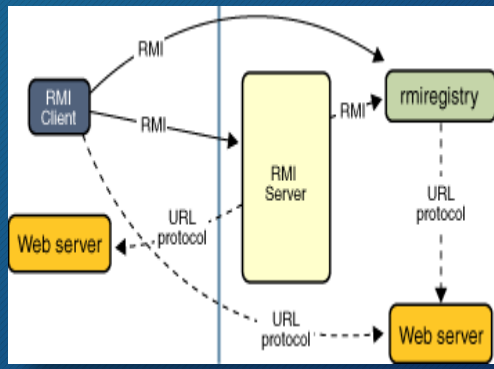
# Sistemas Distribuidos




RMI

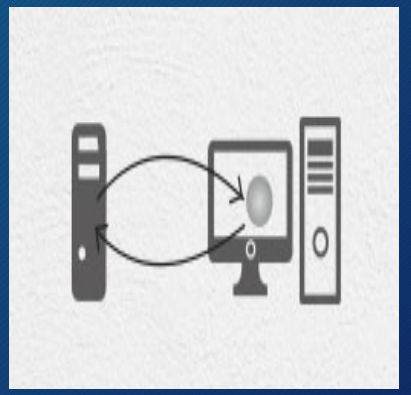
Remote Method Invocation

Java RMI

Prof. Alejandro Reyes Ortiz

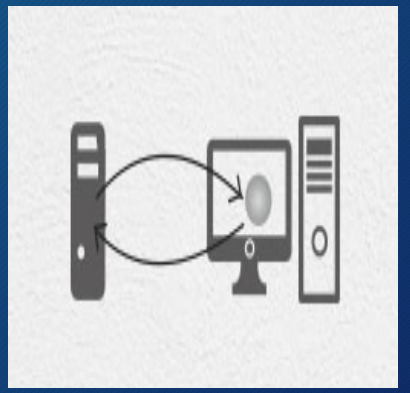
# Invocación de Métodos Remotos (RMI)

- El mecanismo RMI permite que una aplicación se comunique con objetos que residen en programas que se ejecutan en máquinas remotas.
- Un objeto que se ejecuta en una JVM usará métodos de otro objeto que se ejecuta en otra JVM (local o remota)
  - Un paso más allá de los sockets
  - Beneficios del manejo de objetos sobre RPC

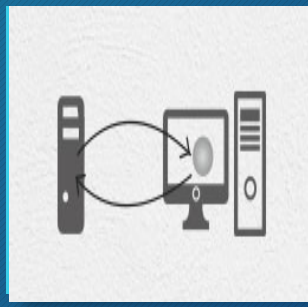


# Invocación de Métodos Remotos (RMI)

En esencia, en lugar de crear un objeto, el programador liga el objeto remoto con un representante local, conocido como *stub*.

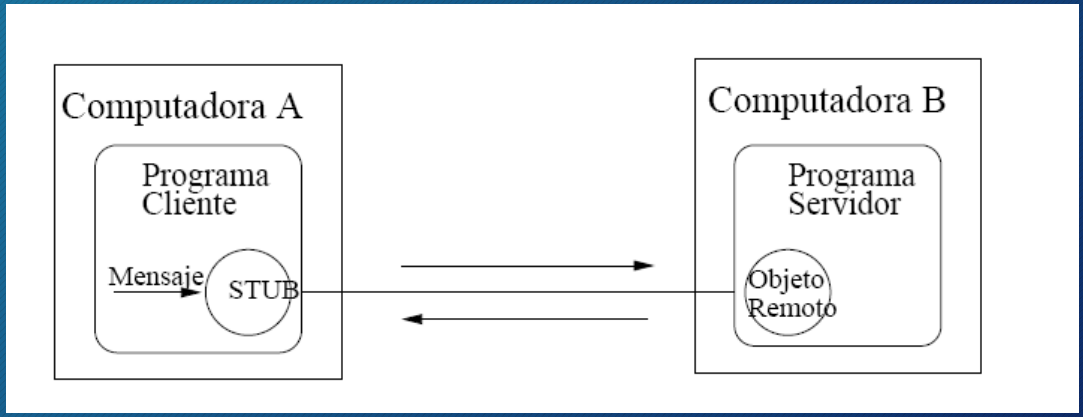


# Resumen del proceso de invocación

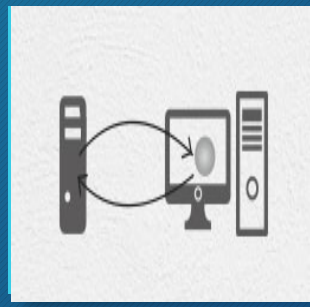


Una aplicación cliente ejecutándose en una máquina A, que envía un mensaje a un objeto remoto contenido en un servidor que se ejecuta en una máquina B.

Cuando la aplicación cliente desea enviar un mensaje al objeto remoto, lo hace mediante su representante local (*stub*) la petición se transmite a la máquina que contiene al objeto real, donde el método es invocado y un resultado es retornado, de modo que la aplicación cliente puede obtener la respuesta apropiada.



# Invocación de métodos remotos (RMI)

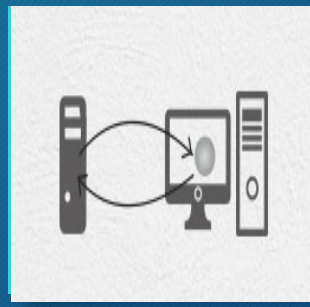


**JAVA RMI**

- Forma parte del entorno estándar de ejecución de Java y proporciona un mecanismo simple para la comunicación de servidores en aplicaciones distribuidas basadas exclusivamente en Java.

- RMI se caracteriza por la facilidad de su uso en la programación por estar específicamente diseñado para Java; proporciona paso de objetos por referencia (no permitido por Sockets, ni con RPC).

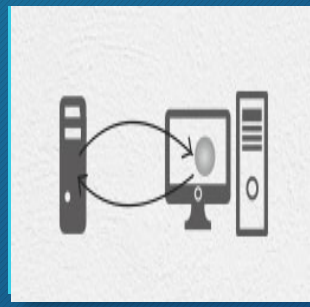
# Invocación de métodos remotos (RMI)



**JAVA RMI**

- A través de RMI, un programa Java puede exportar un objeto, que estará accesible a través de la red y el programa permanece a la espera de peticiones en un puerto.
- A partir de ese momento, un cliente puede conectarse e invocar los métodos proporcionados por el objeto.

# Invocación de métodos remotos (RMI)



**Dos fases fundamentales**

La invocación se compone de los siguientes pasos:

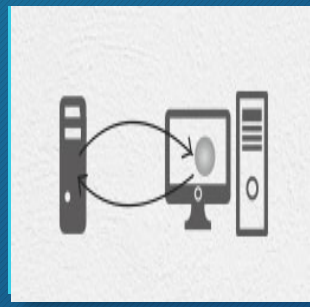
Localizar objetos remotos

- Registrados mediante el registro RMI
- Pasados por referencia en invocaciones remotas

Comunicarse con objetos remotos

- Gestionado por el servidor RMI, para el usuario es como llamar a métodos locales

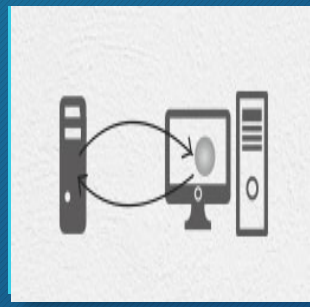
# Invocación de métodos remotos (RMI)



La invocación (comunicación con los objetos remotos) se compone de los siguientes pasos:

- Encapsulado (marshalling) de los parámetros .
- Invocación del método (del cliente sobre el servidor). El invocador se queda esperando una respuesta.
- Al terminar la ejecución, el servidor **serializa** el valor de retorno (si lo hay) y lo envía al cliente.
- El código cliente recibe la respuesta y continúa como si la invocación hubiera sido local.

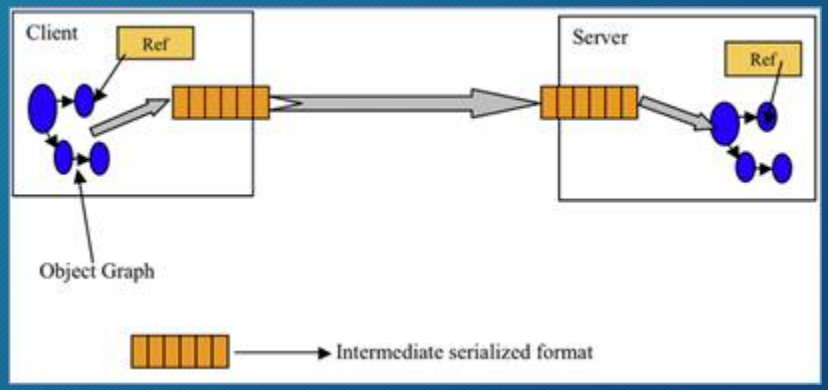
# Invocación de métodos remotos (RMI)



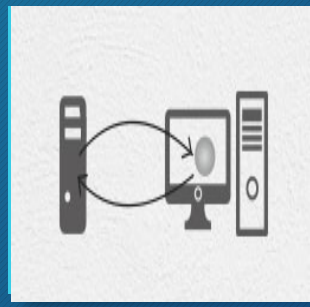
**Serialización de objetos**

Para que un programa en Java pueda convertir un objeto en un conjunto de *bytes* y pueda luego recuperarlo, el objeto necesita ser **Serializado**.

Al poder convertir el objeto a *bytes*, ese objeto se puede enviar a través de red y después reconstruirlo al otra lado de la red.



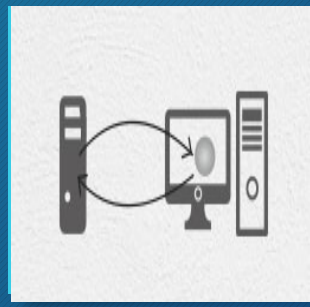
# Invocación de métodos remotos (RMI)



**Implementación**

1. Definir interfaz con los métodos remotos
   - Será conocida por cliente y servidor
2. Implementar el servidor
   - El elemento que dará el servicio de la interfaz
3. Instanciar el servidor y registrarlo :
   - Referencia remota al servidor generada por RMI para el uso de los clientes
4. Implementar el cliente que usará el servicio

# Componentes de un sistema basado en Java RMI

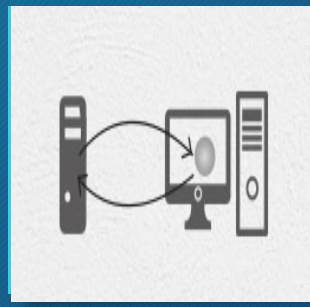


**JAVA RMI**

Toda aplicación RMI se descompone en 3 elementos:

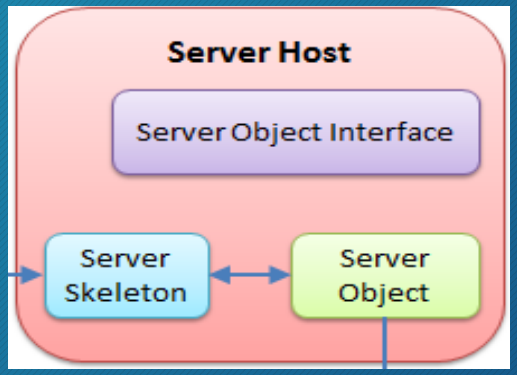
- Un servidor
- Un cliente.
- Registro de objetos: rmi-registry

# Componentes de un sistema basado en Java RMI



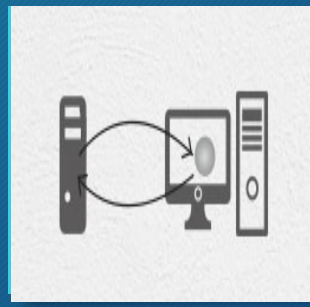
## Servidor RMI

- Un servidor, que crea los objetos remotos, crea referencias para hacerlos accesibles, y espera a que el cliente los invoque.



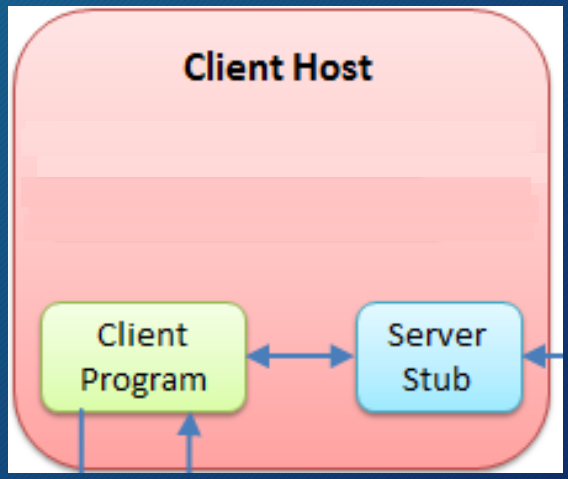

- Server skeleton: una clase que encapsula la funcionalidad de los objetos del servidor y los expone al cliente.

# Componentes de un sistema basado en Java RMI

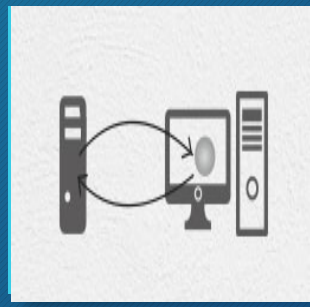


**CLIENTE RMI**

- Un cliente, que obtiene una referencia (Server stub) a objetos remotos del servidor, y los invoca.
- El cliente debe tener la compilación de la clase que implementa el servidor : *Server Stub*.



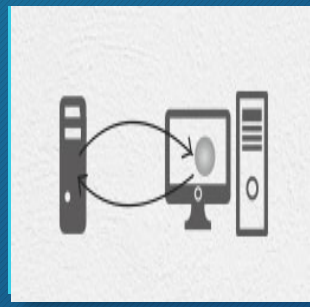
# Componentes de un sistema basado en Java RMI



**RMI-REGISTRY**

- Registrar un nombre y el lugar de los objetos remotos. Esto lo realiza uno o más servidores que contienen objeto y los desean publicar.
- El registro se convierte en un servidor de nombres con su propia dirección y puerto.
- Permitir a un cliente ligar su *stub* local para tener acceso al objeto remoto contenido en la aplicación servidor.

# Componentes de un sistema basado en Java RMI



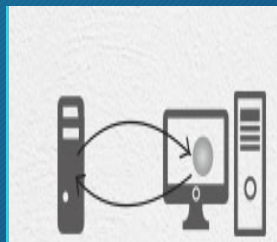
**RMI-REGISTRY**

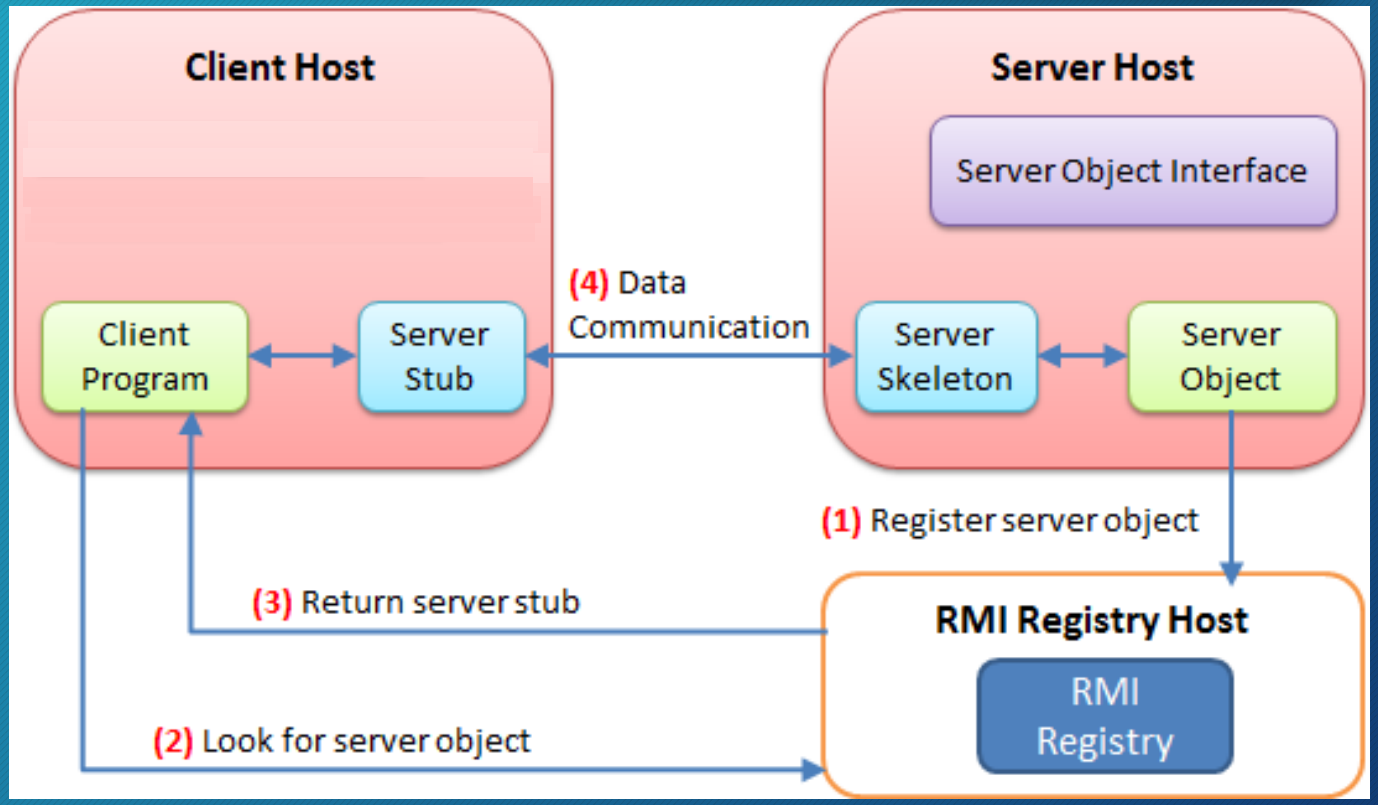
El registro RMI viene incluido a partir de la versión 1.5 de Java.

El RMI-Registry es como las páginas amarillas, donde le indicas el nombre del objeto y él devuelve la referencia al objeto remoto.

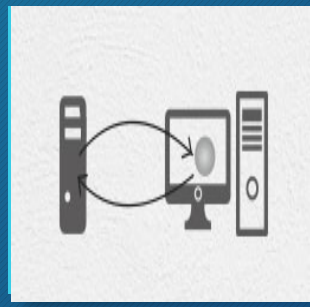
Una vez obtenida esa referencia se pueden invocar los métodos.

# Componentes de un sistema basado en Java RMI





# Java RMI



**IMPLEMENTACIÓN**

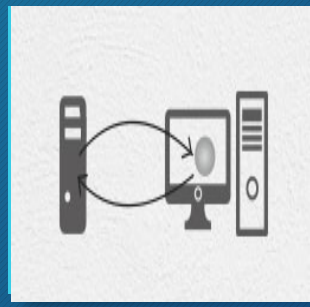
1. Interfaz: clase que extiende java.rmi.Remote

2. Servidor: clase que implementa la interfaz
    - Puede tener más métodos que los de la interfaz

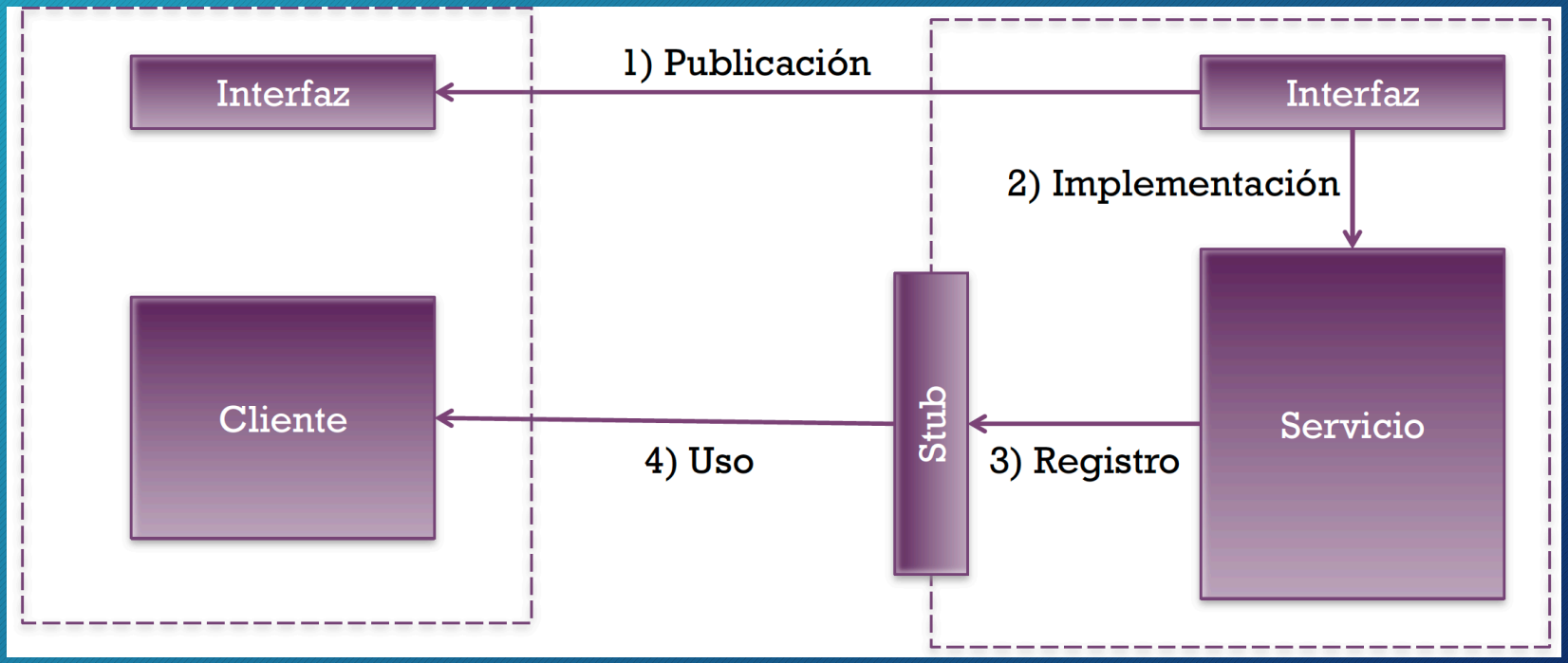
3. Stub: instancia de la interfaz asociada a un servidor
    - Sólo contiene los métodos de la interfaz
    - Es la que se registra en RMI

4. Cliente: cualquier clase que localice el stub y use su interfaz

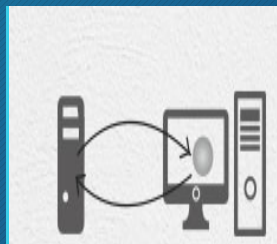
# Java RMI



## IMPLEMENTACIÓN



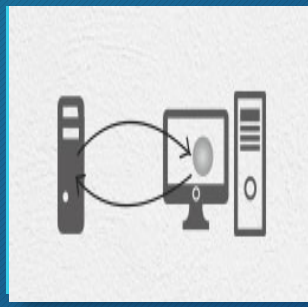
# Manejo de Java RMI



La clase *Naming* contiene los siguientes métodos que permiten el acceso a objetos remotos utilizando un URL para especificar el nombre y lugar del objeto remoto.

| Método | Responsabilidad |
|---|---|
| `bind(url,object)` | Liga un nombre a un objeto remoto. El nombre se especifica como un URL. |
| `list(url)` | Retorna un arreglo de cadenas representando los URLs en el registro. |
| `lookup(url)` | Retorna un objeto remoto (un *stub*) asociado con el URL. |
| `rebind(url, object)` | Similar al `bind()`, pero reemplaza la asociación hecha. |
| `unbind(url)` | Remueve la asociación entre el objeto remoto y el URL. |

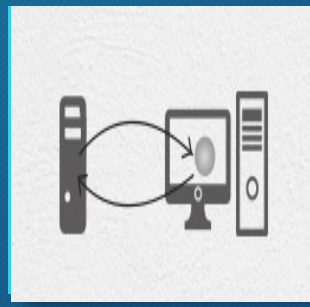
# Manejo de Java RMI



El URL se presenta en la forma rmi://host:port/objectName, donde:

| Componente | Valor por defecto | Especificación |
|---|---|---|
| `rmi` | `rmi` | El método de acceso (debe ser `rmi`). |
| `host` | `localhost` | La máquina servidor (`host`). |
| `port` | `1099` | El puerto utilizado. |
| `objectName` | - | El nombre del objeto remoto. |

# Manejo de Java RMI



**El proceso para levantar una aplicación Java RMI:**

1. Inicialización:

- ✓ Se ejecuta la aplicación *rmiregistry* en el servidor.
- ✓ La aplicación servidor que contiene al objeto remoto se arranca.
- ✓ La aplicación servidor liga (usando bind() o rebind()) al objeto remoto con la aplicación *rmiregistry*.
- ✓ La aplicación cliente se inicia.
- ✓ La aplicación cliente busca (lookup()) al objeto remoto.

2. Acceso:

- ✓ Los mensajes se envían del cliente al servidor donde se encuentra el objeto real. El método correspondiente al mensaje se invoca, y el resultado se retorna.

# Manejo de Java RMI