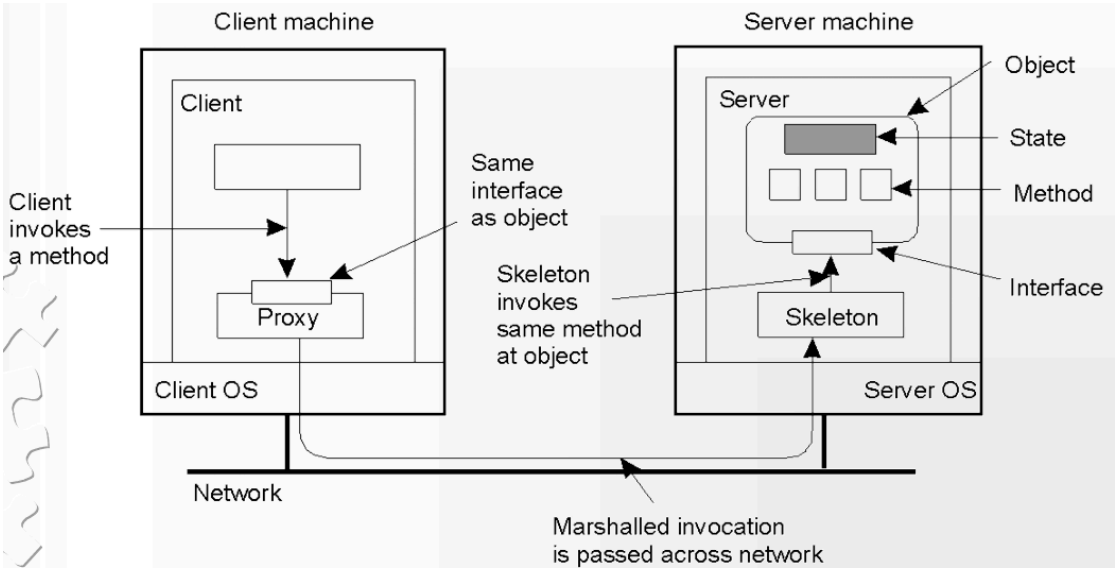
INSTITUTO POLITÉCNICO NACIONAL

ESCUELA SUPERIOR DE CÓMPUTO

Desarrollo de Sistemas Distribuidos

Práctica 5 ‘Objetos Distribuidos - RMI’



Docente:

Carreto Arellano Chadwick

Grupo:

7CM6

Alumnos:

Rodríguez Hernández Marlene Guadalupe

Fecha: martes 14, octubre 2025

Antecedentes

Los objetos distribuidos son aquellos que están gestionados por un servidor y sus clientes invocan sus métodos utilizando un "método de invocación remota". El cliente invoca el método mediante un mensaje al servidor que gestiona el objeto, se ejecuta el método del objeto en el servidor y el resultado se devuelve al cliente en otro mensaje.

Hay distintos métodos, como los siguientes:

- **RMI,** Remote Invocation Method (Sistema de Invocación Remota de Métodos) : Fue el primer framework para crear sistemas distribuidos de Java. Este sistema permite, a un objeto que se está ejecutando en una Máquina Virtual Java (VM), llamar a métodos de otro objeto que está en otra VM diferente. Esta tecnología está asociada al lenguaje de programación Java, es decir, que permite la comunicación entre objetos creados en este lenguaje.
- **DCOM,** Distributed Component Object Model: El Modelo de Objeto Componente Distribuido, está incluido en los sistemas operativos de Microsoft. Es un juego de conceptos e interfaces de programa, en el cual los objetos de programa del cliente, pueden solicitar servicios objetos del programa servidores, en otros ordenadores dentro de una red. Esta tecnología está asociada a la plataforma de productos Microsoft.
- **CORBA,** Common Object Request Broker Architecture: Tecnología introducida por el Grupo de Administración de Objetos OMG, creada para establecer una plataforma para la gestión de objetos remotos independiente del lenguaje de programación.

El método que utilizaremos en esta práctica es el RMI, el cual nos permite distribuir objetos entre distintos procesos o máquinas sin tener que manejar directamente la conexión de bajo nivel como hacíamos en prácticas anteriores al utilizar los sockets.

Planteamiento del problema

Siguiendo los objetos distribuidos, implementaremos el servidor RMI y a su vez crearemos un cliente RMI que pueda interactuar con el servidor, teniendo en cuenta la siguiente estructura:

- Interfaz remota (CarreraInterfaz.java)
    - aqui definimos los metodos que los clientes (tortugas) pueden invocar de manera remota (registrarTortuga, avanzar, obtenerPosiciones e iniciarCarrera)
- Servidor RMI (CarreraServidor.java)
    - implementamos la interfaz RMI y gestionamos la logica la cual lleva el registro de tortugas, controla la meta y la sincronizacion y al mismo tiempo devuelve al cliente su posicion actual o el estado de la carrera

- Cliente (TortugaCliente.java)
    - se conecta al servidor RMI y ejecuta metodos como carrera.registrarTortuga() y carrera.avanzar()
- Registro RMI (rmiregistry)
    - es el proceso de Java que permite publicar los objetos remotos para que otros los encuentren por nombre

Propuesta de solución

Basándonos en la estructura que hemos planteado, al ejecutar el Registro RMI, ejecutaremos también el Servidor RMI este registra el objeto remoto en el registro y cada cliente Tortuga se conecta al registro, y obtiene una referencia al objeto remoto y llama a los métodos (como si estos fueran locales)

Desarrollo de la solución

A continuación se explica el funcionamiento de cada uno de los archivos del código:

- CarreraInterfaz.java

  Esta interfaz es la que usará el servidor y los clientes, aquí es donde ocurre el proceso del RMI, y tambien es donde se encuentran los métodos

```java
import java.rmi.Remote; // se usa a traves de RMI
import java.rmi.RemoteException;
import java.util.Map;

public interface CarreraInterfaz extends Remote {
    String registrarTortuga(String nombre) throws RemoteException;
    String avanzar(String nombre) throws RemoteException;
    Map<String, Integer> obtenerPosiciones() throws RemoteException;
    String obtenerGanador() throws RemoteException;
}
```

- CarreraServidor.java

  Aquí nos conectamos a la interfaz

```
protected CarreraServidor() throws RemoteException {
    super();
}
```

y tenemos los metodos registrarTortuga, en el cual hacemos el registro de cada cliente (Tortuga) y si ya esta registrada muestra que ya esta registrada, sino la registra

```
public synchronized String registrarTortuga(String nombre) throws RemoteException {
    if (!posiciones.containsKey(nombre)) {
        posiciones.put(nombre, value:0);
        System.out.println("Tortuga registrada: " + nombre);
        return "Tortuga " + nombre + " registrada correctamente";
    }
    return "La tortuga " + nombre + " ya esta registrada";
}
```

el de avanzar, en el cual se verifica que la tortuga esté registrada para participar en la carrera, al dar enter cada tortuga avanza a un posición dando pasos aleatorios entre uno y diez para llegar a cien y declararse como ganadora

```
public synchronized String avanzar(String nombre) throws RemoteException {
    if (ganador != null) return "La carrera ha terminado, el ganador es: " + ganador;

    if (!posiciones.containsKey(nombre)) return "Tortuga no registrada";

    int avance = (int) (Math.random() * 10 + 1);
    posiciones.put(nombre, posiciones.get(nombre) + avance);

    if (posiciones.get(nombre) >= META && ganador == null) {
        ganador = nombre;
        System.out.println("La tortuga " + nombre + " ha ganado la carrera!");
        return "¡Has llegado a la meta, " + nombre + "!";
    }

    return nombre + " avanzo a " + posiciones.get(nombre);
}
```

tambien tenemos al obtenerPosiciones, en el cual se hace un HashMap de las posiciones para saber como van las tortugas y sus posiciones, por último tenemos al obtenerGanador, el cual obtiene al ganador de la carrera de acuerdo a las posiciones de estos y el como van avanzando.

- TortugaCliente.java

este es mi favorito, ya que lo podemos ejecutar en distintas computadoras y poder entrar a la misma carrera al momento de que se inicia el RMI y el servidor, inicia pidiendote el nombre de la tortuga para iniciar su registro y te pide que vayas presionando ENTER para ir avanzando y en la parte de abajo se va imprimiendo la tabla de las tortugas conectadas con sus posiciones, de esta forma si alguna tortuga llega a la meta primero se va viendo en el registro quien podria ser el ganador, una vez que alguna llega a la meta finaliza la carrera y se muestra a la tortuga ganadora

```
CarreraInterfaz carrera = (CarreraInterfaz) Naming.lookup(name:"rmi://localhost/CarreraTc
Scanner sc = new Scanner(System.in);

System.out.print(s:"Nombre de la tortuga: "); // se registra la tortuga
String nombre = sc.nextLine();

System.out.println(carrera.registrarTortuga(nombre));

while (true) {
    System.out.println(x:"Presiona ENTER para avanzar..."); // al presionar ENTER se avar
    sc.nextLine();
    System.out.println(carrera.avanzar(nombre));

    String ganador = carrera.obtenerGanador(); // tablero de posiciones
    if (ganador != null) {
        System.out.println(" Carrera finalizada, Ganador: " + ganador);
        break;
    }

    Map<String, Integer> posiciones = carrera.obtenerPosiciones();
    System.out.println("Posiciones: " + posiciones);
}
```

Resultados

Una vez teniendo los archivos, los compilamos y ejecutamos primero el RMI

```
C:\Users\kirit\Downloads\ESCOM\DSD-R\Objetos-Distribuidos-RMI\CARRERA-RMI>rmiregistry
WARNING: A terminally deprecated method in java.lang.System has been called
WARNING: System::setSecurityManager has been called by sun.rmi.registry.RegistryImpl
WARNING: Please consider reporting this to the maintainers of sun.rmi.registry.RegistryImpl
WARNING: System::setSecurityManager will be removed in a future release
```

y después ejecutamos el servidor, el cual al iniciar espera a que se conecten las tortugas ya se de manera local (todas en mi laptop) o remota (tanto en mi laptop como en otras) y cuando alguna gana manda el anuncio

```
C:\Users\kirit\Downloads\ESCOM\DSD-R\Objetos-Distribuidos-RMI\CARRERA-RMI>java CarreraServidor
Servidor RMI listo y esperando tortugas...
Tortuga registrada: Momo
Tortuga registrada: Pekas
Tortuga registrada: Panzon
La tortuga Pekas ha ganado la carrera!
```

y luego lo divertido ocurre al momento de ejecutar cada tortuga en una terminal distinta, pasa que lo divertido es ver quien presiona ENTER más rápido para llegar primero a la meta, viéndose de la siguiente forma, conecte tres tortugas:

- Momo

```
C:\Users\kirit\Downloads\ESCOM\DSD-R\Objetos-Distribuidos-RMI\CARRERA-RMI>java TortugaCliente
Nombre de la tortuga: Momo
Tortuga Momo registrada correctamente
Presiona ENTER para avanzar...

Momo avanzo a 8
Posiciones: {Momo=8, Panzon=0, Pekas=0}
Presiona ENTER para avanzar...

Momo avanzo a 11
Posiciones: {Momo=11, Panzon=2, Pekas=6}
Presiona ENTER para avanzar...

Momo avanzo a 18
Posiciones: {Momo=18, Panzon=3, Pekas=9}
Presiona ENTER para avanzar...

Momo avanzo a 21
Posiciones: {Momo=21, Panzon=3, Pekas=9}
Presiona ENTER para avanzar...

Momo avanzo a 25
Posiciones: {Momo=25, Panzon=11, Pekas=15}
Presiona ENTER para avanzar...

Momo avanzo a 31
Posiciones: {Momo=31, Panzon=11, Pekas=15}
Presiona ENTER para avanzar...

Momo avanzo a 34
Posiciones: {Momo=34, Panzon=11, Pekas=15}
Presiona ENTER para avanzar...
```

```
Momo avanzo a 35
Posiciones: {Momo=35, Panzon=34, Pekas=34}
Presiona ENTER para avanzar...

Momo avanzo a 39
Posiciones: {Momo=39, Panzon=36, Pekas=42}
Presiona ENTER para avanzar...

Momo avanzo a 46
Posiciones: {Momo=46, Panzon=43, Pekas=48}
Presiona ENTER para avanzar...

Momo avanzo a 51
Posiciones: {Momo=51, Panzon=43, Pekas=48}
Presiona ENTER para avanzar...

Momo avanzo a 53
Posiciones: {Momo=53, Panzon=43, Pekas=48}
Presiona ENTER para avanzar...

Momo avanzo a 61
Posiciones: {Momo=61, Panzon=56, Pekas=68}
Presiona ENTER para avanzar...

Momo avanzo a 69
Posiciones: {Momo=69, Panzon=56, Pekas=68}
Presiona ENTER para avanzar...

Momo avanzo a 70
Posiciones: {Momo=70, Panzon=64, Pekas=80}
Presiona ENTER para avanzar...
```

```
Momo avanzo a 76
Posiciones: {Momo=76, Panzon=69, Pekas=86}
Presiona ENTER para avanzar...

Momo avanzo a 83
Posiciones: {Momo=83, Panzon=69, Pekas=86}
Presiona ENTER para avanzar...
```

```
Momo avanzo a 93
Posiciones: {Momo=93, Panzon=69, Pekas=86}
Presiona ENTER para avanzar...
```

```
Momo avanzo a 99
Posiciones: {Momo=99, Panzon=86, Pekas=96}
Posiciones: {Momo=99, Panzon=86, Pekas=96}
Presiona ENTER para avanzar...

La carrera ha terminado, el ganador es: Pekas
 Carrera finalizada, Ganador: Pekas
```

- Pekas

```
C:\Users\kirit\Downloads\ESCOM\DSD-R\Objetos-Distribuidos-RMI\CARRERA-RMI>java TortugaCliente
Nombre de la tortuga: Pekas
Tortuga Pekas registrada correctamente
Presiona ENTER para avanzar...

Pekas avanzo a 6
Posiciones: {Momo=8, Panzon=0, Pekas=6}
Presiona ENTER para avanzar...

Pekas avanzo a 9
Posiciones: {Momo=11, Panzon=2, Pekas=9}
Presiona ENTER para avanzar...

Pekas avanzo a 14
Posiciones: {Momo=21, Panzon=3, Pekas=14}
Presiona ENTER para avanzar...

Pekas avanzo a 15
Posiciones: {Momo=21, Panzon=3, Pekas=15}
Presiona ENTER para avanzar...

Pekas avanzo a 23
Posiciones: {Momo=34, Panzon=11, Pekas=23}
Presiona ENTER para avanzar...

Pekas avanzo a 32
Posiciones: {Momo=34, Panzon=11, Pekas=32}
Presiona ENTER para avanzar...

Pekas avanzo a 34
Posiciones: {Momo=34, Panzon=11, Pekas=34}
Presiona ENTER para avanzar...

Pekas avanzo a 42
Posiciones: {Momo=35, Panzon=34, Pekas=42}
Presiona ENTER para avanzar...
```

```
Pekas avanzo a 46
Posiciones: {Momo=39, Panzon=36, Pekas=46}
Presiona ENTER para avanzar...

Pekas avanzo a 48
Posiciones: {Momo=39, Panzon=40, Pekas=48}
Presiona ENTER para avanzar...

Pekas avanzo a 55
Posiciones: {Momo=53, Panzon=43, Pekas=55}
Presiona ENTER para avanzar...

Pekas avanzo a 61
Posiciones: {Momo=53, Panzon=43, Pekas=61}
Presiona ENTER para avanzar...

Pekas avanzo a 68
Posiciones: {Momo=53, Panzon=43, Pekas=68}
Presiona ENTER para avanzar...

Pekas avanzo a 70
Posiciones: {Momo=69, Panzon=56, Pekas=70}
Presiona ENTER para avanzar...

Pekas avanzo a 80
Posiciones: {Momo=69, Panzon=56, Pekas=80}
Presiona ENTER para avanzar...

Pekas avanzo a 86
Posiciones: {Momo=70, Panzon=69, Pekas=86}
Presiona ENTER para avanzar...

Pekas avanzo a 88
Posiciones: {Momo=93, Panzon=69, Pekas=88}
Presiona ENTER para avanzar...
```

```
Pekas avanzo a 95
Posiciones: {Momo=93, Panzon=69, Pekas=95}
Presiona ENTER para avanzar...
```

```
Pekas avanzo a 96
Posiciones: {Momo=93, Panzon=69, Pekas=96}
Posiciones: {Momo=93, Panzon=69, Pekas=96}
Presiona ENTER para avanzar...

¡Has llegado a la meta, Pekas!
 Carrera finalizada, Ganador: Pekas
```

- Panzon

```
C:\Users\kirit\Downloads\ESCOM\DSD-R\Objetos-Distribuidos-RMI\CARRERA-RMI>java TortugaCliente
Nombre de la tortuga: Panzon
Tortuga Panzon registrada correctamente
Presiona ENTER para avanzar...

Panzon avanzo a 2
Posiciones: {Momo=8, Panzon=2, Pekas=6}
Presiona ENTER para avanzar...

Panzon avanzo a 3
Posiciones: {Momo=11, Panzon=3, Pekas=9}
Presiona ENTER para avanzar...

Panzon avanzo a 10
Posiciones: {Momo=21, Panzon=10, Pekas=15}
Presiona ENTER para avanzar...

Panzon avanzo a 11
Posiciones: {Momo=21, Panzon=11, Pekas=15}
Presiona ENTER para avanzar...

Panzon avanzo a 17
Posiciones: {Momo=34, Panzon=17, Pekas=34}
Presiona ENTER para avanzar...

Panzon avanzo a 27
Posiciones: {Momo=34, Panzon=27, Pekas=34}
Presiona ENTER para avanzar...

Panzon avanzo a 34
Posiciones: {Momo=34, Panzon=34, Pekas=34}
Presiona ENTER para avanzar...

Panzon avanzo a 36
Posiciones: {Momo=35, Panzon=36, Pekas=42}
Presiona ENTER para avanzar...
```

```
Panzon avanzo a 40
Posiciones: {Momo=39, Panzon=40, Pekas=46}
Presiona ENTER para avanzar...

Panzon avanzo a 43
Posiciones: {Momo=39, Panzon=43, Pekas=48}
Presiona ENTER para avanzar...

Panzon avanzo a 46
Posiciones: {Momo=53, Panzon=46, Pekas=68}
Presiona ENTER para avanzar...

Panzon avanzo a 51
Posiciones: {Momo=53, Panzon=51, Pekas=68}
Presiona ENTER para avanzar...

Panzon avanzo a 56
Posiciones: {Momo=53, Panzon=56, Pekas=68}
Presiona ENTER para avanzar...

Panzon avanzo a 59
Posiciones: {Momo=69, Panzon=59, Pekas=80}
Presiona ENTER para avanzar...

Panzon avanzo a 64
Posiciones: {Momo=69, Panzon=64, Pekas=80}
Presiona ENTER para avanzar...

Panzon avanzo a 69
Posiciones: {Momo=70, Panzon=69, Pekas=80}
Presiona ENTER para avanzar...

Panzon avanzo a 70
Posiciones: {Momo=93, Panzon=70, Pekas=96}
Presiona ENTER para avanzar...
```

```
Panzon avanzo a 74
Posiciones: {Momo=93, Panzon=74, Pekas=96}
Presiona ENTER para avanzar...

Panzon avanzo a 75
Posiciones: {Momo=93, Panzon=75, Pekas=96}
Presiona ENTER para avanzar...
```

```
Panzon avanzo a 82
Posiciones: {Momo=93, Panzon=82, Pekas=96}
Presiona ENTER para avanzar...
```

```
Panzon avanzo a 86
Posiciones: {Momo=93, Panzon=86, Pekas=96}
Posiciones: {Momo=93, Panzon=86, Pekas=96}
Presiona ENTER para avanzar...

La carrera ha terminado, el ganador es: Pekas
 Carrera finalizada, Ganador: Pekas
```

Conclusiones

Gracias a esta nueva práctica he comprendido que un servidor RMI puede manejar diferentes clientes (o en este caso varias tortugas), y de esta forma los clientes RMI (las tortugas) se conectan y avanzan en la carrera, también que un sistema distribuido es aquel en donde clientes y servidores pueden ejecutarse en diferentes instancias tanto como en diferentes computadoras, pero por ahora pude observar cómo se conectaban varias tortugas como he estado trabajando y analizando en prácticas anteriores pero ahora de distinta forma, utilizando nuevos métodos y distintas funciones para lograrlo, lo que me resultó un poco más fácil de comprender fue que al utilizar sockets la comunicación por texto era linea por linea mientras que con RMI es comunicación por objetos, y la serialización y parseo se hacían manualmente mientras que ahora Java lo hace de manera automática, la comunicación en vez de ser punto a punto ahora se vuelve orientada a objetos y lo que mas me gusto es que era difícil tratar de escalar mis prácticas a algún servicio, pero RMI permite múltiples objetos remotos.

Lo más curioso realizando esta práctica, es que, al principio consideré algo tedioso tener que presionar ENTER cada vez para avanzar (lo vi tedioso porque lo hice yo varias veces en distintas terminales manualmente), pero el concepto clave es que el RMI convierte los objetos en objetos distribuidos, por lo que un objeto puede vivir en una computadora (el servidor), pero a su vez puede ser invocado desde otra computadora (cliente Tortuga) como si estuviese en la misma computadora, por lo que comprendí que Java se encarga de serializar los datos, abrir los sockets, enviar peticiones y devolver los resultados de manera automática. Esto se logra si todas la tortugas (clientes) están conectadas a la misma red, y solamente cambiamos la línea de Registry registry = LocaleRegistry.getRegistry(“localhost”); por Registry registry = LocateRegistry.getRegistry(“192.168.0.10); qué es la dirección IP de mi laptop y desde cada laptop ejecutamos TortugaCliente.java y entonces ocurre la magia, ya que, todas se conectan al mismo servidor RMI central y se puede hacer una carrera de tortugas con más personas (justo como funcionan los videojuegos).

Enlace a la práctica en GitHub

https://github.com/Marlene0807/Sistemas-Distribuidos.git

Referencias

OBJETOS DISTRIBUIDOS - SISTEMAS DISTRIBUIDOS

Java RMI

Comunicación entre Procesos

Objetos-Distribuidos.pdf

RMIv2019