', '" + Data[23] + "', '" + Data[24] + "')";

                                System.out.println( "Sql query being sent to the db is: " + sqlQuery );

        int rows = dbSvr.runDBInOrUpQuery( sqlQuery );

        sMessage = "Number of rows affected: " + String.valueOf( rows );

        return sMessage;
    }

    // Updates the data for the account object
    public synchronized String updateAccount()
    {
        String sMessage = "";

        String sqlQuery = "UPDATE accounts SET pin_num = '" + Data[1] + "', acct_status_cd = '" + Data[2] + "', acct_type_cd = '" +
Data[3] + "', ssn = '" + Data[4] + "', last_name = '" + Data[5] + "', first_name = '" + Data[6] + "', middle_initial = '" + Data[7] + "',
prim_address = '" + Data[8] + "', prim_city = '" + Data[9] + "', ";
                                sqlQuery = sqlQuery + "prim_state_cd = '" + Data[10] + "', prim_zip_cd = '" + Data[11] + "',
prim_zip_cd_ext = '" + Data[12] + "', sec_address = '" + Data[13] + "', sec_city = '" + Data[14] + "', sec_state_cd = '" + Data[15] + "',
sec_zip_cd = '" + Data[16];
                                sqlQuery = sqlQuery + "', sec_zip_cd_ext = '" + Data[17] + "', prim_phone_num = '" + Data[18] + "',
sec_phone_num = '" + Data[19] + "', fax_num = '" + Data[20] + "', create_dt = '" + Data[21] + "', modified_dt = '" + Data[22] + "',
comments = '" + Data[23] + "', addtl_comments = '" + Data[24] + "' WHERE acct_num = '" + Id + "'";

                                System.out.println( "Sql query being sent to the db is: " + sqlQuery );

        int rows = dbSvr.runDBInOrUpQuery( sqlQuery );

        sMessage = "Number of rows affected: " + String.valueOf( rows );

        return sMessage;
    }

}
// End of VBAccountImpl
```

```
// Michelle McKeller - Graduate Project
// Corba - VisiBroker
// VBTransactionImpl

import java.sql.*;
import VBBankObjects.*;

public class VBTransactionImpl extends VBBankObjects._VBTransactionImplBase {
   private String Id;
   private String Data[];
            private VBDBServer dbSvr;

   // public constructor
   public VBTransactionImpl( String sId, VBDBServer mydbSvr )
   {
      this.Id = sId;
                        this.dbSvr = mydbSvr;
   }

   // overloaded constructor
   public VBTransactionImpl( String sId, String [] sData, VBDBServer mydbSvr )
   {
      this.Id = sId;
      this.Data = sData;
                        this.dbSvr = mydbSvr;
   }

   // Gets data for the transaction object
   public synchronized String [] readTransaction( )
   {
      String [] InfoData = new String [ 4 ];

      String sqlQuery = "SELECT trans_id, acct_num, trans_type_cd, trans_amt, create_dt FROM transactions WHERE acct_num = '"
+ Id + "' AND trans_id = (SELECT MAX(trans_id) FROM transactions WHERE acct_num = '" + Id + "')";

                        System.out.println( "Sql query being sent to the db is: " + sqlQuery );

      ResultSet results = dbSvr.runDBReadQuery( sqlQuery );

                        if ( results == null )
                        {
                                 InfoData[0] = "";
                        }else
                        {
                                 try
                                 {
                                          results.next();
                                          InfoData[0] = results.getString( "acct_num" );
                                          System.out.println( "results: " + InfoData[0]);

                                          InfoData[1] = results.getString( "trans_type_cd" );
                                          System.out.println( "results: " + InfoData[1]);

                                          InfoData[2] = results.getString( "trans_amt" );
                                          System.out.println( "results: " + InfoData[2]);

                                          InfoData[3] = results.getString( "create_dt" );
                                          System.out.println( "results: " + InfoData[3]);

                                 }
                                 catch(SQLException e)
                                 {
                                          System.out.println("Read transaction failed: " + e );
                                 }
                        }

      return InfoData;
   }
```

```
        // Inserts the data for the transaction object
        public synchronized String insertTransaction()
        {
                String sMessage = "";

                String sqlQuery = "INSERT INTO transactions(trans_id, acct_num, trans_type_cd, trans_amt, create_dt)
VALUES (trans_id_sequence.NEXTVAL, '" + Data[0] + "', '" + Data[1] + "', '" + Data[2] + "', '" + Data[3] + "')";

                System.out.println( "Sql query being sent to the db is: " + sqlQuery );

                int rows = dbSvr.runDBInOrUpQuery( sqlQuery );

                sMessage = "Number of rows affected: " + String.valueOf( rows );

                return sMessage;
        }

        // Updates the data for the transaction object
        public synchronized String updateTransaction()
        {
                String sMessage = "";

                String sqlQuery = "UPDATE transactions SET trans_type_cd = '" + Data[1] + "', trans_amt = '" + Data[2] + "',
create_dt = '" + Data[3] + "' WHERE acct_num = '" + Data[0] + "' AND trans_id = (SELECT MAX(trans_id) FROM transactions
WHERE acct_num = '" + Data[0] + "')";

                System.out.println( "Sql query being sent to the db is: " + sqlQuery );

                int rows = dbSvr.runDBInOrUpQuery( sqlQuery );

                sMessage = "Number of rows affected: " + String.valueOf( rows );

                return sMessage;
        }
}

// End of VBTransactionImpl
```

```
// Michelle McKeller - Graduate Project
// Corba - VisiBroker
// VBBank.idl

module VBBankObjects{
  exception Unknown{};

        typedef string VBAcct[25];
        typedef string VBTrans[4];

        interface VBTransaction {
                // Reads the transaction data for a given account number
                VBTrans readTransaction() raises(Unknown);

                // Inserts the transaction data into the database
    // for a given account number
                string insertTransaction() raises(Unknown);

                // Updates the transaction data in the database
    // for a given account number and trans id
                string updateTransaction() raises(Unknown);
        };

        interface VBAccount {

                //Gets Last made transaction for an Account
                VBTransaction getTransaction(in string sId) raises(Unknown);

                //Puts Last made transaction for an Account
                VBTransaction putTransaction(in string sId, in VBTrans objItem) raises(Unknown);

                // Reads the account data for a given account number
                VBAcct readAccount() raises(Unknown);

                // Inserts the account data into the database
    // for a given account number
                string insertAccount() raises(Unknown);

                // Updates the account data in the database
    // for a given account number
                string updateAccount() raises(Unknown);
        };

        interface VBManager {
                // Gets an account object
                VBAccount getAccount(in string Id) raises(Unknown);

                // Puts an account object
                VBAccount putAccount(in string Id, in VBAcct objItem) raises(Unknown);
        };


};

// End of VBBank.idl
```

Makefile for VisiBroker:

```
.SUFFIXES: .java .class .idl .module

.java.class:
	vbjc $<

.idl.module:
	idl2java -boa $<
	touch $@

default: all

clean:
	rm -rf VBBankObjects
	rm -f *.class *.tmp *.module *~

IDLS = \
	VBBank.idl

MODULES = $(IDLS:.idl=.module)

SRCS = \
	VBTransactionImpl.java \
	VBAccountImpl.java \
	VBManagerImpl.java \
	VBDBServer.java \
	VBServer.java \
	VBClientThread.java \
	VBClient.java

CLASSES = $(SRCS:.java=.class)

all:	$(MODULES) $(CLASSES)
```

# APPENDIX N

## Orbix 2000 Application

```
// Michelle McKeller - Graduate Project
// Corba - Orbix 2000
// client

package OBBankObjects;

import java.io.*;
import java.util.*;
import org.omg.CORBA.*;
import org.omg.CosNaming.*;
import org.omg.CosNaming.NamingContextPackage.*;
import java.lang.*;
import java.net.MalformedURLException;


public class client
{
        public static void main(String[] args)
        {
                try
                {
                        String typeTest = "";
                        String numClients = "";
                        String Url = "";
                        String fileName = "";
                        String inLine = "";
                        File name;
                        boolean bToken = false;
                        int iNum = 0;
                        int iIndex = 0;
                        int iTokenCt = 0;

                        //Retrieving command line arguments
                        if ( args.length < 6 )
                        {
                                System.out.println("At least 6 command-line arguments are needed.");
                                System.exit(0);
                        }

                        // Type of test we're running, read, insert, or update
                        typeTest = args[2].trim().toUpperCase();
                        System.out.println("typeTest: " + typeTest );

                        // Number of Clients running
                        numClients = args[3].trim();
                        System.out.println( "numClients: " + numClients );

                        // Convert from string to int we'll need this value for, for loops
                        iNum = Integer.valueOf(numClients).intValue();
                        System.out.println("Number of Clients running will be: " + iNum);

                        // Name of text file we'll be using for the test
                        fileName = args[4].trim();
```

```
System.out.println( "fileName: " + fileName );

// URL should be in the format of "http://neptune.cocse.unf.edu:2400/mmckeller/Orbix/server"
Url = args[5].trim();
System.out.println( "URL: " + Url );


// Reading in data from file given on command line
// However many clients are supposed to run, that is how
// many lines we will read, one for each client, then
// spawn a client thread.
System.out.println( "Searching for file entered on command line..." );

name = new File( fileName );

if( name.exists() )
{

        // Create and initialize the ORB
        org.omg.CORBA.ORB orb = ORB.init( args, null );

        org.omg.CORBA.Object objRef = orb.resolve_initial_references("NameService");

        org.omg.CosNaming.NamingContext ncRef =
                org.omg.CosNaming.NamingContextHelper.narrow(objRef);

        org.omg.CosNaming.NameComponent nc = new NameComponent(Url, "");

        NameComponent path[] = {nc};

        OBBankObjects.OBBankObjects.OBManager mgr =
                OBBankObjects.OBBankObjects.OBManagerHelper.narrow(
ncRef.resolve(path));

        System.out.println( "Getting manager object" );

        if ( mgr == null )
        {
                System.out.println( "Error: Manager object equals null.");
                System.exit(0);
        }

        System.out.println( "File found, reading in 1 line per client thread.");
        FileReader inFile = new FileReader ( name );
        BufferedReader inBuff = new BufferedReader( inFile );

        // Reading in one line per client requested, then spawning the client thread
        for ( iIndex = 0; iIndex < iNum; iIndex++ )
        {
                if ( (inLine = inBuff.readLine()) != null )
                {
                        StringTokenizer st = new StringTokenizer( inLine, "\t\n\r", bToken
);


                        String [] items = new String [ ( st.countTokens() ) ] ;

                        try
                        {
                                iTokenCt = 0;
                                while ( st.hasMoreTokens() )
                                {
                                        String word = st.nextToken();

                                        items[ iTokenCt ] = word;


                                        iTokenCt = iTokenCt + 1;

                                }
                        }catch( NoSuchElementException e )
```

```
                                                {
                                                        // If any of the fields were empty, this is going
                                                        // to get thrown, but thats acceptable
                                                }
                                                // Spawning a client thread and passing it the string array of data
                                                // that we read in

                                                new OBBankObjects.OBClientThread( args, typeTest,
String.valueOf( iIndex + 1), Url, items, mgr ).start();
                                        }else
                                        {
                                                // The file has to have as many rows in it as there are clients listed
                                                // on the command line if not, this error occurs
                                                System.out.println( "File does not support number of clients on
command line." );
                                                System.exit(0);
                                        }
                                }
                        }else
                        {
                                System.out.println( "File Name entered on command line does not exist.");
                                System.exit(0);
                        }
                }catch( Exception e)
                {
                        System.out.println("An error occurred processing request.");
                }
        }
}
// End of client
```

```
// Michelle McKeller - Graduate Project
// Corba - Orbix 2000
// OBClientThread

package OBBankObjects;

import org.omg.CORBA.*;
import org.omg.CosNaming.*;
import org.omg.CosNaming.NamingContextPackage.*;
import java.lang.*;
import java.io.*;
import java.net.MalformedURLException;
import java.util.*;

public class OBClientThread extends Thread
{
        String aTypeTest, aUrl, aClientNum;
        String [] aArgs;
        String [] aDataItems;
        OBBankObjects.OBBankObjects.OBManager mgr = null;

  public OBClientThread(String [] args, String typeTest, String clientNum, String Url, String [] dataItems,
OBBankObjects.OBBankObjects.OBManager obMgr)
        {
                aArgs = args;
          aTypeTest = typeTest;
                aClientNum = clientNum;
                aUrl = Url;
                aDataItems = dataItems;
                mgr = obMgr;
  }

  public void run()
        {
                long time1 = 0;
                long time2 = 0;

                try
                {
                        FileWriter Log = new FileWriter( "dbDataLog.txt", true );
                        BufferedWriter w = new BufferedWriter(Log);


                        OBBankObjects.OBBankObjects.OBAccount acct = null;
                        OBBankObjects.OBBankObjects.OBTransaction trans = null;
                        String [] acctData = null;
                        String [] transData = null;
                        String rowsAffected = "";
                        int i = 0;

                        // Deciding whether to read, insert, or update
                        if ( aTypeTest.equals( "READ" ) )
                        {
                                // Reads an account record
                                // mimics a heavy load

                                // Start timer
                                time1 = System.currentTimeMillis();

                                // Reading the main account record
                                acct = mgr.getAccount( aDataItems[0] );

                                if ( acct == null )
                                {
                                        // Stop timer
                                        time2 = System.currentTimeMillis();

                                        System.out.println( "Read account failed constructing account object." );
                                }else
                                {
```

```
                    acctData = acct.readAccount();

                    // Stop timer
                    time2 = System.currentTimeMillis();

                    if ( acctData[0].trim() == "" )
                    {
                            System.out.println( "Read account failed reading record from the
database." );
                    }else
                    {
                            w.write( "Returned data was: " );

                            for ( i = 0; i < 25; i++ )
                            {
                                    w.write( acctData[i] );
                                    w.newLine();

                                    System.out.println( "Account data returned from server
was: " + acctData[i] );
                            }
                    }
            }

        }else if ( aTypeTest.equals( "INSERT" ) )
        {
            // Inserts a new transaction record
            // mimics a light load

            // Start timer
            time1 = System.currentTimeMillis();

            // Inserting a transaction record
            acct = mgr.getAccount( aDataItems[0] );

            if ( acct == null )
            {
                    // Stop timer
                    time2 = System.currentTimeMillis();

                    System.out.println( "Insert transaction failed constructing account object." );

            }else
            {
                    trans = acct.putTransaction( aDataItems[0], aDataItems );
                    rowsAffected = trans.insertTransaction();

                    // Stop timer
                    time2 = System.currentTimeMillis();

                    if ( rowsAffected.trim() == "" )
                    {
                            System.out.println( "Transaction insert failed when trying to insert
record into the database." );
                    }else
                    {
                            w.write( "Rows affected: " + rowsAffected );
                            w.newLine();

                            System.out.println( "Rows affected: " + rowsAffected );
                    }
            }


        }else if ( aTypeTest.equals( "UPDATE" ) )
        {
            // Updates an existing account record
            // mimics a heavy load

            // Start timer
```

```
                time1 = System.currentTimeMillis();

                acct = mgr.putAccount( aDataItems[0], aDataItems );

                if ( acct == null )
                {
                        // Stop timer
                        time2 = System.currentTimeMillis();

                        System.out.println( "Update account failed constructing account object.");

                }else
                {
                        rowsAffected = acct.updateAccount();

                        // Stop timer
                        time2 = System.currentTimeMillis();

                        if ( rowsAffected == "" )
                        {
                                System.out.println( "Update account failed when trying to update
the record in the database." );
                        }else
                        {
                                w.write( "Rows affected: " + rowsAffected );
                                w.newLine();

                                System.out.println( "Rows affected: " + rowsAffected );
                        }
                }

        }else if ( aTypeTest.equals( "INSERTACCT" ) )
        {
                // Inserts a new account record
                // mimics a heavy load

                // Start timer
                time1 = System.currentTimeMillis();

                // Inserting a new account
                acct = mgr.putAccount( aDataItems[0], aDataItems );

                if ( acct == null )
                {
                        // Stop timer
                        time2 = System.currentTimeMillis();

                        System.out.println( "Insert account failed constructing account object.");

                }else
                {
                        rowsAffected = acct.insertAccount();

                        // Stop timer
                        time2 = System.currentTimeMillis();

                        if ( rowsAffected == "" )
                        {
                                System.out.println( "Insert Account failed when trying to insert
record into the database." );
                        }else
                        {
                                w.write( "Rows affected: " + rowsAffected );
                                w.newLine();

                                System.out.println( "Rows affected: " + rowsAffected );
                        }
                }

        }else if ( aTypeTest.equals( "READTRANS" ) )
```

```
{
		// This retrieves the last transaction record for the account
		// mimics a light load

		// Start timer
		time1 = System.currentTimeMillis();

		acct = mgr.getAccount( aDataItems[0] );

		if ( acct == null )
		{
			// Stop timer
			time2 = System.currentTimeMillis();

			System.out.println( "Read transaction failed constructing the account object."
);
		}else
		{
			trans = acct.getTransaction( aDataItems[0] );
			if ( trans == null )
			{
				// Stop timer
				time2 = System.currentTimeMillis();

				System.out.println( "Read transaction failed constructing
transaction object." );
			}else
			{
				transData = trans.readTransaction();

				// Stop timer
				time2 = System.currentTimeMillis();

				if ( transData[0].trim() == "" )
				{
					System.out.println( "Read transaction failed reading the
record from the database." );
				}else
				{
					w.write( "Returned data was: " );

					for ( i = 0; i < 4; i++ )
					{
						w.write( transData[i] );
						w.newLine();

						System.out.println( "Transaction data
returned from server was: " + transData[i] );
					}
				}
			}
		}
	}else if ( aTypeTest.equals( "UPDATETRANS" ) )
	{
		// Start timer
		time1 = System.currentTimeMillis();

		acct = mgr.getAccount( aDataItems[0] );

		if ( acct == null )
		{
			// Stop timer
			time2 = System.currentTimeMillis();

			System.out.println( "Update transaction failed constructing the account
object." );
		}else
		{
			trans = acct.putTransaction( aDataItems[0], aDataItems );
```

```
                                        if ( trans == null )
                                        {
                                                // Stop timer
                                                time2 = System.currentTimeMillis();

                                                System.out.println( "Update transaction failed constructing the
transaction object." );
                                        }else
                                        {
                                                rowsAffected = trans.updateTransaction();

                                                // Stop timer
                                                time2 = System.currentTimeMillis();

                                                if ( rowsAffected.trim() == "" )
                                                {
                                                        System.out.println( "Update transaction failed when
updating the record in the database." );
                                                }else
                                                {
                                                        w.write( "Rows affected: " + rowsAffected );
                                                        w.newLine();

                                                        System.out.println( "Rows Affected: " + rowsAffected
);
                                                }
                                        }
                                }
                        }

                        w.close();

                }catch( IOException e )
                {
                        System.out.println( "An error occurred writing to the dbDataLog file." );
                        e.printStackTrace();
                }catch( Exception e )
                {
                        System.out.println( "An error has occurred in the Client." );
                        e.printStackTrace();
                }

                try
                {
                        FileWriter Log = new FileWriter( "timelog.txt", true );
                        BufferedWriter w = new BufferedWriter(Log);
                        w.write( "Time spent on Client " + aClientNum + ": " + ( time2 - time1 ) + " Milliseconds" );
                        w.newLine();
                        w.close();
                }catch(IOException e)
                {
                        System.out.println( "Problem occurred creating log file: " + e );
                }

                System.out.println( "Time spent on Client " + aClientNum + ": " + (time2 - time1) + " Milliseconds" );

        }
}

// End of OBClientThread
```

```
// Michelle McKeller - Graduate Project
// Corba - Orbix 2000
// server

package OBBankObjects;

import org.omg.*;
import org.omg.CORBA.*;
import org.omg.PortableServer.*;
import org.omg.CosNaming.*;

import java.text.DateFormat;
import java.io.IOException;
import java.net.MalformedURLException;

public class server {

  public static ORB global_orb = null;

  // create_simple_poa() -- Create a POA for simple servant management.
  static POA create_simple_poa(
    String        poa_name,
    POA           parent_poa,
    POAManager    poa_manager
  )
  {
    // Create a policy list. Policies not set in the list get default values.
    org.omg.CORBA.Policy[] policies = new org.omg.CORBA.Policy[1];
    int i = 0;
    POA new_poa = null;

    // Make the POA multi-threaded.
    // Note: application server code may be called in multiple threads
    // so it must be thread safe.
    policies[i++] = parent_poa.create_thread_policy(ThreadPolicyValue.ORB_CTRL_MODEL);
    if(! (i==1) )
    {
      System.out.println("Policy creation failed");
      System.exit(1);
    }
    try
    {
      new_poa = parent_poa.create_POA(poa_name, poa_manager, policies);
    } catch (org.omg.PortableServer.POAPackage.AdapterAlreadyExists ex)
    {
      System.out.println("Failed to create the POA with exception : " +ex.toString());
      System.exit(1);
    } catch (org.omg.PortableServer.POAPackage.InvalidPolicy ex)
    {
      System.out.println("Failed to create the POA with exception : " +ex.toString());
      System.exit(1);
    }

    return new_poa;

  }


  // main() -- set up a POA, create and export object references.
  public static void main(String args[])
  {
    Servant the_OBBankObjects_OBManager = null;
    org.omg.CORBA.Object tmp_ref = null;

                    OBBankObjects.OBBankObjects.OBDBServer dbServer = null;
                    String ServerUrl = "";
                    String Driver = "";
                    String dbURL = "";
                    String User = "";
                    String Password = "";
```

```
try
{
                if ( args.length < 7 )
                {
                        System.out.println("At least 7 command-line arguments are needed.");
                        System.exit(0);
                }

                ServerUrl = args[2].trim();
                System.out.println( "ServerUrl: " + ServerUrl );
                Driver = args[3].trim();
                System.out.println( "Driver: " + Driver );
                dbURL = args[4].trim();
                System.out.println("dbUrl: " + dbURL );
                User = args[5].trim();
                System.out.println("User: " + User );
                Password = args[6].trim();
                System.out.println( "pass: " + Password);

// Initialise the ORB and Root POA.
System.out.println( "Initializing the ORB" );

try
{
  global_orb = ORB.init(args, null);
  tmp_ref = global_orb.resolve_initial_references("RootPOA");
                        System.out.println( "Created root POA" );
} catch (org.omg.CORBA.ORBPackage.InvalidName ex)
{
  System.out.println( "Caught an unexpected exception while resolving to the RootPOA : " + ex.toString());
  System.exit(1);
}

                // Create a Database Server Object
                try{
                        dbServer = new OBBankObjects.OBBankObjects.OBDBServer( Driver, dbURL, User, Password );
                        System.out.println( "Created oracle database connection" );
                }catch( Exception e )
                {
                        System.out.println( "Failing to establish oracle connection: " );
                        e.printStackTrace();
                }

POA root_poa = POAHelper.narrow(tmp_ref);
POAManager root_poa_manager = root_poa.the_POAManager();

                System.out.println( "Connecting to the name server..." );

tmp_ref = global_orb.resolve_initial_references("NameService");
org.omg.CosNaming.NamingContext default_context =
  org.omg.CosNaming.NamingContextHelper.narrow(tmp_ref);

if(default_context == null)
{
  System.out.println("Failed to get reference to the NameService");
  System.exit(1);
}

                System.out.println( "Name Server Connected");

// Now create our own POA.

                System.out.println( "Creating new poa..." );

POA my_poa = create_simple_poa("my_poa", root_poa, root_poa_manager);

                System.out.println( "POA created." );
```

```
byte [] oid;
org.omg.CosNaming.NameComponent name;

try{

  // Create a servant for interface OBBankObjects.OBManager.
                        System.out.println( "Creating manager servant for use...");

  the_OBBankObjects_OBManager = new
    OBBankObjects.OBBankObjects.OBManagerImpl(my_poa, dhServer);

                        oid = my_poa.activate_object(the_OBBankObjects_OBManager);

                        System.out.println( "Manager servant created." );

  tmp_ref = my_poa.id_to_reference(oid);

                        name = new NameComponent(ServerUrl, "");
                        NameComponent path[] = {name};

                        default_context.rebind(path, tmp_ref);

                        System.out.println( "Binded to Manager Impl Object" );

} catch (org.omg.PortableServer.POAPackage.ServantAlreadyActive ex)
{
  System.out.println("Caught an exception while trying to activate an object : " + ex.toString());
  System.exit(1);
} catch (org.omg.PortableServer.POAPackage.WrongPolicy ex)
{
  System.out.println("Caught an exception while trying to activate an object : " + ex.toString());
  System.exit(1);
} catch (org.omg.PortableServer.POAPackage.ObjectNotActive ex)
{
  System.out.println("Caught an exception while trying to create a reference : " + ex.toString());
  System.exit(1);
                }

// Activate the POA Manager.
try
{
  root_poa_manager.activate();

                        System.out.println( "Activated root poa manager" );

} catch (org.omg.PortableServer.POAManagerPackage.AdapterInactive ex)
{
  System.out.println("Failed while trying to activate the root poa manager : " + ex.toString());
  System.exit(1);
}

//Let the ORB process requests.
System.out.println("Waiting for requests..." );
global_orb.run();
}
catch (Exception e)
{
  System.out.println("Unexpected CORBA exception: " + e.toString());
}

// Ensure that the ORB is properly shutdown and cleaned up.
try
{
  global_orb.shutdown(true);
}
catch (Exception e)
{
  // Do nothing.
}
return;
```

```
  }
}

// End of server
```

```
// Michelle McKeller - Graduate Project
// Corba - Orbix 2000
// OBDBServer

package OBBankObjects.OBBankObjects;

import OBBankObjects.*;
import java.net.*;
import java.sql.*;
import java.io.*;

public class OBDBServer {

        String sDriver = "";
        String sdbURL = "";
        String sUser = "";
        String sPassword = "";
        String dbaseURL = "";
        Connection dbConnection;
        Statement query;

        // Constructor
        public OBDBServer( String Driver, String dbURL, String User, String Password )
        {
                sDriver = Driver; //oracle.jdbc.driver.OracleDriver
                sdbURL = dbURL; // "jdbc:oracle:thin:@manatee.coese.unf.edu:1521:sid1
                sUser = User;
                sPassword = Password;

                initialize();
        }

        public void initialize()
        {
                try
                {
                        Class.forName( sDriver );
                        System.out.println( "Looking for oracle driver." );
                        dbaseURL = sdbURL;
                        dbConnection = DriverManager.getConnection(dbaseURL, sUser, sPassword);
                        System.out.println( "Making db connection.");
                        query = dbConnection.createStatement();

    }
    catch(SQLException e)
                {
      System.out.println("failed");
                        e.printStackTrace();
    }
                catch(ClassNotFoundException e)
                {
                        System.out.println("failed cl");
                }

        }

        public int runDBInOrUpQuery( String sSqlQuery )
        {
                int rows = 0;

                try
                {
                        rows = query.executeUpdate( sSqlQuery );

    }
    catch(SQLException e)
                {
      System.out.println("failed");
                        e.printStackTrace();
    }
```

```
                return rows;
        }

    public ResultSet runDBReadQuery( String sSqlQuery )
        {
                ResultSet results = null;

      try
                {
                        results = query.executeQuery( sSqlQuery );
      }
      catch(SQLException e)
                {
        System.out.println("failed");
                        e.printStackTrace();
      }

                return results;
        }
}

// End of OBDBServer
```

```
// Michelle McKeller - Graduate Project
// Corba - Orbix 2000
// OBManagerImpl

package OBBankObjects.OBBankObjects;

import OBBankObjects.*;
import org.omg.CORBA.ORB;
import org.omg.PortableServer.*;
import org.omg.CORBA.StringHolder;


public class OBManagerImpl
   extends OBManagerPOA
{
   org.omg.PortableServer.POA m_poa = null;
            OBBankObjects.OBBankObjects.OBDBServer dbSvr;
            String myMgrPoa = "";

   // Constructor of OBManagerImpl
   public OBManagerImpl(
      org.omg.PortableServer.POA the_poa, OBBankObjects.OBBankObjects.OBDBServer mydbServer
   )
   {
      m_poa = the_poa;
                        this.dbSvr = mydbServer;

                        try
                        {

                                 org.omg.CORBA.Policy[] policies = new org.omg.CORBA.Policy[1];

                                 int i = 0;
                                 policies[i++] = the_poa.create_thread_policy(ThreadPolicyValue.ORB_CTRL_MODEL);

                                 myMgrPoa = String.valueOf( i );

                                 m_poa = the_poa.create_POA( myMgrPoa, the_poa.the_POAManager(), policies);

                                 //m_poa = the_poa.create_POA( "ManagerPOA", the_poa.the_POAManager(), policies);

                        }catch (Exception e)
                        {
                                 System.out.println( "Exception occurred in the manager impl: " );
                                 e.printStackTrace();
                        }
   }

   public OBBankObjects.OBBankObjects.OBAccount getAccount(
         java.lang.String          Id
   )
   throws org.omg.CORBA.SystemException,
    OBBankObjects.OBBankObjects.Unknown
    {
                        OBBankObjects.OBBankObjects.OBAccount myAcct = null;

                        try
                        {
                                 OBBankObjects.OBBankObjects.OBAccountImpl myAcctImpl = new
OBBankObjects.OBBankObjects.OBAccountImpl( m_poa, Id, dbSvr);

                                 long time1 = 0;
                                 String xId = "";

                                 time1 = System.currentTimeMillis();

                                 xId = Id + String.valueOf( time1 );

                                 m_poa.activate_object_with_id( xId.getBytes(), myAcctImpl );
```

```
                        // get the CORBA Object Reference for this servant
                        myAcct = OBBankObjects.OBBankObjects.OBAccountHelper.narrow(
m_poa.servant_to_reference( myAcctImpl ));

                }catch (Exception c)
                {
                        System.out.println( "Exception occurred getting account object: " );
                        e.printStackTrace();
                }

                return myAcct;
  }

  public OBBankObjects.OBBankObjects.OBAccount putAccount(
        java.lang.String        Id,
        java.lang.String[]      Data
  )
  throws org.omg.CORBA.SystemException,
   OBBankObjects.OBBankObjects.Unknown
   {
                OBBankObjects.OBBankObjects.OBAccount myAcct = null;

                try
                {
                        OBBankObjects.OBBankObjects.OBAccountImpl myAcctImpl = new
OBBankObjects.OBBankObjects.OBAccountImpl( m_poa, Id, Data, dbSvr);

                        long time1 = 0;
                        String xId = "";

                        time1 = System.currentTimeMillis();

                        xId = Id + String.valueOf( time1 );

                        m_poa.activate_object_with_id( xId.getBytes(), myAcctImpl );

                        // get the CORBA Object Reference for this servant
                        myAcct = OBBankObjects.OBBankObjects.OBAccountHelper.narrow(
m_poa.servant_to_reference( myAcctImpl ));

                }catch (Exception e)
                {
                        System.out.println( "Exception occurred putting account object: " );
                        e.printStackTrace();
                }

                return myAcct;
  }

  public org.omg.PortableServer.POA _default_POA()
  {
    return m_poa;
  }

}

// End of OBManagerImpl
```

```
// Michelle McKeller - Graduate Project
// Corba - Orbix 2000
// OBAccountImpl

package OBBankObjects.OBBankObjects;

import OBBankObjects.*;
import java.sql.*;
import org.omg.CORBA.ORB;
import org.omg.PortableServer.*;
import org.omg.CORBA.StringHolder;


public class OBAccountImpl
  extends OBAccountPOA
{
  org.omg.PortableServer.POA m_poa = null;
  private String Id;
  private String Data[];
        private OBBankObjects.OBBankObjects.OBDBServer dbSvr;

  //Constructor of OBAccountImpl
  public OBAccountImpl(
    org.omg.PortableServer.POA the_poa, String sId, OBBankObjects.OBBankObjects.OBDBServer mydbSvr
  )
  {
    m_poa = the_poa;
    this.Id = sId;
                this.dbSvr = mydbSvr;

                try
                {
                        org.omg.CORBA.Policy[] policies = new org.omg.CORBA.Policy[1];

                        String myAcctPoa = "";
                        long time1 = 0;
                        int i = 0;

                        time1 = System.currentTimeMillis();

                        policies[i++] = the_poa.create_thread_policy(ThreadPolicyValue.ORB_CTRL_MODEL);

                        myAcctPoa = Id + String.valueOf( time1 );

                        m_poa = the_poa.create_POA( myAcctPoa, the_poa.the_POAManager(), policies);

                } catch ( Exception e )
                {
                        System.out.println( "Exception occurred in the account impl: " );
                        e.printStackTrace();
                }
  }

        // Overloaded Constructor
  public OBAccountImpl(
    org.omg.PortableServer.POA the_poa, String sId, String [] sData, OBBankObjects.OBBankObjects.OBDBServer mydbSvr
  )
  {
    // initialise all the fields in the object so that there are no null values
    m_poa = the_poa;
    this.Id = sId;
    this.Data = sData;
                this.dbSvr = mydbSvr;

                try
                {
                        org.omg.CORBA.Policy[] policies = new org.omg.CORBA.Policy[1];

                        String myAcctPoa = "";
                        long time1 = 0;
```

```
                int i = 0;

                time1 = System.currentTimeMillis();

                policies[i++] = the_poa.create_thread_policy(ThreadPolicyValue.ORB_CTRL_MODEL);

                myAcctPoa = Id + String.valueOf( time1 );

                m_poa = the_poa.create_POA( myAcctPoa, the_poa.the_POAManager(), policies);

        } catch ( Exception e )
        {
                System.out.println( "Exception occurred in the account impl: " );
                e.printStackTrace();
        }
    }

    // Used to get the transaction object
    public OBBankObjects.OBBankObjects.OBTransaction getTransaction(
        java.lang.String        sId
    )
    throws org.omg.CORBA.SystemException,
     OBBankObjects.OBBankObjects.Unknown
     {
        OBBankObjects.OBBankObjects.OBTransaction myTrans = null;

        try
        {
                OBBankObjects.OBBankObjects.OBTransactionImpl myTransImpl = new
OBBankObjects.OBBankObjects.OBTransactionImpl( m_poa, sId, dbSvr);

                long time1 = 0;
                String xId = "";

                time1 = System.currentTimeMillis();

                xId = Id + String.valueOf( time1 );

                m_poa.activate_object_with_id( xId.getBytes(), myTransImpl );

                // get the CORBA Object Reference for this servant
                myTrans = OBBankObjects.OBBankObjects.OBTransactionHelper.narrow(
m_poa.servant_to_reference( myTransImpl ));

        }catch (Exception e)
        {
                System.out.println( "Exception occurred creating transaction object: " );
                e.printStackTrace();
        }

        return myTrans;
    }

    // Used to put a transaction object
    public OBBankObjects.OBBankObjects.OBTransaction putTransaction(
        java.lang.String        sId,
        java.lang.String[]      sData
    )
    throws org.omg.CORBA.SystemException,
     OBBankObjects.OBBankObjects.Unknown
     {
        OBBankObjects.OBBankObjects.OBTransaction myTrans = null;

        try
        {
                OBBankObjects.OBBankObjects.OBTransactionImpl myTransImpl = new
OBBankObjects.OBBankObjects.OBTransactionImpl( m_poa, sId, sData, dbSvr);

                long time1 = 0;
                String xId = "";
```

```
                    time1 = System.currentTimeMillis();

                    xId = Id + String.valueOf( time1 );

                    m_poa.activate_object_with_id( xId.getBytes(), myTransImpl );

                    // get the CORBA Object Reference for this servant
                    myTrans = OBBankObjects.OBBankObjects.OBTransactionHelper.narrow(
m_poa.servant_to_reference( myTransImpl ));

                }catch (Exception e)
                {
                    System.out.println( "Exception occurred creating transaction object: " );
                    e.printStackTrace();
                }

                return myTrans;
    }

    // Used to read an account
    public synchronized java.lang.String[] readAccount(
    )
    throws org.omg.CORBA.SystemException,
     OBBankObjects.OBBankObjects.Unknown
     {
        String [] InfoData = new String [ 25 ];

        String sqlQuery = "SELECT acct_num, pin_num, acct_status_cd, acct_type_cd, ssn, last_name, first_name, middle_initial,
prim_address, prim_city, prim_state_cd, prim_zip_cd, prim_zip_cd_ext, sec_address, sec_city, sec_state_cd, sec_zip_cd,
sec_zip_cd_ext, prim_phone_num, sec_phone_num, fax_num, create_dt, modified_dt, comments, addtl_comments FROM accounts
WHERE acct_num = '" + Id + "'";

                    System.out.println( "Sql query being sent to the db is: " + sqlQuery );

        ResultSet results = dbSvr.runDBReadQuery( sqlQuery );

                    try
                    {
                        results.next();

                        InfoData[0] = results.getString( "acct_num" );
                        System.out.println( "result: " + InfoData[0] );

                        InfoData[1] = results.getString( "pin_num" );
                        System.out.println( "result: " + InfoData[1] );

                        InfoData[2] = results.getString( "acct_status_cd" );
                        System.out.println( "result: " + InfoData[2] );

                        InfoData[3] = results.getString( "acct_type_cd" );
                        System.out.println( "result: " + InfoData[3] );

                        InfoData[4] = results.getString( "ssn" );
                        System.out.println( "result: " + InfoData[4] );

                        InfoData[5] = results.getString( "last_name" );
                        System.out.println( "result: " + InfoData[5] );

                        InfoData[6] = results.getString( "first_name" );
                        System.out.println( "result: " + InfoData[6] );

                        InfoData[7] = results.getString( "middle_initial" );
                        System.out.println( "result: " + InfoData[7] );

                        InfoData[8] = results.getString( "prim_address" );
                        System.out.println( "result: " + InfoData[8] );

                        InfoData[9] = results.getString( "prim_city" );
                        System.out.println( "result: " + InfoData[9] );
```

```
                                        InfoData[10] = results.getString( "prim_state_cd" );
                                        System.out.println( "result: " + InfoData[10] );

                                        InfoData[11] = results.getString( "prim_zip_cd" );
                                        System.out.println( "result: " + InfoData[11] );

                                        InfoData[12] = results.getString( "prim_zip_cd_ext" );
                                        System.out.println( "result: " + InfoData[12] );

                                        InfoData[13] = results.getString( "sec_address" );
                                        System.out.println( "result: " + InfoData[13] );

                                        InfoData[14] = results.getString( "sec_city" );
                                        System.out.println( "result: " + InfoData[14] );

                                        InfoData[15] = results.getString( "sec_state_cd" );
                                        System.out.println( "result: " + InfoData[15] );

                                        InfoData[16] = results.getString( "sec_zip_cd" );
                                        System.out.println( "result: " + InfoData[16] );

                                        InfoData[17] = results.getString( "sec_zip_cd_ext" );
                                        System.out.println( "result: " + InfoData[17] );

                                        InfoData[18] = results.getString( "prim_phone_num" );
                                        System.out.println( "result: " + InfoData[18] );

                                        InfoData[19] = results.getString( "sec_phone_num" );
                                        System.out.println( "result: " + InfoData[19] );

                                        InfoData[20] = results.getString( "fax_num" );
                                        System.out.println( "result: " + InfoData[20] );

                                        InfoData[21] = results.getString( "create_dt" );
                                        System.out.println( "result: " + InfoData[21] );

                                        InfoData[22] = results.getString( "modified_dt" );
                                        System.out.println( "result: " + InfoData[22] );

                                        InfoData[23] = results.getString( "comments" );
                                        System.out.println( "result: " + InfoData[23] );

                                        InfoData[24] = results.getString( "addtl_comments" );
                                        System.out.println( "result: " + InfoData[24] );

                                }
                                catch(SQLException e)
                                {
                                        System.out.println("Read transaction failed: " + e );
                                }

        return InfoData;
}

// Used to insert a new account
public synchronized java.lang.String insertAccount(
)
throws org.omg.CORBA.SystemException,
 OBBankObjects.OBBankObjects.Unknown
 {
        String sMessage = "";

                                String sqlQuery = "INSERT INTO accounts( acct_num, pin_num, acct_status_cd, acct_type_cd, ssn,
last_name, first_name, middle_initial, prim_address, prim_city, prim_state_cd, prim_zip_cd, prim_zip_cd_ext, sec_address, sec_city,
sec_state_cd, sec_zip_cd, sec_zip_cd_ext, prim_phone_num, sec_phone_num, fax_num, create_dt, modified_dt, comments,
addtl_comments ) ";
                                sqlQuery = sqlQuery + "VALUES( '" + Data[0] + "', '" + Data[1] + "', '" + Data[2] + "', '" + Data[3] +
"', '" + Data[4] + "', '" + Data[5] + "', '" + Data[6] + "', '" + Data[7] + "', '" + Data[8] + "', '" + Data[9] + "', '" + Data[10] + "', '" +
Data[11] + "', '" + Data[12] + "', '" + Data[13] + "', '" + Data[14] + "', '" + Data[15] + "', '";
```

```
                                    sqlQuery = sqlQuery + Data[16] + "', '" + Data[17] + "', '" + Data[18] + "', '" + Data[19] + "', '" +
Data[20] + "', '" + Data[21] + "', '" + Data[22] + "', '" + Data[23] + "', '" + Data[24] + "')";

                                    System.out.println( "Sql query being sent to the db is: " + sqlQuery );

        int rows = dbSvr.runDBInOrUpQuery( sqlQuery );

        sMessage = "Number of rows affected: " + String.valueOf( rows );

        return sMessage;

    }

    // Used to update an existing account
    public synchronized java.lang.String updateAccount(
    )
    throws org.omg.CORBA.SystemException,
     OBBankObjects.OBBankObjects.Unknown
     {
        String sMessage = "";

        String sqlQuery = "UPDATE accounts SET pin_num = '" + Data[1] + "', acct_status_cd = '" + Data[2] + "', acct_type_cd = '" +
Data[3] + "', ssn = '" + Data[4] + "', last_name = '" + Data[5] + "', first_name = '" + Data[6] + "', middle_initial = '" + Data[7] + "',
prim_address = '" + Data[8] + "', prim_city = '" + Data[9] + "', ";
                                    sqlQuery = sqlQuery + "prim_state_cd = '" + Data[10] + "', prim_zip_cd = '" + Data[11] + "',
prim_zip_cd_ext = '" + Data[12] + "', sec_address = '" + Data[13] + "', sec_city = '" + Data[14] + "', sec_state_cd = '" + Data[15] + "',
sec_zip_cd = '" + Data[16];
                                    sqlQuery = sqlQuery + "', sec_zip_cd_ext = '" + Data[17] + "', prim_phone_num = '" + Data[18] + "',
sec_phone_num = '" + Data[19] + "', fax_num = '" + Data[20] + "', create_dt = '" + Data[21] + "', modified_dt = '" + Data[22] + "',
comments = '" + Data[23] + "', addtl_comments = '" + Data[24] + "' WHERE acct_num = '" + Id + "'";

                                    System.out.println( "Sql query being sent to the db is: " + sqlQuery );

        int rows = dbSvr.runDBInOrUpQuery( sqlQuery );

        sMessage = "Number of rows affected: " + String.valueOf( rows );

        return sMessage;
    }

    public org.omg.PortableServer.POA _default_POA()
    {
      return m_poa;
    }

}

// End of OBAccountImpl
```

```
// Michelle McKeller - Graduate Project
// Corba - Orbix 2000
// OBTransactionImpl

package OBBankObjects.OBBankObjects;

import OBBankObjects.*;
import java.sql.*;
import org.omg.CORBA.ORB;
import org.omg.PortableServer.*;
import org.omg.CORBA.StringHolder;

public class OBTransactionImpl
  extends OBTransactionPOA
{
  org.omg.PortableServer.POA m_poa = null;
  private String Id;
  private String Data[];
        private OBBankObjects.OBBankObjects.OBDBServer dbSvr;

        // Constructor
  public OBTransactionImpl(
    org.omg.PortableServer.POA the_poa, String sId, OBDBServer mydbSvr
  )
  {
    m_poa = the_poa;
    this.Id = sId;
                this.dbSvr = mydbSvr;

                try
                {
                        String myTransPoa = "";
                        long time1 = 0;
                        int i = 0;

                        org.omg.CORBA.Policy[] policies = new org.omg.CORBA.Policy[1];

                        policies[i++] = the_poa.create_thread_policy(ThreadPolicyValue.ORB_CTRL_MODEL);

                        time1 = System.currentTimeMillis();

                        myTransPoa = Id + String.valueOf( time1 );

                        m_poa = the_poa.create_POA( myTransPoa, the_poa.the_POAManager(), policies);

                }catch (Exception e)
                {
                        System.out.println( "Exception occurred in the transaction impl: " );
                        e.printStackTrace();
                }
 }

  // Overloaded Constructor
  public OBTransactionImpl(
    org.omg.PortableServer.POA the_poa, String sId, String [] sData, OBDBServer mydbSvr
  )
  {
    m_poa = the_poa;
    this.Id = sId;
    this.Data = sData;
                this.dbSvr = mydbSvr;

                try
                {
                        String myTransPoa = "";
                        long time1 = 0;
                        int i = 0;

                        org.omg.CORBA.Policy[] policies = new org.omg.CORBA.Policy[1];
```

```
                    policies[i++] = the_poa.create_thread_policy(ThreadPolicyValue.ORB_CTRL_MODEL);

                    time1 = System.currentTimeMillis();

                    myTransPoa = Id + String.valueOf( time1 );

                    m_poa = the_poa.create_POA( myTransPoa, the_poa.the_POAManager(), policies);

                }catch (Exception e)
                {
                    System.out.println( "Exception occurred in the transaction impl: " );
                    e.printStackTrace();
                }
    }

    // Used to read a transaction record for an account
    public synchronized java.lang.String[] readTransaction(
    )
    throws org.omg.CORBA.SystemException,
     OBBankObjects.OBBankObjects.Unknown
     {
        String [] InfoData = new String [ 4 ];

        String sqlQuery = "SELECT trans_id, acct_num, trans_type_cd, trans_amt, create_dt FROM transactions WHERE acct_num = '"
+ Id + "' AND trans_id = (SELECT MAX(trans_id) FROM transactions WHERE acct_num = '" + Id + "')";

                System.out.println( "Sql query being sent to the db is: " + sqlQuery );

        ResultSet results = dbSvr.runDBReadQuery( sqlQuery );

                if ( results == null )
                {
                    InfoData[0] = "";
                }else
                {
                    try
                    {
                        results.next();
                        InfoData[0] = results.getString( "acct_num" );
                        System.out.println( "results: " + InfoData[0]);

                        InfoData[1] = results.getString( "trans_type_cd" );
                        System.out.println( "results: " + InfoData[1]);

                        InfoData[2] = results.getString( "trans_amt" );
                        System.out.println( "results: " + InfoData[2]);

                        InfoData[3] = results.getString( "create_dt" );
                        System.out.println( "results: " + InfoData[3]);

                    }
                    catch(SQLException e)
                    {
                        System.out.println("Read transaction failed: " + e );
                    }
                }

        return InfoData;
    }

    // Used to insert a transaction record for an account
    public synchronized java.lang.String insertTransaction(
    )
    throws org.omg.CORBA.SystemException,
     OBBankObjects.OBBankObjects.Unknown
     {
                String sMessage = "";

                String sqlQuery = "INSERT INTO transactions(trans_id, acct_num, trans_type_cd, trans_amt, create_dt)
VALUES (trans_id_sequence.NEXTVAL, '" + Data[0] + "', '" + Data[1] + "', '" + Data[2] + "', '" + Data[3] + "')";
```

```
                    System.out.println( "Sql query being sent to the db is: " + sqlQuery );

                    int rows = dbSvr.runDBInOrUpQuery( sqlQuery );

                    sMessage = "Number of rows affected: " + String.valueOf( rows );

                    return sMessage;
  }

  // Used to update a transaction record for an account
  public synchronized java.lang.String updateTransaction(
  )
  throws org.omg.CORBA.SystemException,
   OBBankObjects.OBBankObjects.Unknown
   {
                    String sMessage = "";

                    String sqlQuery = "UPDATE transactions SET trans_type_cd = '" + Data[1] + "', trans_amt = '" + Data[2] + "',
create_dt = '" + Data[3] + "' WHERE acct_num = '" + Data[0] + "' AND trans_id = (SELECT MAX(trans_id) FROM transactions
WHERE acct_num = '" + Data[0] + "')";

                    System.out.println( "Sql query being sent to the db is: " + sqlQuery );

                    int rows = dbSvr.runDBInOrUpQuery( sqlQuery );

                    sMessage = "Number of rows affected: " + String.valueOf( rows );

                    return sMessage;
  }

  public org.omg.PortableServer.POA _default_POA()
  {
    return m_poa;
  }

}


// End of OBTransactionImpl
```

```
// Michelle McKeller - Graduate Project
// Corba - Orbix 2000
// OBBank.idl

module OBBankObjects {
  exception Unknown{};

        typedef string OBAcct[25];
        typedef string OBTrans[4];

        interface OBTransaction {
                // Reads the transaction data for a given account number
                OBTrans readTransaction() raises(Unknown);

                // Inserts the transaction data into the database
    // for a given account number
                string insertTransaction() raises(Unknown);

                // Updates the transaction data in the database
    // for a given account number and trans id
                string updateTransaction() raises(Unknown);
        };

        interface OBAccount {

                //Gets Last made transaction for an Account
                OBTransaction getTransaction(in string sId) raises(Unknown);

                //Puts Last made transaction for an Account
                OBTransaction putTransaction(in string sId, in OBTrans objItem) raises(Unknown);

                // Reads the account data for a given account number
                OBAcct readAccount() raises(Unknown);

                // Inserts the account data into the database
    // for a given account number
                string insertAccount() raises(Unknown);

                // Updates the account data in the database
    // for a given account number
                string updateAccount() raises(Unknown);
        };

        interface OBManager {
                // Gets an account object
                OBAccount getAccount(in string Id) raises(Unknown);

                // Puts an account object
                OBAccount putAccount(in string Id, in OBAcct objItem) raises(Unknown);
        };


};

// End of OBBank.idl
```

## Build.xml document:

```
<project name="generated_app" default="build_all" basedir=".">

 <property name="idl_flags" value="-I/opt/iona/orbix_art/1.2/idl -jbase=-POBBankObjects:-Ojava_output -jpoa=-POBBankObjects:-
Ojava_output "/>
 <property name="clpath"
value="/usr/local/jdk1.3/lib/classes111.zip:$ORACLE_HOME/jdbc/lib/classes111.zip:/etc/opt/iona/domains:/opt/iona/orbix_art/1.2/cl
asses/orbix2000.jar:/opt/iona/orbix_art/1.2/classes/omg.jar:/etc/opt/iona:/etc/opt/iona/domains:/opt/iona/orbix_art/1.2/classes/orbix200
0.jar:/opt/iona/orbix_art/1.2/classes/omg.jar:/etc/opt/iona:/etc/opt/iona:/etc/opt/iona/domains:export:/opt/iona/orbix_art/1.2/demos/clas
ses:/etc/opt/iona:/etc/opt/iona/domains"/>
 <target name="init">
  <tstamp/>
  <property name="classes" value="classes"/>
  <mkdir dir="${classes}"/>
 </target>

 <target name="idl_compile" depends="init">
  <exec dir="./" command="/opt/iona/orbix_art/1.2/bin/idl ${idl_flags} OBBank.idl" output="idl_compiler.out"/>
  <exec dir="./" command="cat idl_compiler.out"/>
 </target>

 <target name="build_all" depends="idl_compile">
 <javac classpath="./classes:${clpath}"
       srcdir="./" destdir="./classes"/>
 </target>

 <target name="runserver" depends="">
  <java classpath="./classes:${clpath}"
            jvmargs="-Dorg.omg.CORBA.ORBClass=com.iona.corba.art.artimpl.ORBImpl -
Dorg.omg.CORBA.ORBSingletonClass=com.iona.corba.art.artimpl.ORBSingleton"
       args="-ORBdomain_name default-domain
http://neptune.cocse.unf.edu:2400/mmckeller/Orbix/OBBankObjects/OBBankObjects/server oracle.jdbc.driver.OracleDriver
jdbc:oracle:thin:@manatee.unf.edu:1521:sid1 mmckel unfunf2001"
            fork="yes"
            classname="OBBankObjects.server"/>
 </target>

 <target name="runclient" depends="">
  <java classpath="./classes:${clpath}"
            jvmargs="-Dorg.omg.CORBA.ORBClass=com.iona.corba.art.artimpl.ORBImpl -
Dorg.omg.CORBA.ORBSingletonClass=com.iona.corba.art.artimpl.ORBSingleton"
       args="-ORBdomain_name default-domain UPDATETRANS 200 UpdateTrans.txt
http://neptune.cocse.unf.edu:2400/mmckeller/Orbix/OBBankObjects/OBBankObjects/server"
            fork="yes"
            classname="OBBankObjects.client"/>
 </target>

 <target name="info" depends="">
  <echo message=" help:" />
  <echo message="build.xml options:"/>
  <echo message=""/>
  <echo message="info      Prints out this message." />
  <echo message="build_all  Deletes class files, IDL compiler generated files"/>
  <echo message="and rebuilds everything."/>
  <echo message="clean      Removes all class files."/>
  <echo message="clean_all  Removes all generated files."/>
  <echo message="runserver  Run the server.   "/>
  <echo message="runclient  Run the client."/>
 </target>



 <target name="clean" depends="">
  <deltree dir="classes"/>
  <delete dir="./" includes="idl_compiler.out,ant_env.csh ant_env.sh,*.ref"/>
 </target>
 <target name="clean_all" depends="">
  <deltree dir="OBBankObjects"/>
```

```
 <deltree dir="classes"/>
 <deltree dir="java_output"/>
 <delete dir="./" includes="idl_compiler.out,ant_env.csh ant_env.sh,*.ref"/>
 <deltree dir="idlgen"/>
 <delete dir="./" includes="./*.ref"/>
 <delete dir="./" includes="build.xml"/>
 <delete dir="./" includes="idl_compiler.out"/>
</target>

</project>
```

# APPENDIX O

## Directory Structure for Project CD

Under the folder GPMcKeller on the CD are all of the folders you will need to run the three applications. When trying to run the applications, you need to keep the same directory structure that exists under the folders. Also, the scripts are set to test with 25 clients, if you wish to change this number you must edit the script. If you need to run the applications on different machines other than neptune for the server, dsp for the client, and manatee for the database, then you will have to change the scripts to the new machine names.

1. BashProfile for Server – the .bash_profile file you need to have setup as your .bash_profile on your server

2. Corba – all of the files to run the Java 2 ORB client-server application.
   To test each one of the test files, a separate script has been established that starts the client, they are:
   clcbinsacct – tests inserting an account
   clcbinstrans - tests inserting a transaction for an account
   clcbreadacct – tests reading an account
   clcbreadtrans – tests reading the last transaction for an account
   clcbupdacct – tests updating an account
   clcbupdtrans – tests updating the last transaction for an account
   To start the server: svrcb
   To start the naming service: svrcbnamesvr
   In order to test a different number of clients, the particular clcb* file you are running has to be modified with the number you want to run.

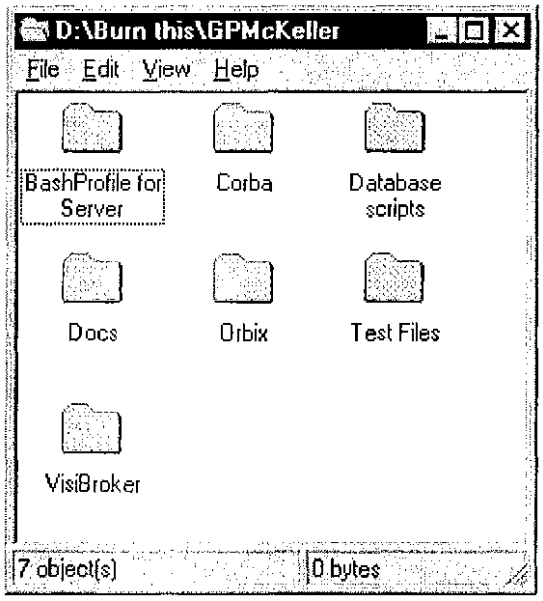
3. Database scripts – All of the scripts you need to create the database that is used by the applications. AD.sql is the script to run, it will run the rest of the scripts. To drop the database, run the droptables.sql file.

4. Docs – the graduate project paper.

5. Orbix – all of the files to run the Orbix 2000 client-server application. It is extremely important that this directory structure is kept and the directory names stay the same, or the application will not run. Orbix uses .xml files to run their applications instead of standard scripts. A separate .xml document has been provided to test each one of the test files, they are in the script folder under the Orbix folder, in order to run them with the ant shell that Orbix provides, you must rename the one you want to use to build.xml. The .xml documents are:
   InsertAcct.xml – tests inserting an account

InsertTrans.xml – tests inserting a transaction for an account
ReadAcct.xml – tests reading an account
ReadTrans.xml – tests reading the last transaction for an account
UpdateAcct.xml – tests updating an account
UpdateTrans.xml – tests updating the last transaction for an account
The build.xml file is also used with ant to start the server and client.
The Orbix services are automatically running as background daemons under userid mmckeller.

6. Test Files – includes the original Excel spreadsheets used to create the flat text files with that were used for testing, and the actual text test files. In order to create a new test file out of a spreadsheet, when saving the spreadsheet as a text file, the file type must be Text (Tab delimited) (*.txt) or the client cannot process the file correctly due to tokenizing the record based on tabs.
7. VisiBroker – all of the files needed to run the VisiBroker client-server application. To test each one of the test files, a separate script has been established that starts the client, they are:
   clvbinsacct – tests inserting an account
   clvbinstrans - tests inserting a transaction for an account
   clvbreadacct – tests reading an account
   clvbreadtrans – tests reading the last transaction for an account
   clvbupdacct – tests updating an account
   clvbupdtrans – tests updating the last transaction for an account
   To start the server: svrvb
   To start the naming service: svrvbnamesvr
   In order to test a different number of clients, the particular clvb* file you are running has to be modified with the number you want to run.




Directory structure for project CD

# VITA

Michelle Leigh McKeller has a Bachelor of Science degree in Computer Information Systems, Fall 1996 from the University of North Florida and will be graduating from the University of North Florida with a Master of Science degree in Computer and Information Sciences, Fall 2001. Dr. Sanjay P. Ahuja of the University of North Florida is serving as Michelle's graduate project adviser. Michelle is currently employed as a senior systems programmer analyst at Homeside Lending, Inc and has been with the company since receiving her Bachelor's degree in 1996.

Michelle has worked extensively with VB, C++, and SQL in her current employed position and has hopes of broadening her career toward more development using Java and Web development. Michelle's academic work has included use of Java, C, COBOL, Oracle, TCL, PHP, Perl, Python, and various other languages. Michelle enjoys spending time with her long-time boyfriend, friends, and family.