Assignment No 2

Title: Develop Any Distributed Application using CORBA to Demonstrate Object Brokering (Calculator or String Operations).

Objectives

- To understand the principles of CORBA (Common Object Request Broker Architecture) in distributed systems.
- To learn how CORBA enables communication between objects across different platforms and programming languages.
- To design and implement a CORBA-based distributed application using object brokering.
- To demonstrate real-time remote object invocation through a calculator or string operation service.

Problem Statement

Design and develop a distributed application using CORBA where the client can perform operations (either Calculator operations like addition, subtraction, etc., or String operations like concatenation, reverse, etc.) by invoking methods on remote objects. The server should expose these operations through an interface defined using IDL (Interface Definition Language). The system must demonstrate object brokering where the Object Request Broker (ORB) facilitates communication between clients and server objects running on different machines or platforms.

Expected Outcomes

- A functioning CORBA-based distributed system that performs either arithmetic or string operations.
- Practical understanding of:
  - Writing IDL files.
  - Compiling IDL to generate stubs and skeletons.
  - Implementing server and client applications using CORBA.
- Understanding of object brokering and distributed object communication.
- Experience with platform-independent distributed programming.

Software Requirements

- **Operating System**: Windows / Linux / macOS
- **Programming Languages**: Java or C++ (Java preferred for educational CORBA tools)
- **CORBA Implementation**:
  - Java: `JavaIDL` or `ORBacus`
  - C++: `TAO` or `OmniORB`
- **IDL Compiler**: Comes with the CORBA implementation (e.g., `idlj` for Java)
- **IDE**: Eclipse / NetBeans / Command-line tools

Hardware Requirements

- **Processor**: Minimum Dual-core CPU
- **RAM**: 4 GB or more
- **Storage**: At least 100 MB of disk space
- **Network**: Localhost or LAN for client-server communication testing

Theory:

CORBA (Common Object Request Broker Architecture) is a standard defined by the OMG (Object Management Group) that allows objects written in different programming languages and running on different platforms to work together in a distributed environment.

At the heart of CORBA is the **Object Request Broker (ORB)**, which is responsible for:

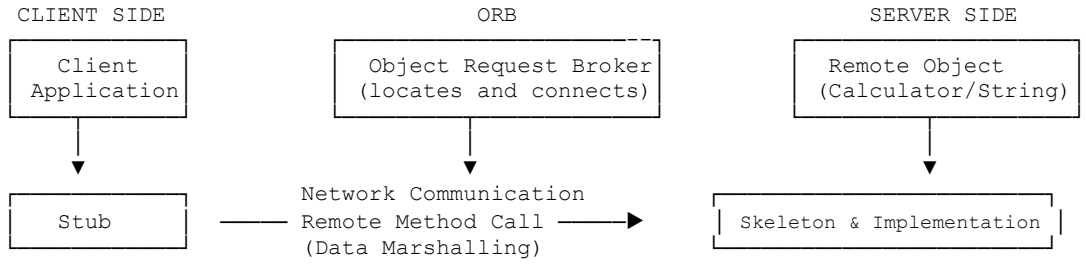
- Locating the object implementation.
- Communicating requests and responses between clients and servers.
- Converting data (marshalling/unmarshalling) to and from a transportable format.

Key Components of CORBA:

Component	Description
IDL (Interface Definition Language)	Used to define the interfaces that remote objects expose.
ORB (Object Request Broker)	Middleware that handles method invocation between clients and remote objects.
Stub (Client Proxy)	Acts as a local representative of the remote object on the client side.
Skeleton (Server Proxy)	Receives method calls on the server and forwards them to the actual object.
Naming Service	Allows clients to look up server objects by name.

CORBA Architecture Diagram

Here's a simplified diagram showing how CORBA components interact:



Flow of Execution:

1. The client invokes a method on the **stub**.
2. The **ORB** marshals the request and sends it over the network.
3. On the server side, the ORB uses the **skeleton** to invoke the actual method.
4. The result is sent back to the client through the same path.

How CORBA Works – Step-by-Step

1. **IDL Definition**:
   The remote interface is defined in IDL, describing all the methods that a client can call.
2. **Stub & Skeleton Generation**:
   The IDL file is compiled using an **IDL compiler** to generate:
   - Client-side **stub** (acts like a proxy).
   - Server-side **skeleton** (receives and delegates method calls).
3. **Implementation**:
   The developer writes:
   - The **server implementation** of the interface (servant).
   - The **client application** that uses the stub to call remote methods.
4. **Object Registration**:
   The server registers its object reference with the **Naming Service**.
5. **Client Lookup**:
   The client contacts the Naming Service to retrieve the reference to the server object.
6. **Method Invocation**:
   The client invokes a method on the stub. The ORB takes over:
   - Marshals method name and parameters.
   - Sends the request over the network.
   - The server ORB receives the request, unmarshals it, and invokes the method on the servant through the skeleton.
   - The return value is sent back through the same channel.

Why CORBA?

- Platform and language independence.
- Encourages component reuse and scalability in enterprise applications.
- Transparent communication between distributed components.
- Defines standard interfaces and protocols for distributed computing.

Conclusion: