Práctica 1: PROCESO BÁSICO DE DESARROLLO CON CORBA

Práctica 1. (Será realizada en la Sala de Computo)

El objetivo de esta práctica es crear una completa aplicación basada en CORBA utilizando el lenguaje de definición de interfaces (IDL) y el compilador de java JIDL para generar los stubs y skeleton. En este ejercicio se implementará al servant por herencia y se utilizaran parámetros out en las operaciones. En esta práctica se requiere implementar un servicio que permita registrar un paciente y consultar un paciente mediante el siguiente menú:

=== Menu ===

1- Registrar paciente
2- Consultar paciente
3. Salir

Restricciones

En la implementación del servidor se deberá validar que no existan dos pacientes con el mismo número de habitación, y máximo se podrán registrar 5 pacientes. La estructura de datos utilizada para almacenar los pacientes será un vector.

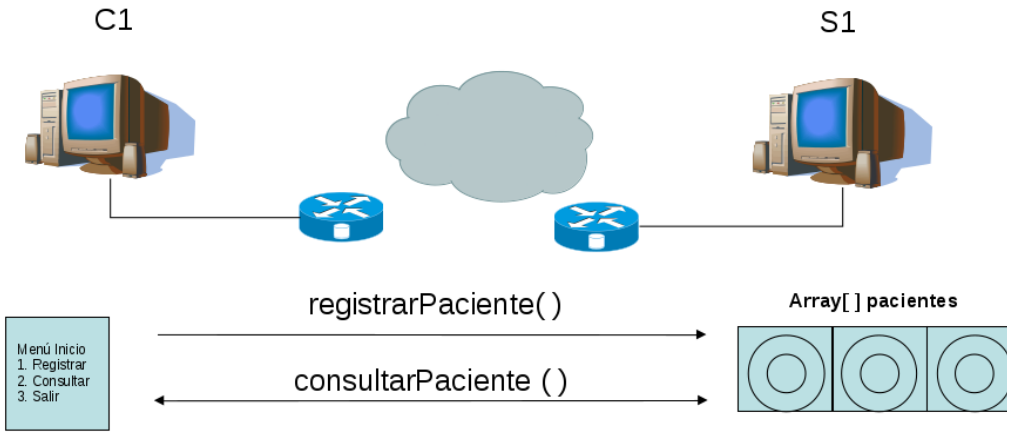

Figura 1: Diagrama de contexto de la aplicación

Tener en cuenta: La solución de este ejercicio debe ser comprimida en formato rar y enviada a la plataforma. El nombre del archivo comprimido debe seguir el siguiente formato lsd_corba_p1_apellidoN1_apellidoN2.rar. Donde apellidoN1 corresponde al primer apellido de uno de los integrantes y apellidoN2 corresponde al primer apellido del segundo integrante del grupo. En la carpeta se deben especificar el rol de cada integrante.

Organizar los archivos de acuerdo a la estructura de directorios que se muestra en la Figura 2.

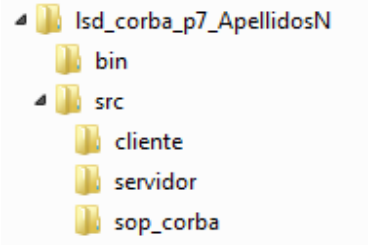

Figura 2: Estructura de directorios

1. Guía de compilación y ejecución.

El primer paso para crear aplicaciones basadas en CORBA es especificar las operaciones remotas utilizando el lenguaje de definición de interfaces (IDL), el cual puede ser mapeado a una variedad de lenguajes de programación.

La interface IDL para la práctica es la siguiente:

```
module sop_corba{

  interface GestionPacientes{
          struct pacienteDTO{
                  string nombre;
                  string apellido;
                  long edad;
                  long numeroHabitacion;
          };

    void registrarPaciente(in pacienteDTO objAnteproyecto,out boolean resultado);
    boolean consultarPaciente(in long numeroHabitacion, out pacienteDTO objPacienteResultado);
   };
};
```

Compilar la interface idl utilizando el siguiente comando:

idlj -fall operaciones.idl

El compilador jidl de java permite convertir una interface definida en lenguaje IDL a correspondientes interfaces java, clase y métodos, los cuales pueden ser utilizados para implementar el código del cliente y servidor.

Este comando generará varios archivos. Revisar el contenido del directorio de trabajo desde donde ejecuto este comando. Un nuevo subdirectorio será creado, debido a que el módulo 'sop_corba' será mapeado como un paquete.

La opción - **fall** genera los archivos del stub y skeleton para el cliente y servidor. Además genera el paquete **GestionPacientesPackage, el cual contiene clases que proveen operaciones para leer y escribir sobre argumentos de las operaciones remotas.**

2. Utilizar y completar las plantillas del cliente y el servidor

Utilizar las plantillas que se encuentran almacenadas en el sitio destinado al curso. Crear los archivos fuente ClienteDeObjetos.java, ServidorDeObjetos.java y GestionPacientesImpl.java, teniendo en cuenta que:

ClienteDeObjetos.java: Implementa la lógica de negocio del cliente, consulta una referencia del servant y llama las operaciones del objeto remoto a través de un menú.

GestionPacientesImpl.java: Hereda de GestionPacientesPOA. Implementa la lógica de las operaciones definidas en la interface IDL.

ServidorDeObjetos.java: Implementa la lógica para crear el servant y registrarlo en el ns.

Distribuir estos archivos fuente en los subdirectorios pertinentes de src/. Para registrar el servant y consultarlo se utilizara como nombre "**objPaciente**".

3. Compilar los códigos fuente

Compilar los códigos fuente del ítem 2 y los stubs y skeleton que fueron generados por el precompilador idlj en el paso 1, por medio de los siguientes comandos:

Para compilar los soportes:
javac -d ../bin sop_corba/*.java

Para compilar los fuentes del servidor:
javac -d ../bin servidor/*.java

Para compilar los soportes:
javac -d ../bin cliente/*.java

4. Lanzar el Object Request Broker Daemon (ORBD)

Antes de correr el cliente y servidor, se debe correr el ORBD, el cual es un servicio de nombrado. Un servicio de nombres de CORBA es un servicio que permite a una referencia de un objeto CORBA asociarla con un nombre. El nombre del enlace se puede almacenar en el servicio de nombres, y un cliente puede proporcionar el nombre para obtener la referencia al objeto deseado.

a. Lanzar el orbd mediante el comando:

orbd –ORBInitialHost dir_IP -ORBInitialPort N_Puerto

b. Ubicarse en el directorio bin/. Lanzar el Servidor mediante el comando:

java servidor.ServidorDeObjetos -ORBInitialHost dir_IP -ORBInitialPort N_Puerto

dir_IP y N_Puerto son la dirección ip y puerto donde se encuentra escuchando el servicio orbd.

c. Ubicarse en el directorio bin/. Lanzar el Cliente mediante el comando:

java cliente.ClienteDeObjetos -ORBInitialHost dir_IP -ORBInitialPort N_Puerto

dir_IP y N_Puerto son la dirección ip y puerto donde se encuentra escuchando el servicio orbd.