# Santa Clara University

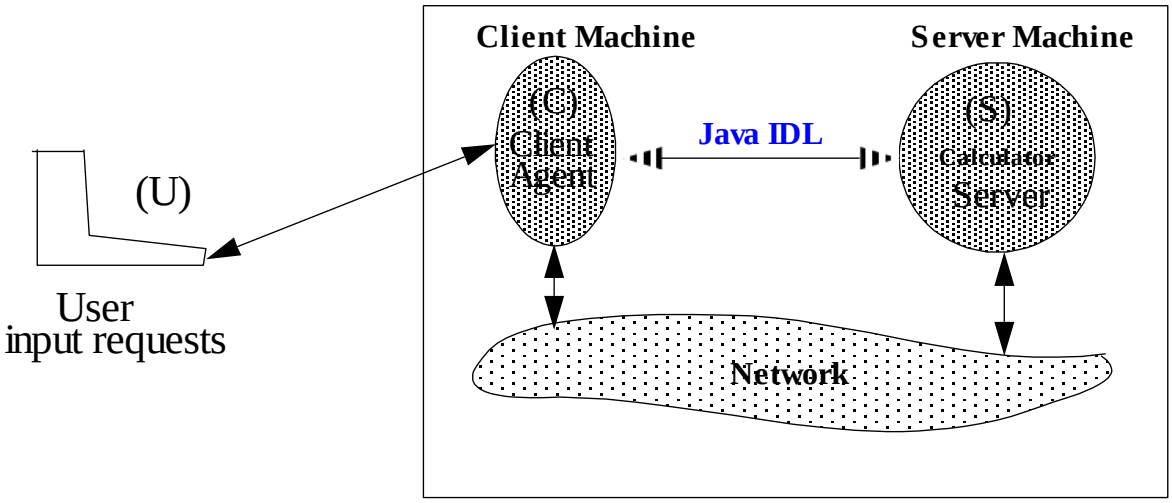
Department of Computer Engineering

Client/Server Programming (COEN 235)

# Project-2

## Project Overview:

The goal of this project is to implement a Client/Server Java objects running on different machines. The Client object invoke methods on the Server objects through Java IDL capability.




## Overview:

In this project, we would like to build a simple "**Calculator" server (S)** and a **Client** agent (**C**) Java objects. The client agent object reads input command/arithmetic operation from the user (**U**), invokes the **calculate**() method on the server IDL interface, gets the result back and pass it back to the User (**U**). The interactions between the **User**, **Client Agent**, and the "**Calculator" Server** are synchronous, i.e., the user has to wait to receive the response to the calculation operation request before responding by another request, etc.

## Functional Requirements:

1. The client agent (**C**) process reads a question string from the user "from the input device (terminal)" as a string.
2. The client invokes the calculate() method on the calculator server object with parameters corresponding to the user operation and waits for the response.
3. The client gets the result back from the server and writes it to the output device.
4. The client agent needs to recognize an Exit command when user is done. (**C**) needs to invoke the exit() method on the server object which results in cleaning up the server (**S**) object/process.

## Programming Model:

The calculator server implements the following IDL interface is:

```
module final {
        interface Calc{
                long calculate (in long opcode, in long op1, in long op2);
                long exit();
        }
}
```

## Interaction Example:

**Client**: + 2 4

**Server**: 6

**Client**: - 6 4

**Server**: 2

.....

**Client**: Exit

**Server**: "I am out"

## **Programming Hints:**

1. Start the Server process first. The server object needs to register itself with its local ORB, acquire reference to CORBA Naming Service and register itself with the Naming Service.
2. Start the client process. The client agent needs to lookup for the "calculator" server object reference before invoking methods on the server object.

3. Include one/two pages description of your project implementation. Remember to have good comments in your code. Also, include a hard copy of user interaction with your "**Calculator**" server application. Finally, write clearly on the Front page the **absolute UNIX pathname** for your application to the grader to try (make sure RWX permissions for others).

4. Bring to class two complete copy sets of your project (one for the grader and the second for myself).