Instituto Politécnico Nacional

Escuela Superior de Cómputo


# Programación Orientada a Objetos

# Reporte de Práctica #8

# RMI

Profesor: Roberto Tecla Parra

Alumno: Calva Hernández José Manuel

Grupo: 2CM3

**ChatBot**

Codificar un programa que actué como un ChatBot con al menos 10 preguntas y respuestas predefinidas. El servidor debe tener un método remoto que reciba la pregunta como parámetro y que dicho método tenga como valor de retorno la respuesta.

Ejemplos de preguntas tipo y respuestas tipo:

En que ciudad vives? D.F

Cuantos años tienes? 20

En que escuela estudias? ESCOM

Cuantas materias llevas? 6

**Objetivos**

- ➢ Aprender el uso de RMI en Java.
- ➢ Comprender el protocolo de comunicación entre los RMI.
- ➢ Implementar exitosamente un RMI por medio de la codificación en tres partes del mismo.

**Desarrollo**

La primera parte de la construcción será la de codificar una interfaz remota de la siguiente forma:

```
import java.rmi.*;

public interface ChatBotI extends Remote {

    public String responde(String pregunta) throws RemoteException;

}
```

En donde la interfaz hará extensión hacia Remote para ejemplificar la aplicación del Método Remoto. Posteriormente codificaremos su implementación valiéndonos de las siguientes librerías:

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

A continuación crearemos nuestra clase ChatBotImp que implementará la interfaz antes creada por medio de la palabra reservada implements, por tanto, tendrá que

implementar el método remoto llamado responde, para lo cual se declarará un HashMap al que se asociará tanto preguntas como respuestas para la posterior interacción entre el cliente y el servidor.

```
public class ChatBotImp extends UnicastRemoteObject implements ChatBotI {

    public ChatBotImp() throws RemoteException {
        super();
    }

    public String responde(String pregunta) throws RemoteException {

        String respuesta = "LO SIENTO, NO TE ENTIENDO";

        HashMap<String,String> map = new HashMap<>();
        map.put("HOLA" , "HOLA, MUCHO GUSTO");
        map.put("ADIOS" , "ADIOS, FUE UN PLACER");
        map.put("COMO TE LLAMAS?" , "ME LLAMO MANUEL");
        map.put("CUANTOS AÑOS TIENES?" , "22");
        map.put("DONDE ESTUDIAS?" , "EN ESCOM");
        map.put("QUE CLASE ES ESTA?" , "PROGRAMACION ORIENTADA A OBJETOS");
        map.put("QUE LENGUAJE UTILIZAS?" , "JAVA");
        map.put("COLOR FAVORITO?" , "ROJO");
        map.put("QUE MUSICA ESCUCHAS?" , "ROCK");
        map.put("A QUE HORA SALES?" , "3 DE LA TARDE");
        map.put("DONDE NACISTE?" , "CIUDAD DE MEXICO");
        map.put("COMIDA FAVORITA?" , "MOLE CON POLLO");

        if(map.containsKey(pregunta.toUpperCase())){
            respuesta = map.get(pregunta.toUpperCase());
        }

        return respuesta;
    }
}
```

Procederemos a codificar el Servidor valiéndonos de nuestra interfaz remota creada anteriormente, para ello lo primero que haremos será conectarnos al registro de interfaces remotas local, por medio del url local host y el puerto 1025.

```java
public class Servidor {

    private String ip;
    private int puerto;

    public Servidor() {
        try {
            ChatBotImp remote = new ChatBotImp();
            ip = "localhost";
            puerto = 1025;
            java.rmi.registry.LocateRegistry.createRegistry(puerto);
            Naming.rebind("//" + ip + ":" + puerto + "/ChatBot", remote);
        } catch (Exception ex) {
            System.out.println("ERROR " + ex);
            System.exit(0);
        }
    }

    public static void main(String[] s) {
        new Servidor();
    }
}
```

Una vez conectados, el Cliente podrá interactuar con la interfaz remota por medio del Servidor. Así que procederemos a crear la clase Cliente, para lo cual primeramente declararemos un objeto de la interfaz remota, así como un BufferedReader para poder leer las preguntas que realice el usuario.

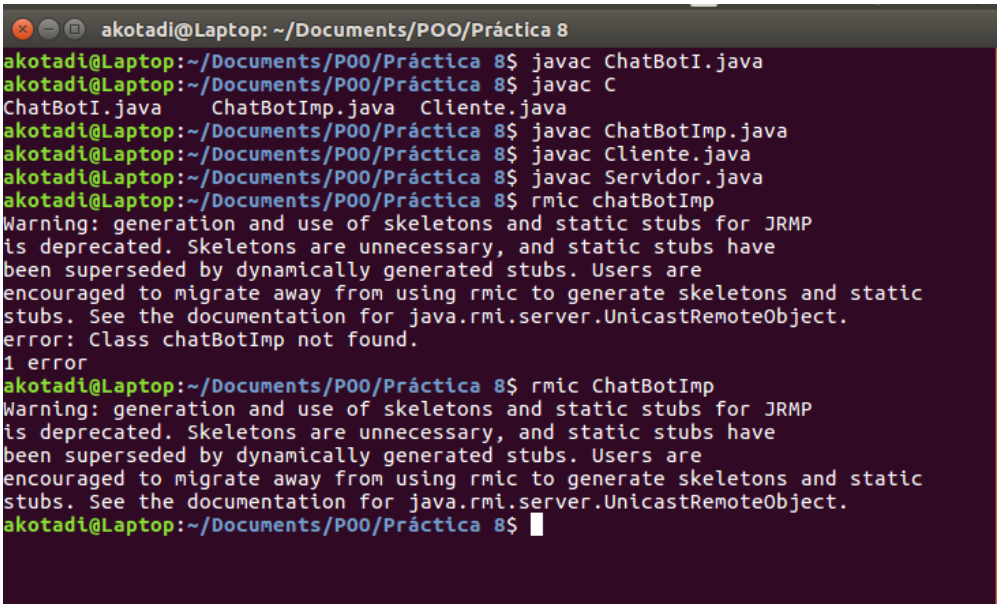
```java
public class Cliente {

    private ChatBotI remote;
    private BufferedReader inLine = new BufferedReader(new InputStreamReader(System.in));
    private String ip, respuesta;
    private int puerto;
```

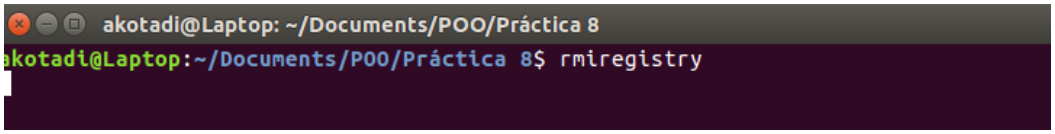
A continuación, crearemos el constructor del Cliente, el cual se conectará a la interfaz remota previamente registrada en la misma url y puerto que declararemos para el Cliente, una vez conseguidos, procederemos a iniciar la interacción entre el usuario y la interfaz por medio del método remoto que contenía la misma.

```java
public Cliente() {
    try {
        ip = "localhost";
        puerto = 1025;
        remote = (ChatBotI)Naming.lookup("//" + ip + ":" + puerto + "/ChatBot");
    } catch (Exception ex) {
        System.out.println("ERROR " + ex);
    }
    try {
        while (true) {
            System.out.print("\nEscribe tu pregunta: ");
            respuesta = remote.responde(inLine.readLine());
            System.out.println("\nRespuesta: " + respuesta);
        }
    } catch (Exception ex) {
        System.out.println("ERROR " + ex);
    }
}
```

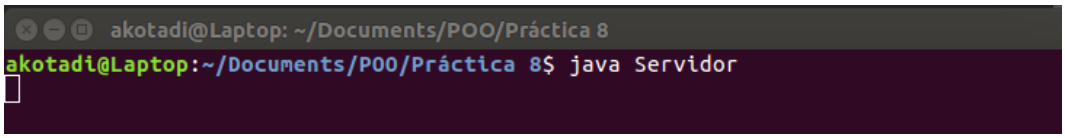
Para ejecutar el código, primeramente deberemos compilar todos los archivos de forma ordinaria con javac, sin embargo, al archivo que implementa la interfaz remota, además vamos a compilarlo con el comando rmic.




Procederemos a ejecutar rmiregistry que nos permitirá la comunicación entre el cliente y el servidor.




Continuaremos con el servidor.




Ambos se quedan en stanby, a la espera de que el cliente acceda e interactue con ellos, por tanto, avanzaremos y ejecutaremos el cliente.


## Conclusiones

Los RMI son herramientas importantes para los programadores de Java debido a que nos permiten una interacción distinta con el cliente, ya que permiten la interacción del mismo con métodos remotos en servidores distribuidos.

Su facilidad de uso es una de las ventajas que tiene el mismo ya que está pensado concretamente para Java. Por medio del mismo se puede exportar objetos que estará disponible en la red para peticiones de otros clientes que lo soliciten por medio de peticiones concretas.

Por ello, RMI representa una de las facilidades que nos da Java como programadores para su aplicación a posteriori.