This lab will be used to bring together many of the concepts the lecturer has described. Figure 1 below shows three applications (these could be executing on different machines but, for this exercise they will be running on a single machine).

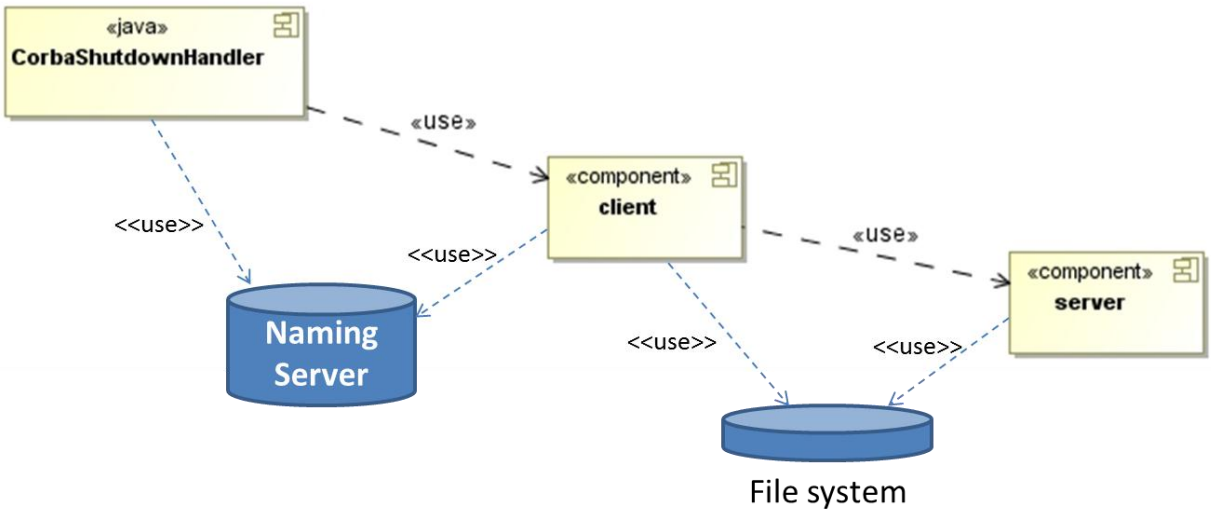


Figure 1.

As with any multi-tiered environment, there are components that represent business applications and there are those that represent naming and directory services.

Service type	Component
Business Application	**CorbaShutdownHandler** {UI} **Client** {DOS application, no UI} **Server** {DOS application, writes to dos console}
Naming Service	TAO **Naming Server** DOS **file system**

Bootstrapping

There is always a bootstrapping order that must be carefully followed if an architecture of this kind is to run successfully. Given that the all three business applications require the use of some kind of Naming Service, it's only reasonable to assume that TAO Naming Server and the DOS file system must be running before the business applications can be started. Obviously the file system will already be on your computer and running but, the TAO Naming Server will have to be started manually

- Run the batch the file start_tao_ns.bat in the directory C:\tao-2.1

If you successfully run this application, a new dos console window will start with the title c:\tao-2.1\bin_ms\tao_cosnaming.exe. This is a Naming Service application. This is just one of many types of Naming Service applications available in the market.

Starting the Server

As in figure 1, we need to start the server. This is binary executable, written in C++ and built using TAO CORBA C++ libraries. Code has been generated for Windows 32-bit platform.

- Run the batch file run_server.bat in the directory apps\bin

If you successfully run this application, a new dos console window will be created. It's contents will be a number of dos commands, followed by some output from the application that ends with the lines

Binding helloServer in the Naming Service ...
done
Binding helloServer_2 in the Naming Service ...
done

Starting the Client

As in figure 1, we need to start the client. This is binary executable, written in C++ and built using TAO CORBA C++ libraries. Code has been generated for Windows 32-bit platform.

- Run the batch file run_client.bat in the directory apps\bin

If you successfully run this application, a new dos console window will be created. The server console window will begin to show a rapid succession of output "[XXX] HelloServant::Hello World, where XXX is an incrementing number.

Stopping the counting

The output being produced in the server dos console can be stopped in one of two ways

1. Killing the client application
2. Sending a message to the client application to tell it stop running

Killing the client application

Let's try killing the client application. Simply press ctrl-C on the dos console, terminate the batch file when requested. This will close the dos console and the output from the server console will stop incrementing.

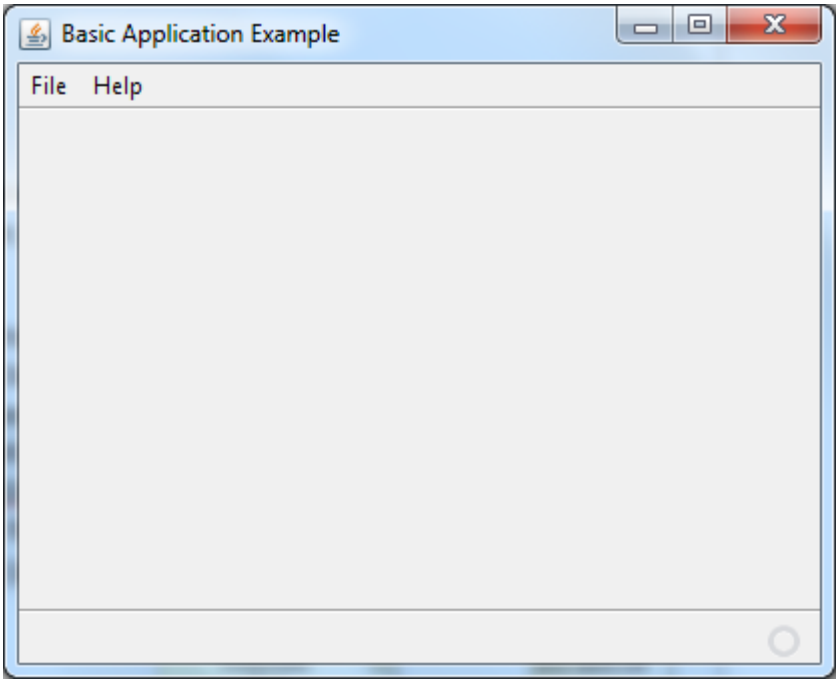
The client application can be restarted by running the batch file run_client.bat.

Sending a message to the client

An application exists that can be used to send a message to the client dos console. The application is called CorbaShutdownHandler. This is a java binary executable, built using JacORB java libraries.

- Run the jar file CorbaShutdownHandler.jar using the command: java –jar CorbaShutdownHandler.jar. It can be found in apps\bin.

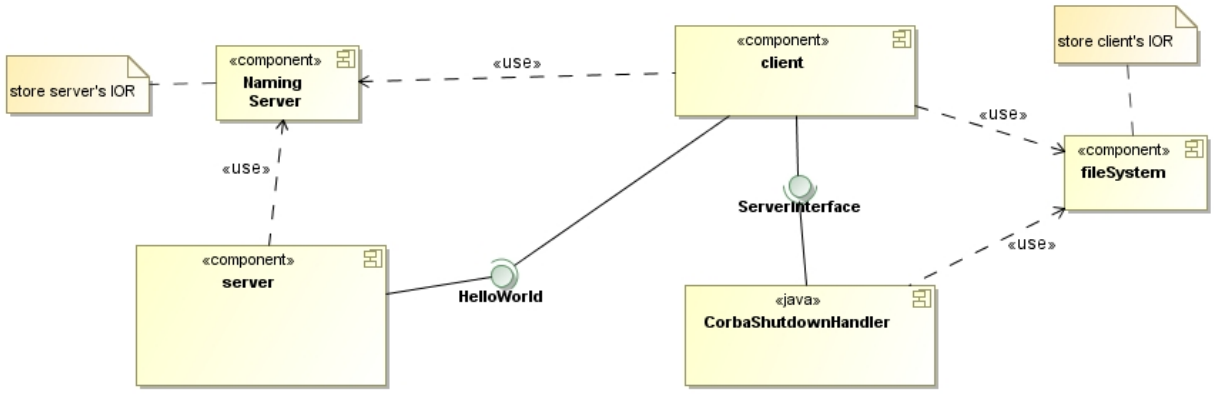
If successfully run you should see the following GUI application window



Use the File… commands to stop the server. Selecting “Kill Server” will send a IIOP message to the client dos console, this application in turn stop sending IIOP messages to the server dos console and immediately terminate The “Exit” command may then be used. Selecting the “Exit” command before the “Kill Server” command, will simply exit this application without sending a STOP command to the client dos console.

Software Architecture

Figure 1 showed the application. Figure 2 outlines the software architecture. It's main focus is the interfaces that each software component consumes or produces.




```
interface HelloWorld
{
	void hello();
};

//	Put this in a module because Java doesn't like the use of the
//	default namespace for packages

module		SimpleServer
{
	interface ServerInterface
	{
		void shutdown();
	};
};
```