UNIVERSIDAD CARLOS III DE MADRID
DEPARTAMENTO DE INFORMÁTICA
INGENIERÍA EN INFORMÁTICA.
DESARROLLO DE APLIACIONES DISTRIBUIDAS

18 de febrero de 2010

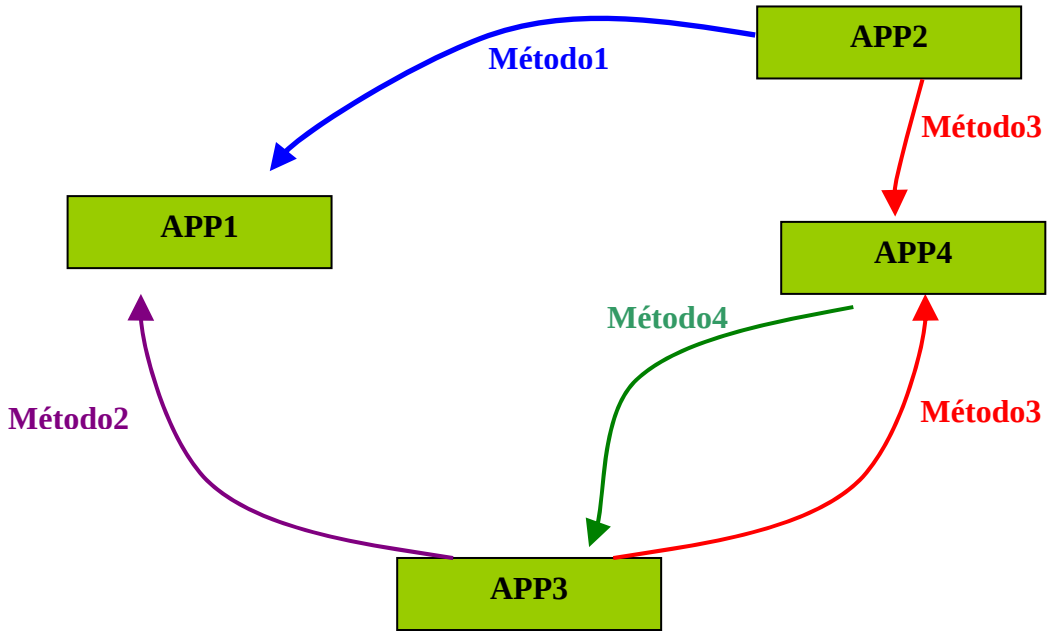
Para la realización del presente examen se dispondrá de **2 horas y media. NO** se podrán utilizar libros ni apuntes.

Ejercicio 1. (2 puntos) Se desea realizar el diseño de un sistema distribuido en JAVA RMI con cuatro aplicaciones denominadas APP1, APP2, APP3 y APP4. La figura representa mediante flechas el esquema de invocación de métodos remotos entre las aplicaciones. El sistema tiene los siguientes requisitos:

1. Sólo APP1 y APP4 pueden tener activo y público (localmente) un rmiregistry.
2. APP3 debe poder acceder de forma dinámica a los *stubs* que necesite.
3. Los métodos remotos Método1 y Método2 pertenecen a la misma clase de APP1.
4. APP3 debe tener restringido el acceso al Método1.

Bajo estos requisitos se pide lo siguiente:

1. Indicar los roles (de servidor o de cliente) de APP1, APP2, APP3 y APP4. Justificar la respuesta.
2. Comentar qué interfaces tendrán almacenados localmente APP1, APP2 y APP3. Indicar qué métodos estarán declarados en cada interface.
3. Indicar la distribución de los *stubs* que cada aplicación requiere almacenar.



Ejercicio 2. (1.5 puntos) Modelos multidifusión. Define brevemente los tipos de sistemas multidifusión existentes.

SOLUCIÓN:
No fiable, Fiable, Sin orden (Los mensajes no tienen orden entre procesos, y los de un mismo proceso), FIFO (Se mantiene el orden de los mensajes de un mismo proceso (distintos entrelazados)), Orden casual (Relación sucede-antes/casual entre todos los mensajes), Orden atómico (Relación atómica sucede-antes/casual entre ciertos mensajes).

Ejercicio 3. (1.5 puntos) Define brevemente los siguientes elementos de una arquitectura corba: ORB (Object Request Broker), COS (Common Object Services) y IOR (Interoperable Object Reference).

SOLUCIÓN:
ORB: El ORB es el servicio distribuido que implementa la petición al objeto remoto. Localiza el objeto remoto en la red, le comunica la petición, espera a los resultados y cuando están disponibles se los devuelve al cliente. El ORB implementa transparencia de localización. Se usa exactamente el mismo mecanismo de peticiones, sin importar donde esté localizado. El ORB implementa independencia del lenguaje de programación para las peticiones. El cliente que lanza la petición se puede escribir en uno de los diferentes lenguajes que aceptan objetos ORBA. El ORB hace las conversiones necesarias entre lenguajes de programación.

COS: Otra parte importante del estándar CORBA es la definición de un conjunto de servicios distribuidos para facilitar la integración e interoperabilidad de los objetos distribuidos.
Los servicios de CORBA, conocidos como CORBA Services o COS, se encuentran definidos
en la parte superior del ORB. Están definidos como objetos CORBA estándares con interfaces IDL, algunas veces llamados Object Services.

IOR: Protocolo Interoperable Object Reference (IOR). Codifica la siguiente información: Tipo de objeto. Servidor del objeto. Puerto asociado al servidor del objeto. Clave del objeto.

Ejercicio 4. (2 puntos) Dado el siguiente código de Corba sobre Java, explica brevemente el significado de las líneas destacadas (L1, L2, L3, L4). **Indica cuáles producen comunicaciones entre procesos y el rol de cada proceso.**

```
public static void main(String[] args) {
       try {
            Properties props = System.getProperties();
            props.put("org.omg.CORBA.ORBInitialPort", "1050");
            props.put("org.omg.CORBA.ORBInitialHost", "<MyHost>");
            ORB orb = ORB.init(args, props);

             POA rootPOA = POAHelper.narrow(
                  orb.resolve_initial_references("RootPOA"));

L1           MessageServerImpl msImpl = new MessageServerImpl();
L2           rootPOA.activate_object(msImpl);
             MessageServer msRef = MessageServerHelper.narrow(
                  rootPOA.servant_to_reference(msImpl));

             NamingContext namingContext = NamingContextHelper.narrow(
                  orb.resolve_initial_references("NameService"));

            NameComponent[] nc = { new NameComponent("MessageServer", "") };
L3           namingContext.rebind(nc, msRef);
            rootPOA.the_POAManager().activate();

L4           orb.run();
          }
```

Solución:

- **L1: Crear instancia de la clase remota (emplea implementación que contiene el código asociado a la clase).**
- **L2: Activación de la clase remota en el Adaptador de Objetos de modo que sea posible accederla remotamente.**
- **L3: Registro del objeto con la etiqueta anterior. La opción de rebind permite sobrescribir cualquier otro objeto que tuviera la misma etiqueta. Comunicaciones entre el proceso main y el proceso que ejecuta el registro CORBA.**
- **L4: Creación de instancia e inicialización de objeto ORB. Necesita la dirección y el número de puerto del orbd dada al comienzo a través de properties.**

Ejercicio 5. (1.5 puntos) ¿Qué métodos de activación de objetos existen en .NET Remoting? ¿Cuáles de ellos mantienen el estado entre múltiples invocaciones?

SOLUCIÓN:
Modos de activación:
Activados por el servidor (well-known objects): Se activan cada vez que el cliente invoca el objeto Reduce tráfico de red
Existen dos clases: Singleton (Existe un único objeto atendiendo a múltiples clientes), Singlecall (Creados y destruidos en cada invocación de un cliente).
Activados por el cliente: Se activa cada vez que el cliente crea el objeto. Sólo sirve a un único cliente

Ejercicio 6. (1.5 puntos)
¿Qué protocolo de comunicación utiliza REST? ¿A través de qué representación se manipulan e identifican recursos con REST?

Se desea realizar una aplicación de REST sobre Facebook. Esta aplicación debe tener dos funcionalidades:

1. Una para acceder a los recursos del usuario, lista de amigos que han escrito en el muro, y lista de amigos que han comentado en las fotos.
2. Otra para crear un nuevo contenido en el muro de Facebook.

Indica qué métodos (asociados al protocolo de comunicación) emplearías para cada funcionalidad.

SOLUCIÓN: El protocolo es HTTP, la representación es URI o URL. Y los métodos son el GET para acceder a los recursos y POST para crear un nuevo contenido.