

**ACTIVIDAD 3** OBJETIVOS: Caracterizar el mecanismo de Invocación a Objeto Remoto (ROI).

Ordene, según el orden en que tienen lugar, los siguientes pasos de una ROI. ¿Falta algún paso para que todo el mecanismo ROI funcione? Si es así, descríbalos.

- 4 A. El método llamado finaliza y se desbloquea el esqueleto
- 6 B. El proxy desempaqueta los resultados y los devuelve al proceso cliente
- 1 C. El proceso cliente invoca el método del proxy local relacionado con el objeto remoto
- 5 D. El esqueleto empaqueta los resultados y llama al ORB, el cual hace llegar el mensaje al proxy
- 2 E. El proxy empaqueta los argumentos y, utilizando la referencia al objeto, llama al ORB
- 3 F. El ORB gestiona la invocación, haciendo que el mensaje llegue al esqueleto.
- Handwritten notes:*  
 → Sí. El esqueleto desempaqueta argumentos e invoca al método remoto entre 3 y 4 quedando a la espera de que finalice.  
 C E F → ADB  
 el esqueleto invoca al método quedando a la espera hasta que finalice.

**ACTIVIDAD 4** OBJETIVOS: Caracterizar el mecanismo de Invocación a Objeto Remoto (ROI).

Sobre el paso de argumentos en el mecanismo de comunicación ROI, indique si las siguientes afirmaciones son Verdaderas (V) o falsas (F), justificando su respuesta:

V	En un paso de parámetros por referencia, ésta puede pertenecer a uno nodo que no sea ni el invocador ni el invocado. <b>Justificación:</b>
V	Los argumentos se pueden pasar por valor, no sólo mediante referencias a objetos. <b>Justificación:</b>
F	Los argumentos que se pasan por referencia se serializan antes de transmitirlos al nodo destino. <b>Justificación:</b> <i>Handwritten note:</i> → por valor se serializan
V	En un paso de parámetros por referencia, ésta puede pertenecer al nodo invocador. <b>Justificación:</b>

**ACTIVIDAD 5** OBJETIVOS: Caracterizar el mecanismo de Invocación a Objeto Remoto (ROI).

En el mecanismo ROI, la creación de objetos puede realizarse mediante dos procedimientos distintos. ¿Cuáles son? Indique, para los siguientes pasos, a qué procedimiento se corresponde y en qué orden tienen lugar (Nota: un mismo paso puede pertenecer a los dos procedimientos).

→ a iniciativa del cliente o del servidor

- A. El servidor <sup>factoría</sup> obtiene una referencia al objeto.
- B. Un proceso (cliente) solicita a un servidor (factoría) que cree un determinado objeto.
- C. El proceso servidor usa la referencia del objeto para registrarlo en un servidor de nombres, proporcionando una cadena de texto como nombre del objeto.
- D. El servidor (factoría) devuelve al cliente una copia de la referencia del objeto que ha solicitado crear.
- E. Un proceso que conozca el nombre utilizado para registrar el objeto contacta con el servidor de nombres y obtiene una referencia al objeto.
- F. Un proceso crea un objeto y lo registra en el ORB.
- G. El servidor <sup>factoría</sup> crea el objeto (que le han solicitado crear) y lo registra en el ORB.

cliente: BGAD

A iniciativa del servidor: FACE

**ACTIVIDAD 6** OBJETIVOS: Caracterizar el mecanismo de comunicación Java RMI.

Indique si las siguientes afirmaciones son verdaderas (V) o falsas (F). Justifique su respuesta.

F	Los objetos remotos deben residir en la misma JVM. <b>Justificación:</b>
V	Java construye automáticamente los esqueletos y los proxies a partir de la especificación de la interfaz del objeto remoto. <b>Justificación:</b>
F	Todos los objetos que se pasan como argumentos en Java RMI deben ser remotos, no permitiéndose por tanto pasar <u>objetos locales</u> . <b>Justificación:</b> ↳ objetos locales se serializan y se pasan por valor
F	El servidor de nombres de Java RMI almacena, para cada objeto registrado, su nombre y esqueleto. <b>Justificación:</b> → referencia simbólico
F	El mecanismo de comunicación de Java RMI no tiene nada que ver con el mecanismo de comunicación ROI (invocación a objeto remoto). <b>Justificación:</b>

**ACTIVIDAD 8** OBJETIVOS: Caracterizar el mecanismo de comunicación Java RMI.

Se desea crear un servicio básico de *eco*, que ofrece únicamente un método remoto que retorna la cadena de caracteres recibida como argumento, pero pasándola a mayúsculas.

1.- Actualice la siguiente definición del servicio “eco”, para que pueda ser utilizado de forma remota.

```
import java.rmi.*;

interface ServicioEco extends Remote
{
    String eco (String s) throws RemoteException;
}
```

2.- Actualice la clase ServicioEcoImpl (que se muestra a continuación) para que implemente el servicio remoto.

```
import java.rmi.*;
import java.rmi.server.*;

class ServicioEcoImpl extends UnicastRemoteObject implements ServicioEco {
    public String eco (String s) throws RemoteException
    {
        return s.toUpperCase();
    }
}
```

3.- En la clase ServidorEco, que actúa como servidor, indique cómo se inicia el servicio remoto y cómo se hace accesible usando rmiregistry. ¿Con qué nombre se ha registrado el servicio?

```
import java.rmi.*;
import java.rmi.registry.*;

class ServidorEco{
    static public void main (String args[]){
        if (args.length!=1) {
            System.err.println("Uso: ServidorEco numPuertoRegistro");
            return;
        }

        try {
            ServicioEco srv = new ServicioEcoImpl();
            Registry reg = LocateRegistry.getRegistry("localhost",1099);
            reg.rebind("Eco", srv);
        }
    }
}
```

```

        catch (RemoteException e) {
            System.err.println("Error de comunicacion: " +
                e.toString());
            System.exit(1);
        }
        catch (Exception e) {
            System.err.println("Excepcion en ServidorEco:");
            e.printStackTrace();
            System.exit(1);
        }
    }
}

```

4. Finalmente, se muestra a continuación el código del cliente (fichero ClienteEco.java). Actualice dicho código para que el cliente obtenga una referencia remota asociada al servicio “eco” que hemos implementado.

```

import java.rmi.*;
import java.rmi.registry.*;
class ClienteEco {
    static public void main (String args[]) {
        if (args.length<2) {
            System.err.println("Uso: ClienteEco hostregistro
                numPuertoRegistro ...");
            return;
        }

        try { Registry reg = LocateRegistry.getRegistry(args[0], 1099);
            ServicioEco srv = (ServicioEco) reg.lookup("Eco");

            for (int i=2; i<args.length; i++)
                System.out.println(srv.eco(args[i]));
        } catch (Exception e) {
            System.err.println("Excepcion en ClienteEco:");
            e.printStackTrace();
        }
    }
}

```

**ACTIVIDAD 7** OBJETIVOS: Caracterizar el mecanismo de comunicación Java RMI.

Dada la siguiente aplicación:

```
public interface Hola {
    String saluda();
}
class ImpleHola implements Hola {
    ImpleHola() {...} // constructor
    public String saluda() {return "Hola a todos";}
}
...
Hola h = new ImpleHola();
System.out.println(h.saluda());
```

Se desea implementarla como una aplicación Java RMI. Indique brevemente todos los pasos que se deben realizar, tanto para el desarrollo del cliente como para el desarrollo del servidor.

**ACTIVIDAD 9** OBJETIVOS: Caracterizar las referencias a recursos en REST

Para las siguientes llamadas, indique:

- Si siguen el estándar REST o no. En caso de no ser REST, realice los cambios necesarios para que sean REST.
- ¿Qué es lo que pretende realizar esa llamada?

1) GET <https://api.github.com?type=users&id=captainkidd> → No es REST

GET <https://api.github.com/users/captainkidd>  
Devuelve la información sobre el usuario captainkidd de github

2) GET <https://api.github.com/users/captainkidd/edit> No es REST  
PUT <https://api.github.com/users/captainkidd> y en el cuerpo del mensaje se proporciona el contenido actualizado del usuario captainkidd. Actualiza la información del usuario captainkidd de github.

3) GET [https://api.github.com/gists/page/22/per\\_page/2](https://api.github.com/gists/page/22/per_page/2)

Obtener la información de los gists, <sup>?page=22&per-page=2</sup> indicando por parámetro cuáles nos interesan

4) POST <https://weatherapp.com/messages>

OK. Crea un nuevo mensaje en el servidor. En el cuerpo del mensaje POST se proporciona el contenido del mensaje a crear.

5) GET weatherapp.com/wheaterLookup.do?zipcode=46017 /zipcodes/46017 No es REST

Devuelve la información del tiempo de código postal 46017

6) GET weatherapp.com/getMessages.do?id=10  
/messages/10

Devuelve el mensaje con identificador 10

7) GET https://myapp.com/deleteOrder.do?id=10 → No es REST

Para borrar la orden con id=10

DELETE https://myapp.com/orders/10

8) DELETE https://myapp.com/messages/10

OK. Elimina el mensaje con identificador 10

9) GET https://myapp.com/messages

OK. Devuelve todos los mensajes.

### ACTIVIDAD 10 OBJETIVOS: Caracterizar las operaciones en REST

Dadas las siguientes URIs, indique el resultado que se obtiene al aplicar sobre ellas los métodos HTTP indicados:

Método	URI	Resultado
GET	/messages	Obtiene todos los mensajes
POST		Añade un nuevo mensaje (que debe proporcionarse en el contenido del POST)
GET	/messages/10	Obtiene el mensaje con id=10
PUT		Actualiza el contenido del mensaje con id=10
DELETE		Elimina el mensaje con id=10
GET	/messages/10/comments	Obtiene todos los comentarios del mensaje con id=10
DELETE		Elimina todos los comentarios del mensaje con id=10
POST		Crea un nuevo comentario para el mensaje con id=10
PUT		Reemplaza todos los comentarios del mensaje 10 con una nueva lista de comentarios.

**ACTIVIDAD 11** OBJETIVOS: Caracterizar el mecanismo de comunicación JMS

ENUNCIADO: Dadas las siguientes afirmaciones, justifique si son Verdaderas (V) o Falsas (F):

1. Generalmente es preferible usar JMS frente a Java RMI cuando es necesario que todos los componentes de la aplicación estén <u>simultáneamente en ejecución</u> . JUSTIFICACIÓN: <i>al revés, no están en ejecución</i>	F
2. La comunicación se considera débilmente acoplada. JUSTIFICACIÓN:	V
3. Un cliente JMS es un objeto administrado. JUSTIFICACIÓN: <i>los objetos administrados son las factorías de conexiones y los destinos (colas y temas)</i>	F
4. Las colas de mensajes se crean normalmente utilizando las herramientas administrativas del proveedor JMS. JUSTIFICACIÓN:	V
5. Un proveedor JMS es una empresa que ofrece <u>servicios de consultoría</u> relativos a JMS. JUSTIFICACIÓN:	F
6. Los objetos que implementan la interfaz Queue se crean llamando a métodos de la interfaz JMSConsumer. JUSTIFICACIÓN: <i>→ Las colas se crean desde herramientas administrativas</i>	F
7. Los objetos que implementan la interfaz JMSProducer se crean llamando a métodos de la interfaz JMSContext. JUSTIFICACIÓN:	V
8. La comunicación normalmente es persistente. JUSTIFICACIÓN:	V
9. La comunicación es sincrónica en la respuesta. JUSTIFICACIÓN: <i>asíncronica. El emisor sigue cuando entrega el mensaje al proveedor JMS</i>	F
10. El direccionamiento empleado es del tipo directo. JUSTIFICACIÓN: <i>→ indirecto a colas, temas... a través del proveedor JMS</i>	F