Advanced Programming

Week 11

Distributed Programming

- CORBA
- CORBA architecture
- CORBA IDL
- Naming services

Tilahun Melak(PhD)

November, 2025

Objectives

At the end of this lecture, students will be able to:

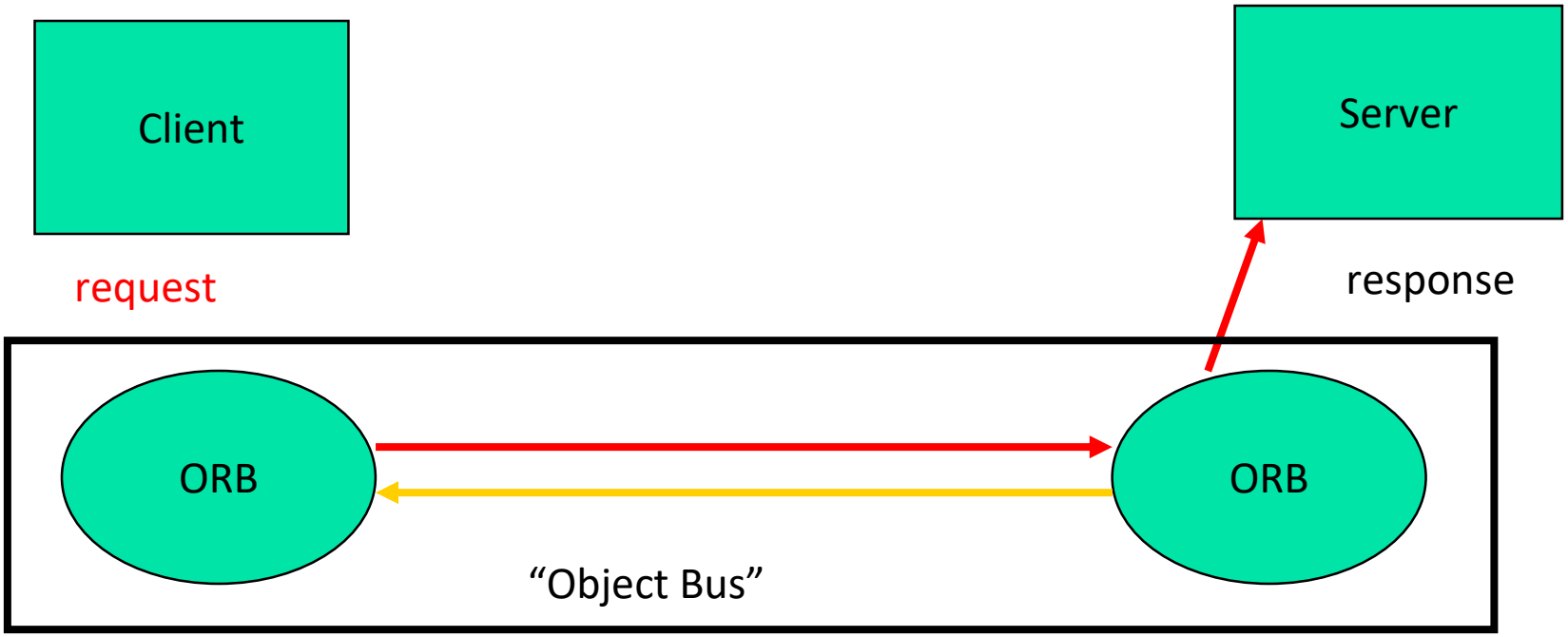
- Explain the importance and purpose of CORBA.
- Describe the architecture and key components of CORBA, including the Object Request Broker (ORB), Interface Definition Language (IDL), and communication mechanisms in distributed object systems.
- Explain the role of IDL compilers in generating language mappings and integrating interface definitions with object implementations.
- Describe how the CORBA Naming Service enables clients to locate objects and servers to register object references using names.

What is CORBA good for?

- Developing distributed applications
- Locating remote objects on a network
- Sending messages to those objects
- Common interface for transactions, security, etc.

Object Management Group (OMG). (2012). *Common Object Request Broker Architecture (CORBA) Specification, Version 3.3.*

Basic CORBA Architecture



Object Management Group (OMG). (2012). Common Object Request Broker Architecture (CORBA) Specification, Version 3.3.

Object Request Broker (ORB)

- Gives the communication infrastructure that is capable of relaying object requests across distributed platforms.
- Client calls the Object implementation through interfaces provided by ORB.

Advantages:

- Separates Client and Server implementation
- Separates both client and Server from underlying communication infrastructure and protocol stack and so replaceable while migration from one implementation to other

Object Management Group (OMG). (2012). Common Object Request Broker Architecture (CORBA) Specification, Version 3.3.

Object Request Broker (ORB)…

IIOP

- Internet Inter-Orb Protocol
- Network or "wire" protocol
- Works across TCP/IP (the Internet protocol)

Object Management Group (OMG). (2012). Common Object Request Broker Architecture (CORBA) Specification, Version 3.3.

Creating CORBA IDL Files (Contents)

- The CORBA Interface Definition Language (IDL)
- Declaring data members, methods, and parameters
- The interface compiler
- Separating client and server on different machines

Object Management Group (OMG). (2012). Common Object Request Broker Architecture (CORBA) Specification, Version 3.3.

CORBA IDL

- IDL is used to describe the interfaces that client objects call and that object implementations provide.
- It is a specification that enables interoperability by separating interface from implementation.
- It is not a programming language – has no constructs
- It maps to many programming languages like C, C++, Java, and COBOL via OMG standards

Object Management Group. (2012). CORBA specification: Common Object Request Broker Architecture, Version 3.3

The Interface Definition

- The interface is the syntax part of the contract that the server object offers to the clients that invoke it
- Clients access objects only through their advertised interface, invoking only those operations that the object exposes through its IDL interface, with only those parameters (input and output) that are included in the invocation
- The IDL interface definition is independent of programming language.
- It is mapped to programming languages using the interface compiler.

Schildt, H. (2018). *Java: The complete reference* (11th ed.). McGraw-Hill Education.

The Interface Definition…

Example:

```
 module Calc
{ interface Calculator
  {
   float  calculate(in float val1, in float val2, in
  char operator);
  };
};
```

- This is an interface for a **Calculator** with one method – **calculate**.
- A module can have many interfaces.

Schildt, H. (2018). *Java: The complete reference* (11th ed.). McGraw-Hill Education.

Mapping IDL to Java, C++

IDL	Java	C++
module	package	namespace
interface	interface	abstract class
operation	method	member function
attribute	pair of methods	pair of functions
exception	exception	exception

Oracle. (n.d.). Getting started with RMI. Oracle Java Documentation.

Declaring Data Members

- Data members are declared using the **attribute** keyword.
- The declaration must include a name and a type.
- Attributes are readable and writable by default. To make a readonly attribute, use the **readonly** keyword.
- IDL compiler generates public read and write methods for the data member as required.

Object Management Group. (2023). CORBA Interface Definition Language (IDL) specification.

Declaring Data Members

- For example,

  attribute long assignable ;

  generates,

  int assignable();

  void assignable (int i);

Object Management Group. (2023). CORBA Interface Definition Language (IDL) specification.

Declaring Data Members…

- CORBA **const** values declared in an IDL map to **public static final** fields in the corresponding Java interface.
- Constants not declared inside the interface are mapped to **public interface** with the same name containing a field **value**.

  const float sample = 2.3;
- It is also possible to define your own types using the typedef keyword.

  typedef string name;

Object Management Group. (2023). CORBA Interface Definition Language (IDL) specification.

Declaring Methods

- Methods are declared by specifying name, return type, and parameters.

```
float calculate(in float val1, in float val2, in char operator);
```

- Methods can optionally throw exceptions. User-defined exceptions must be declared in the IDL.

Object Management Group. (2023). CORBA Interface Definition Language (IDL) specification.

Methods Types

- Methods are *synchronous* by default.
- The client program will wait for the remote method to execute and return.
- *Asynchronous* methods are defined using the **oneway** keyword.
  - **oneway** methods have no return value, can have input parameters only and cannot throw exceptions.

Object Management Group (OMG). (2012). Common Object Request Broker Architecture (CORBA) specification, version 3.3.

Methods Types…

- The client makes the call to the **oneway** method and continues processing while the remote object executes it – the client is not blocked.

Object Management Group (OMG). (2012). Common Object Request Broker Architecture (CORBA) specification, version 3.3.

Declaring Methods

```
module Calc{
  interface Calculator{
  //User-defined exception
     exception MyException{};
  //synchronous method
  float calculate(in float val1, in float val2, in
  char operator) raises (MyException);
  //asynchronous method
  oneway void setvalue(in long val);
  };
};
```

Exceptions

- There are two types of CORBA exceptions:
  - *System Exceptions* are thrown when something goes wrong with the system
  - *User Exceptions* are generated if something goes wrong inside the execution of the remote method

Object Management Group (OMG). (2012). Common Object Request Broker Architecture (CORBA) specification, version 3.3.

Exceptions…

- They are declared inside the IDL definition for the object, and are automatically generated by the IDL compiler
- Implementation of these exceptions depends on the language mapping

Object Management Group (OMG). (2012). Common Object Request Broker Architecture (CORBA) specification, version 3.3.

Declaring Parameters

- Parameters can be of following types:
  - Basic (**char**, **long**, **short**, **float**, **bool**, etc.),
  - Constructed (**struct**, **array**, **sequence**),
  - Typed objects, or **any**.
- information and are copied both ways.

Object Management Group. (2023). CORBA Interface Definition Language (IDL) specification.

Declaring Parameters…

- Parameters can be declared as **in**, **out**, or **inout**.
  - **in** parameters are copied from client to server
  - **out** parameters are copied from server to client
  - **inout** parameters are used both for incoming and outgoing information and are copied both ways.

Object Management Group. (2023). CORBA Interface Definition Language (IDL) specification.

Interface Compilers

- IDL Compilers implement language mappings in software.
- They compile the interface definition (defined using IDL) to produce output that can be compiled and linked with an object implementation and its clients.
- Every ORB comes with one or more IDL compilers, one for each language that it supports.

Object Management Group (OMG). (2012). Common Object Request Broker Architecture (CORBA) specification, version 3.3.

Interface Compilers…

- Many vendors supply their IDL compilers. Because the compilers implement mapping standards, every vendor's IDL compiler produces language code with the same mappings, making the code vendor independent.
- Sun's JDK provides the **`idlj`** compiler.

  VisiBroker provides **`idl2cpp`** and **`idl2java`** compilers.

Object Management Group (OMG). (2012). Common Object Request Broker Architecture (CORBA) specification, version 3.3.

Need for Interface Compiler

- The interface compiler converts the language independent IDL to language specific code that C++, Java or other language clients and servers can understand.

  For example, the **`idlj`** compiler compiles the IDL to Java code.
- IDL allows an object implementer to choose the appropriate programming language for the object.
- Client and Server can be developed in parallel.
- Clients depend only on the interface and not the implementation of the server code.

Object Management Group (OMG). (2012). Common Object Request Broker Architecture (CORBA) specification, version 3.3.

Need for Interface Compiler…

- By using IDL and interface compiler, the following can be defined independent of the programming language
  - Modularized object interfaces
  - Operations and attributes that an object supports
  - Exceptions raised by an operation
  - Data types of an operation return value, its parameters, and an object's attributes

Object Management Group (OMG). (2012). Common Object Request Broker Architecture (CORBA) specification, version 3.3.

The idlj compiler

- To compile the IDL to java using **`idlj`**,

  `idlj -fall <idl_file_name>`
- The **`-fall`** option creates both server and client files
- **`-fclient`** generates client files only (default)
- **`-fserver`** generates server files only

Object Management Group (OMG). (2012). Common Object Request Broker Architecture (CORBA) specification, version 3.3.

Files created by idlj

The following classes are created when Calc.idl is compiled using **`idlj`**

- **`Calc.Calculator`** - The IDL interface represented as a Java interface
- **`Calc.CalculatorHelper`** - Implements the type operations for the interface
- **`Calc.CalculatorHolder`** - Used for **`out`** and **`inout`** parameters
- **Calc.CalculatorOperations** – The interface that defines the exposed remote methods

Object Management Group (OMG). (2012). Common Object Request Broker Architecture (CORBA) specification, version 3.3.

Files created by idlj…

Calc.CalculatorStub - The client stub. Implements a local object representing the remote CORBA object

- This object forwards all requests to the remote object. The client does not use this class directly
- **Calc.CalculatorImplBase** - An abstract class that implements the Calculator interface.
- It is the server skeleton

Object Management Group (OMG). (2012). Common Object Request Broker Architecture (CORBA) specification, version 3.3.

Separating Client and Server

Client

- _CalculatorStub
- Calculator
- CalculatorHelper
- CalculatorHolder
- CalculatorOperations

Server

- _CalculatorImplBase
- Calculator
- CalculatorOperations

Object Management Group (OMG). (2012). Common Object Request Broker Architecture (CORBA) specification, version 3.3.

What next?

- Implement the client
- Compile the client
- Implement the server
- Compile the server
- Start ORB
- Start server
- Start client

Object Management Group (OMG). (2012). Common Object Request Broker Architecture (CORBA) specification, version 3.3.

Naming & Directory Services

- A naming service maintains a set of *bindings* that relate names to objects
- All objects in a naming system are named in the same way (that is, they subscribe to the same *naming convention*)
- Clients use the naming service to locate objects by name

Object Management Group (OMG). (2012). *Common Object Request Broker Architecture (CORBA) specification, version 3.3*.

Naming & Directory Services…

- Making a request to a **service** or accessing an **object** by means of inter-process communication requires that one must first locate the **service** or **object**.
- Service are abstractions of objects.
- They are usually represented by processes with a service access point.
- Object may be users, computers, communication links or other resources such as files.

Object Management Group (OMG). (2012). *Common Object Request Broker Architecture (CORBA) specification, version 3.3*.

Naming & Directory Services…

- Services and Objects are normally identified by textual names.
- Alternatively, if names are unknown, service or object entities can be described by using attributes associated with them.
- Although services and objects have distinct meanings, their naming issues are similar
- Name and Directory services, in a narrow sense are look-up operations.
- The terms name service and directory service are often used interchangeably.

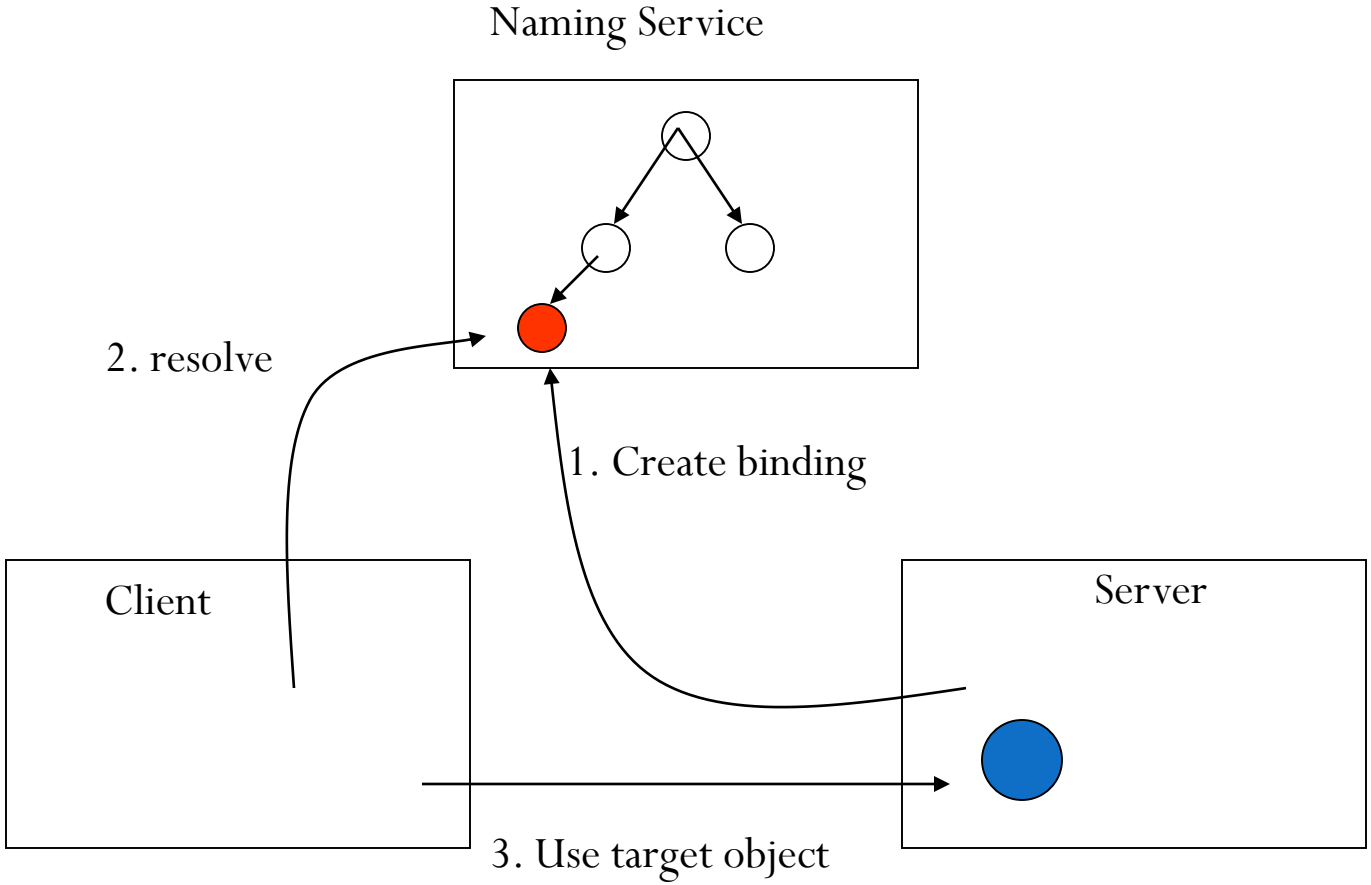
Object Management Group (OMG). (2012). *Common Object Request Broker Architecture (CORBA) specification, version 3.3*.

Why do we need Naming & Directory Services

- Object-oriented middleware uses object references to address server objects
- We need to find a way to get hold of these object references without assuming physical locations
- A name is a sequence of character strings that can be bound to an object reference
- A name binding can be resolved to obtain the object reference
- There may be many server objects in a distributed object system
- Server objects may have several names

Object Management Group (OMG). (2012). *Common Object Request Broker Architecture (CORBA) specification, version 3.3*.

Naming Service



Object Management Group (OMG). (2012). *Common Object Request Broker Architecture (CORBA) specification, version 3.3*.

COS (Common Object Services) Naming

- The naming service for CORBA applications; allows applications to store and access references to CORBA objects.
- The CORBA naming service is used by
  - Clients to locate CORBA objects
  - Servers to advertise specific CORBA objects
- Server creates associations between names and object references for the CORBA objects (*object binding)*
- Client retrieve the object references by querying the naming service and using the name as the key.

Object Management Group (OMG). (2012). Common Object Request Broker Architecture (CORBA) specification, version 3.3.

Summary

- In today's lecture we have discussed about;
  - Importance of CORBA
  - CORBA architecture
  - CORBA components—such as the Object Request Broker, Interface Definition Language, and communication mechanisms.
  - IDL compilers create language mappings that link CORBA interface definitions with specific programming languages for interoperability.
  - The CORBA Naming Service enables servers to register objects by name and allows clients to locate and access them in distributed systems.

References

- Object Management Group (OMG). (2012). Common Object Request Broker Architecture (CORBA) Specification, Version 3.3.
- Object Management Group (OMG). (2023). CORBA and IDL specification.
- Schildt, H. (2018). Java: The complete reference (11th ed.). McGraw-Hill Education.
- Object Management Group. (2012). *CORBA specification: Common Object Request Broker Architecture, Version 3.3*
- Object Management Group. (2023). CORBA Interface Definition Language (IDL) specification.