Asignación de Prácticas Número 11
Programación Concurrente y de Tiempo Real

Kevin J. Valle Gómez

Departamento de Ingeniería Informática
Universidad de Cádiz

2020

Contenido

Objetivos de la práctica

- Aplicar y consolidar los conocimientos adquiridos sobre el framework RMI.
- Desarrollar soluciones que permitirán:
  - Comprender la diferencia entre RMI y el ya conocido MPJ-Express.
  - Definir procedimientos que puedan efectuarse de forma remota: en máquinas distintas.
  - Practicar la creación de los tres elementos principales: interfaz, servidor y cliente.

Un breve repaso...

- Programación distribuida: concepto de paso de mensajes.
- MPJ-Express.
  - Muy bajo nivel.
  - Debemos porveer la estructura de los mensajes que comunican los procesos.
  - *send* y *receive*.
- Cliente y Servidor.

Introducción a RMI

- Supone un paso más en la programación distribuída.
- Modelo de llamada a procedimiento remoto (RPC).
- **Invocación Remota de Métodos (RMI).**

Introducción a RMI - Ventajas

- Mayor nivel de abstracción.
- No es necesario diseñar la arquitectura de los mensajes.
- El programador se centra en la lógica de la aplicación: no en la comunicación.
- Se definen los procedimientos que pueden efectuarse de forma remota (en otra máquina).

Introducción a RMI - ¿Cómo funciona?

El funcionamiento es muy sencillo:

1. El cliente envía la petición al servidor y queda bloqueado hasta recibir respuesta.
2. El servidor recibe la petición, realiza el procesamiento y devuelve la respuesta.
3. El cliente recibe la respuesta, se desbloquea y continúa su ejecución con normalidad.

A continuación veremos algunos elementos a destacar

El concepto de stub

- *Resguardo o proxy*.
- Módulos de software que enmascara los detalles de la comunicación entre procesos.
- **Cliente:** stub.
- **Servidor:** skeleton. (OJO: A partir de jdk 1.2 no es necesario)

Elementos principales en Java

- Interfaz remota.
- Servidor: implementación de la interfaz definida.
- Cliente: se comunica con la interfaz.

Elementos principales en Java: Interfaz

- Oculta la implementación de los aspectos relativos a los métodos remotos.
- El cliente obtiene una referencia a una interfaz, con la que se comunicará.

A tener en cuenta...

- *public* y extiende a *Remote*.
- Declara todos los métodos ofrecidos por el servidor, pero **no los implementa**.
- Todos los métodos lanzan la excepción *RemoteException*.

Elementos principales en Java: Servidor

- Implementa los métodos declarados en la interfaz.

A tener en cuenta...

- Debe contener una clase que extienda a *UnicastRemoteObject* e implemente a la interfaz.
- Debe tener un constructor que lance la excepción *RemoteException*.
- El método *main* debe:
  - Lanzar un gestor de seguridad.
  - Crear los objetos remotos.
  - Registrar al menos uno de los objetos remotos.
- Linux: rmiregistry
- Windows: start rmiregistry

Elementos principales en Java: Cliente

A tener en cuenta...

- Se comunica con la interfaz.
- Los objetos remotos se pasan por referencia.
- Los clientes que llaman a métodos remotos deberán manejar las excepciones.

Elementos principales en Java - Ejemplo

Observa los ficheros **EjemploRMI1.java* incluídos en el directorio de esta asignación de prácticas.

En resumen...

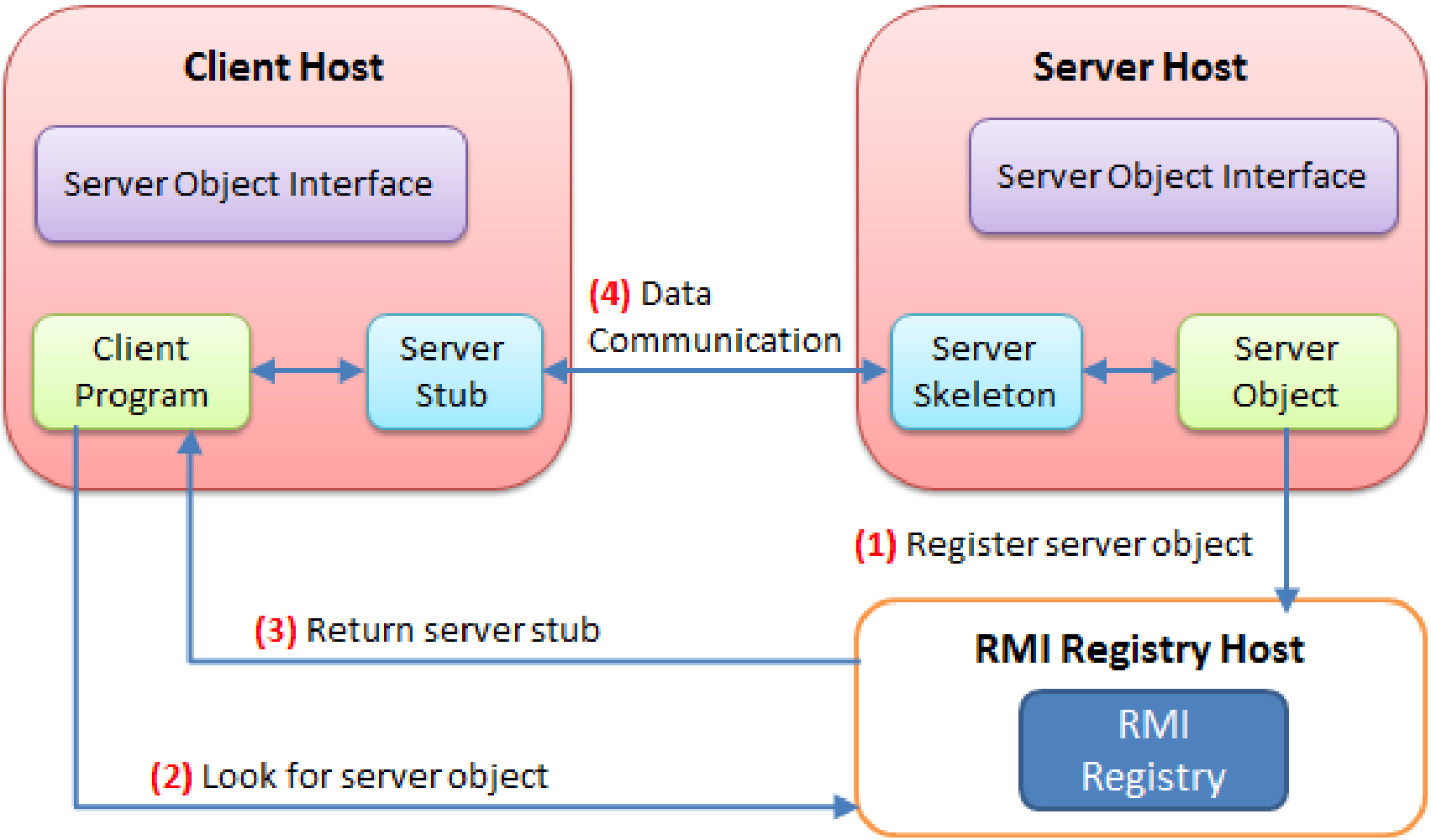


Figure 1: Java RMI Overview

Ejercicio 1: la Bonoloto

- Sistema remoto para jugar a la Bonoloto.
- Seis números aleatorios entre 1 y 49.

Ejercicio 1: ¿Qué debo hacer?

- Se proporciona la interfaz en el Campus Virtual con el nombre *iBonoLoto.java*

```
public void resetServidor() throws RemoteException;
public boolean compApuesta(int[] apuesta) throws
RemoteException;
```

Ejercicio 1: ¿Qué debo hacer? (II)

- El servidor generará la apuesta ganadora:
  - Genera seis números aleatorios entre 1 y 49.
  - Deberán ser acertados para que la apuesta sea ganadora.
  - El servidor responderá si se ha acertado o no.
- Los clientes enviarán sus apuestas al servidor:
  - Un array con seis números que definen la apuesta.

Ejercicio 2: la Biblioteca

- Servidor remoto de base de datos bibliotecarios.
- Combinación de un array de objetos de clase Libro.java junto con la tecnología RMI.
- Implementa dicha biblioteca con gestión remota.

Ejercicio 2: ¿Qué debo hacer? - Clase Libro

- Implementa la clase Libro en Libro.java
- Esta clase debe representar una entrada en la BD
- Algunos datos para las referencias bibliográficas:
  - Autor o autores.
  - Título.
  - ISBN.
  - Año de publicación.
  - Edición.
  - Editorial.

Ejercicio 2: ¿Qué debo hacer? (II) - Interfaz

- Implementa la interfaz en iLibros.java
- Pueden ser útiles los ejemplos proporcionados.
- Declara los métodos que creas necesarios para:
  - Insertar datos en la base de datos.
  - Extraer y consultar datos en la base de datos.

Ejercicio 2: ¿Qué debo hacer? (III) - Servidor

- Implementa el servidor en sLibros.java
- Inicializa un array que contenga objetos de la clase Libro.
  - Servirá como base de datos en este problema.
- Implementa la funcionalidad de las clases declaradas en la interfaz.

Ejercicio 2: ¿Qué debo hacer? (IV) - Cliente

- Implementa el cliente en cLibros.java
- Inicializa el objeto que representa al servidor.
- Haz uso de los métodos para la lectura, escritura y consulta de libros.
- Puedes crear tus propios libros para insertarlos, realizar comparaciones, etc.

Ejercicio 3: el método de Monte-Carlo

- Cálculo remoto de la aproximación a π.
- Problema sobradamente conocido ;-)
  - Volvamos a la asignación de prácticas **número 1**.
- A través del método de Monte-Carlo, iremos calculando la aproximación a π.

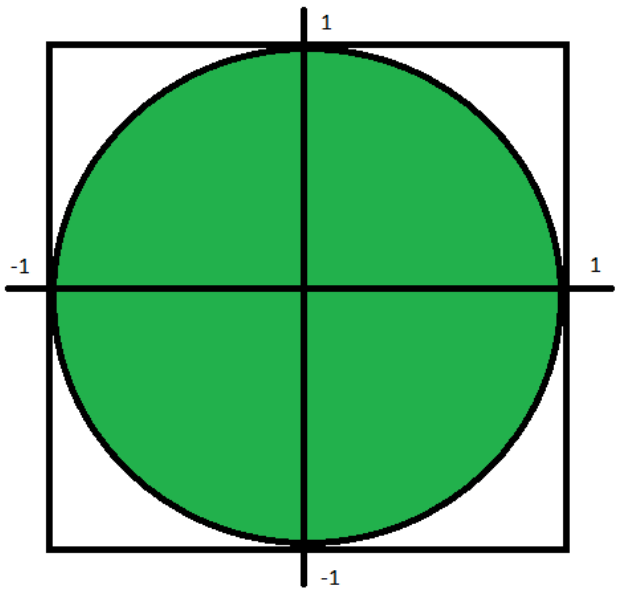
Ejercicio 3: el método de Monte-Carlo

- Anteriormente usamos este método para conocer la aproximación a dos funciones.
  - $f(x) = \sin(x)$
  - $f(x) = x$
- Contábamos los puntos que caían debajo de la curva.
- La cuestión es: ¿cuál es nuestra curva en este problema?

Ejercicio 3: el método de Monte-Carlo

- Considera un círculo de radio = 1.
- Si introducimos este círculo en un cuadrado y escogemos un punto al azar, la probabilidad de que el punto se encuentre dentro del círculo se da como la proporción entre el área del círculo y el cuadrado.

$$P(x^2 + y^2 < 1) = \frac{A_circulo}{A_cuadrado} = \frac{\pi}{4}$$



Ejercicio 3: el método de Monte-Carlo

- Consideraremos acierto cuando el punto esté dentro del área del círculo.
- `bool acierto(double x, double y, double radio)`
- Calcula: $\sqrt{x^2 + y^2} <= radio$
- Aproximación número pi: $\pi = \frac{Aciertos}{Puntos} * 4$

Ejercicio 3: ¿Qué debo hacer? - Interfaz

- Se proporciona la interfaz `iPiMonteCarlo.java`

```
public void reset() throws RemoteException;
public void masPuntos(int nPuntos) throws
RemoteException;
```

Ejercicio 3: ¿Qué debo hacer? - Servidor

- Implementa el servidor en SPiMonteCarlo.java
- El servidor deberá:
  - Implementar los métodos declarados en la interfaz.
  - Guardar el número de intentos y aciertos para poder mostrar la solución aproximada en el momento de la petición.

Ejercicio 3: ¿Qué debo hacer? - Cliente

- Implementa el servidor en CPiMonteCarlo.java
- El cliente deberá:
  - Enviar tantos puntos como desee el usuario al servidor.
  - Mostrar por pantalla el resultado de la aproximación actual.