PRACTICA

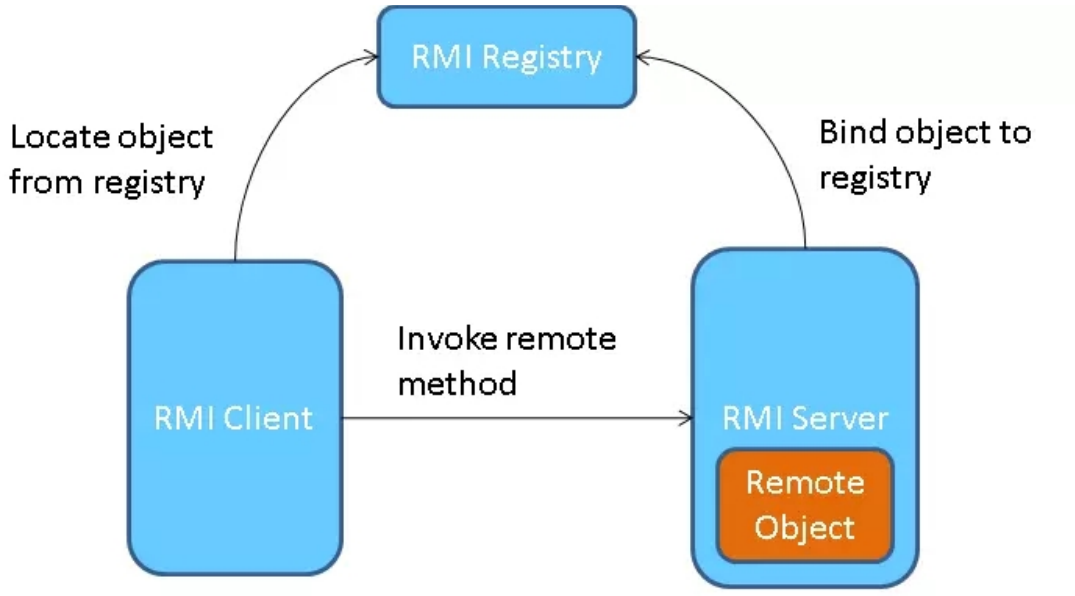
21/11/2024

“Voluntaria: Tarea de Investigación
y Prueba sobre RMI”

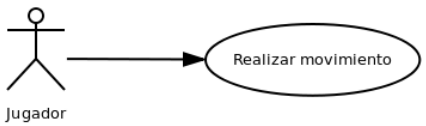
Ahmed Hassan Khamis

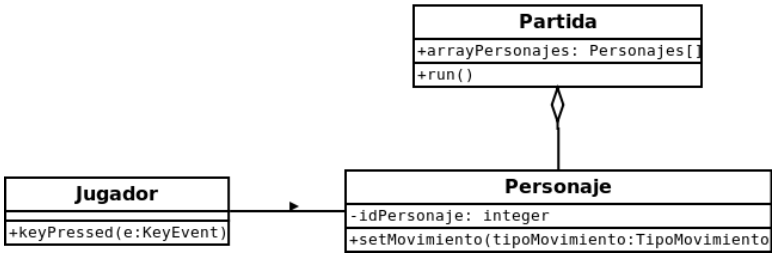
Remote Method Invocation (RMI)

Remote Method Invocation (RMI) es una tecnología de Java que permite a los objetos en una máquina virtual Java (JVM) invocar métodos en objetos ubicados en otra JVM. Esta capacidad es fundamental para la construcción de aplicaciones distribuidas, donde los componentes de la aplicación pueden estar dispersos en diferentes ubicaciones geográficas pero necesitan interactuar como si estuvieran en la misma máquina.



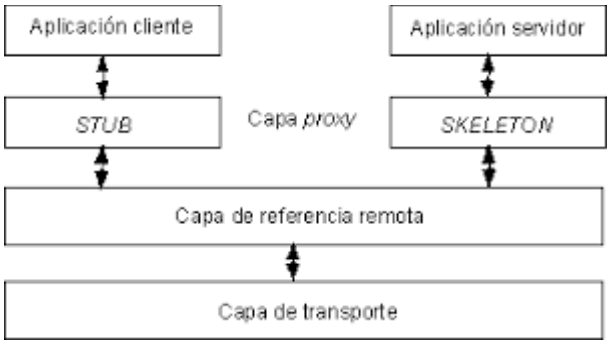
RMI facilita la comunicación entre objetos distribuidos al proporcionar una abstracción de alto nivel que oculta los detalles de la red y la serialización de objetos. Los desarrolladores pueden invocar métodos en objetos remotos de la misma manera que lo harían con objetos locales, lo que simplifica el desarrollo de aplicaciones distribuidas.





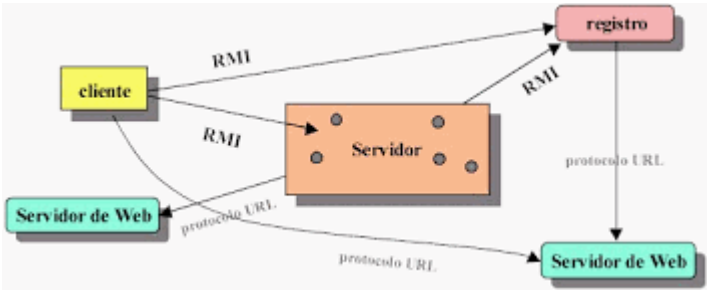
Componentes de RMI:

1. **Interfaz Remota**: Define los métodos que pueden ser invocados remotamente. Esta interfaz debe extender "java.rmi.Remote".
2. **Implementación Remota**: La clase que implementa la interfaz remota. Debe extender "java.rmi.server.UnicastRemoteObject" y manejar las excepciones "RemoteException".
3. **Cliente**: El programa que invoca métodos en el objeto remoto. Utiliza el servicio de nombres RMI para localizar el objeto remoto.
4. **Registro RMI**: Un servicio de nombres que permite a los clientes localizar objetos remotos por nombre.



Proceso de RMI:

1. **Definición de la Interfaz Remota**: Se define una interfaz que extiende "java.rmi.Remote" y declara los métodos que pueden ser invocados remotamente.
2. **Implementación del Servidor**: Se implementa la interfaz remota y se extiende "UnicastRemoteObject". El servidor registra el objeto remoto con el registro RMI.
3. **Implementación del Cliente**: El cliente busca el objeto remoto en el registro RMI y llama a los métodos remotos como si fueran locales.



Ejemplo de RMI:

NOTA: el siguiente ejemplo esta traído a partir de un pequeño esquema en java, luego pasado por una ia para darle lo necesario y restante.

Un ejemplo simple de **RMI** en el sector de videojuegos podría ser un sistema que permita a los jugadores consultar la puntuación de otro jugador o guardar sus puntuaciones en un servidor central. Este tipo de funcionalidad es común en sistemas de clasificación global o "leaderboards".

Ejemplo: Sistema de consulta y registro de puntuaciones

1. Interfaz remota

Define los métodos que estarán disponibles de forma remota, como consultar y actualizar puntuaciones.

```
import java.rmi.Remote;
import java.rmi.RemoteException;
import java.util.Map;

public interface ScoreService extends Remote {
    // Método para obtener las puntuaciones
    Map<String, Integer> getScores() throws RemoteException;

    // Método para actualizar la puntuación de un jugador
    void updateScore(String player, int score) throws RemoteException;
}
```

---

2. Implementación del servicio

Esta clase implementa la lógica del servidor para manejar las puntuaciones.

```
import java.rmi.server.UnicastRemoteObject;
import java.rmi.RemoteException;
import java.util.HashMap;
import java.util.Map;

public class ScoreServiceImpl extends UnicastRemoteObject implements
ScoreService {
    private Map<String, Integer> scores;

    // Constructor
    protected ScoreServiceImpl() throws RemoteException {
        super();
        scores = new HashMap<>();
    }

    @Override
    public Map<String, Integer> getScores() throws RemoteException {
        return scores;
    }

    @Override
    public void updateScore(String player, int score) throws RemoteException {
        scores.put(player, score);
        System.out.println("Puntuación actualizada: " + player + " -> " +
score);
    }
}
```

---

3. Servidor

El servidor registra el objeto remoto en el registro RMI.

```
import java.rmi.registry.LocateRegistry;
import java.rmi.registry.Registry;

public class GameServer {
    public static void main(String[] args) {
        try {
            // Crear la implementación del servicio
            ScoreService scoreService = new ScoreServiceImpl();

            // Registrar el objeto remoto
            Registry registry = LocateRegistry.createRegistry(1099);
            registry.rebind("ScoreService", scoreService);

            System.out.println("Servidor de puntuaciones listo.");
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

---

4. Cliente

El cliente puede consultar y actualizar puntuaciones remotas.

```
import java.rmi.registry.LocateRegistry;
import java.rmi.registry.Registry;
import java.util.Map;

public class GameClient {
    public static void main(String[] args) {
        try {
            // Conectar al registro RMI
            Registry registry = LocateRegistry.getRegistry("localhost", 1099);

            // Buscar el servicio remoto
            ScoreService scoreService = (ScoreService)
registry.lookup("ScoreService");

            // Actualizar puntuaciones
            scoreService.updateScore("Jugador1", 1500);
            scoreService.updateScore("Jugador2", 2000);

            // Consultar puntuaciones
            Map<String, Integer> scores = scoreService.getScores();
            System.out.println("Puntuaciones actuales:");
            scores.forEach((player, score) ->
                System.out.println(player + ": " + score)
            );
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

Pasos para ejecutar

1. **Compilar todos los archivos:**

   ```
   javac *.java
   ```

2. **Iniciar el registro RMI:**

   ```
   rmiregistry
   ```

3. **Ejecutar el servidor:**

   ```
   java GameServer
   ```

4. **Ejecutar el cliente:**

   ```
   java GameClient
   ```

Ejemplo de salida

En el servidor:

```
Servidor de puntuaciones listo.
Puntuación actualizada: Jugador1 -> 1500
Puntuación actualizada: Jugador2 -> 2000
```

En el cliente:

```
Puntuaciones actuales:
Jugador1: 1500
Jugador2: 2000
```

Aplicaciones prácticas

1. **Leaderboards globales:** Mostrar clasificaciones de jugadores en tiempo real.
2. **Sincronización multijugador:** Compartir estadísticas o datos entre jugadores.
3. **Sistemas de logros:** Almacenar y consultar logros en un servidor centralizado.