INSTITUTO POLITÉCNICO NACIONAL

ESCUELA SUPERIOR DE COMPÚTO

PROGRAMACIÓN ORIENTADA A OBJETOS

PRÁCTICA 8: CHATBOT

PROF. ROBERTO TECLA PARRA

GRUPO 2CM4

ZEPEDA FLORES ALEJANDRO DE JESÚS

15/06/3018

**Objetivo:**

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.

**Implementación:**

El primer paso de RMI primero debemos crear una interfaz remota.

public interface ChatBot extends Remote{

String chatear(String str) throws RemoteException;

}

Después debemos implementar el servidor que implementa IR. En esta clase declaramos un arreglo de preguntas y un arreglo de respuestas, cada una debe corresponder con la otra en cuanto a posición.

Después, verificamos la petición del cliente y la buscamos en el arreglo de preguntas, en caso de no encontrarse se manda un mensaje de error.

public class ChatBotImpl extends UnicastRemoteObject implements ChatBot{

String[] preguntas = {"Cual es tu nombre?","Cual es tu edad?","Que materia es esta?","Lenguaje de programacion favorito", "Que idiomas hablas?","Cual es tu comida favorita?","Cual es la mejor pelicula de la historia?","Cual es tu auto favorito?","Quien es el mejor jugador del mundo?","Quien quieres que sea tu novia?"};

String[] respuestas={"Alejandro Zepeda","21 years", "Programacion Orientada","Java","Ingles y Frances","Lasagna", "Avengers Infinity War","Mazda C3","Lionel Messi", "Monserrat"};

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

public String chatear (String str) throws RemoteException{

String e = "No tengo esa respuesta en mi repertorio";

String result = "";

for(int i = 0; i < preguntas.length; ++i)

if(preguntas[i].equalsIgnoreCase(str)) {

result = respuestas[i];

break;

}

else{result = e;}

return result;

public static void main(String s[]){

try{

ChatBotImpl c = new ChatBotImpl();

Naming.rebind("Calcular",c);

}catch(RemoteException re){

System.out.println("Exception in ChatBotImpl.main: " + re);}

catch(MalformedURLException e) {

System.out.println("MalformedURLException en ChatBotImpl.main: " + e);

}

}

Después, hay que implementar el cliente. Aquí hay que definir el constructor para dar los parámetros de la ventana de dialogo y después tenemos que programar el actionPerformed, en esa sección del código se realizará la petición.

public class ChatBotClient extends JFrame implements ActionListener {

static JLabel respuesta;

JButton button;

JTextField pregunta;

static String string = "";

public ChatBotClient(){

pregunta = new JTextField(20);

respuesta = new JLabel();

button = new JButton("Preguntar");

button.addActionListener(this);

}

public void actionPerformed (ActionEvent e){

try{

ChatBot c = (ChatBot)Naming.lookup("Calcular");

string = pregunta.getText();

String mensaje = c.chatear(string);

respuesta.setText(mensaje);

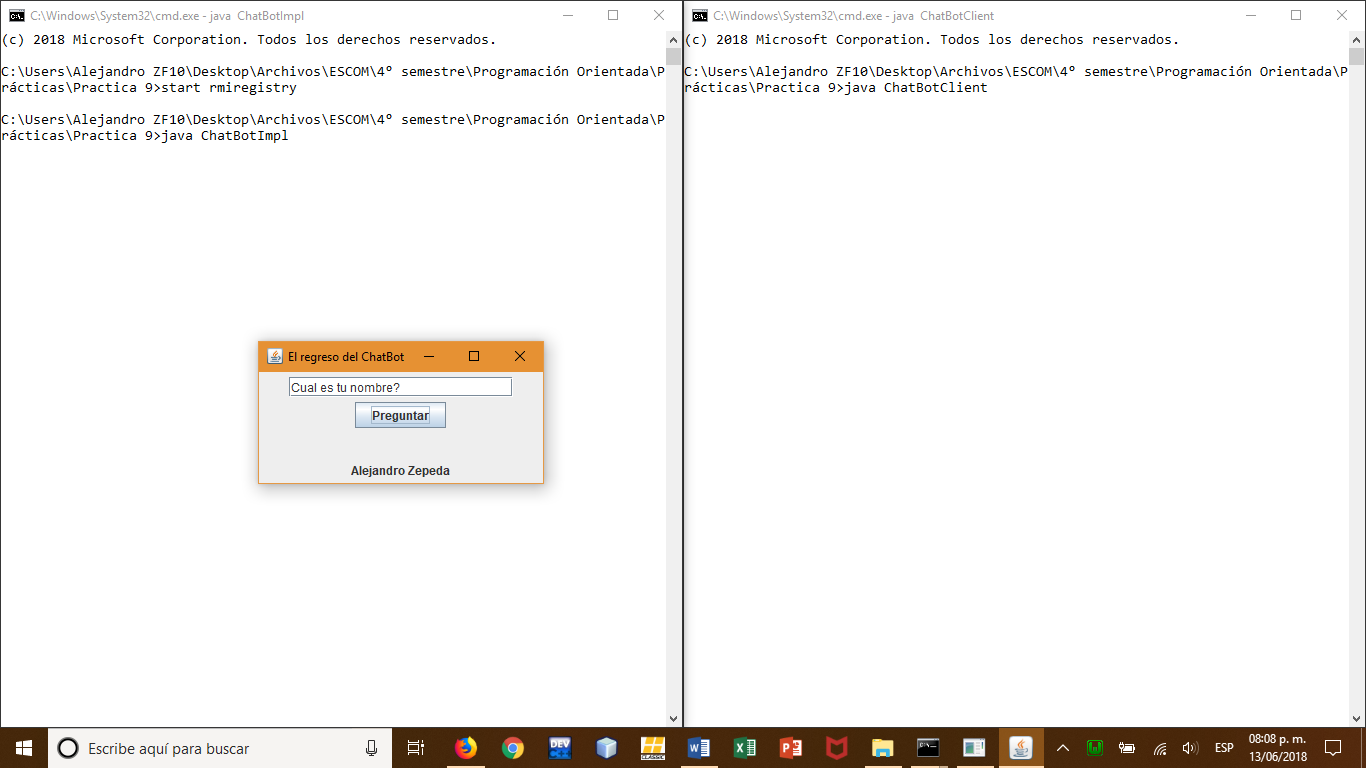
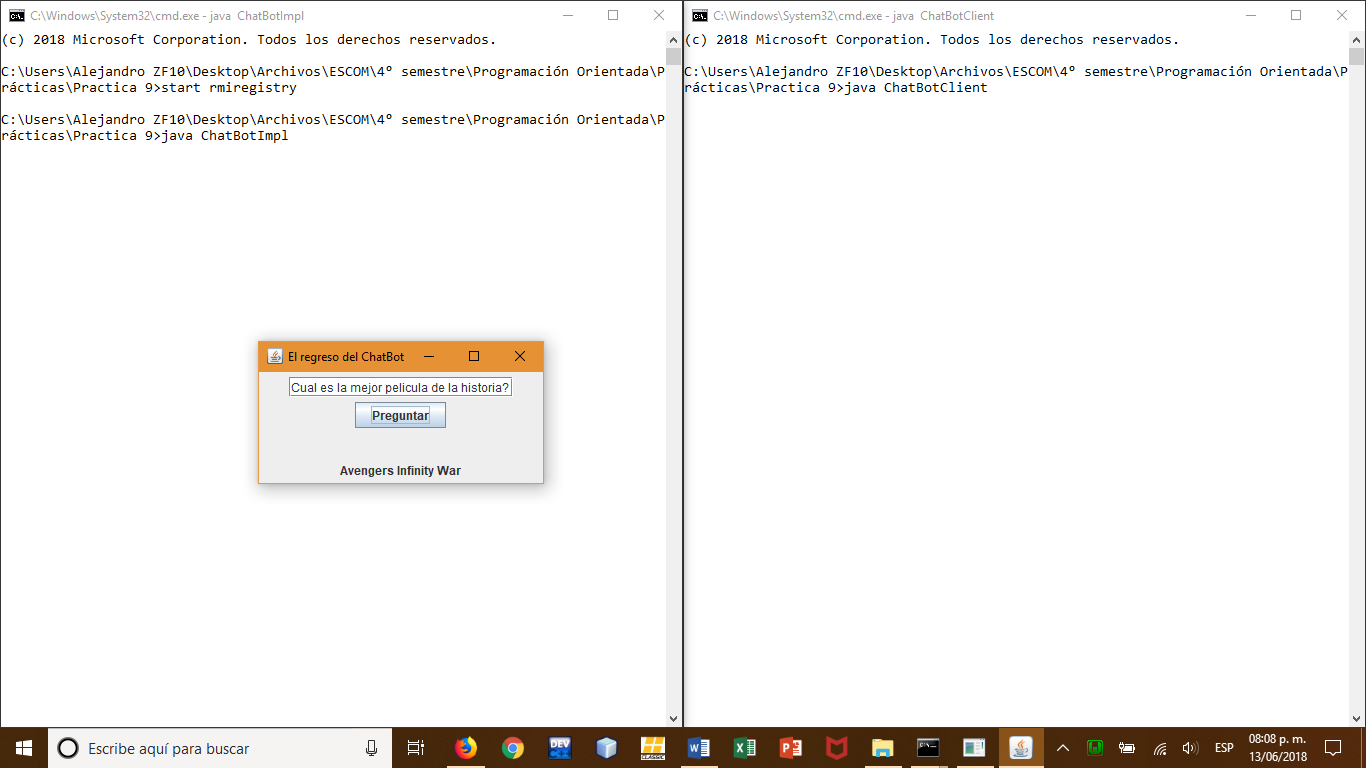
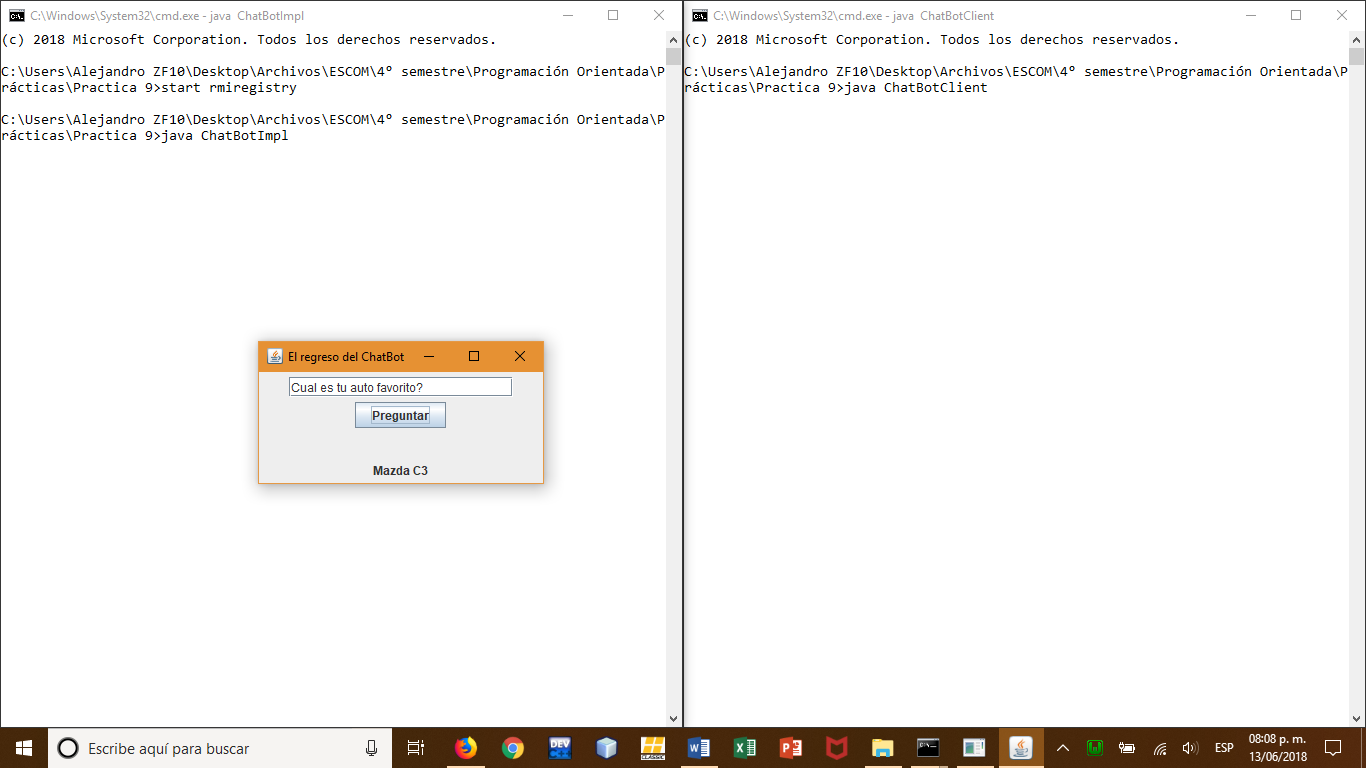
//System.out.println(mensaje);

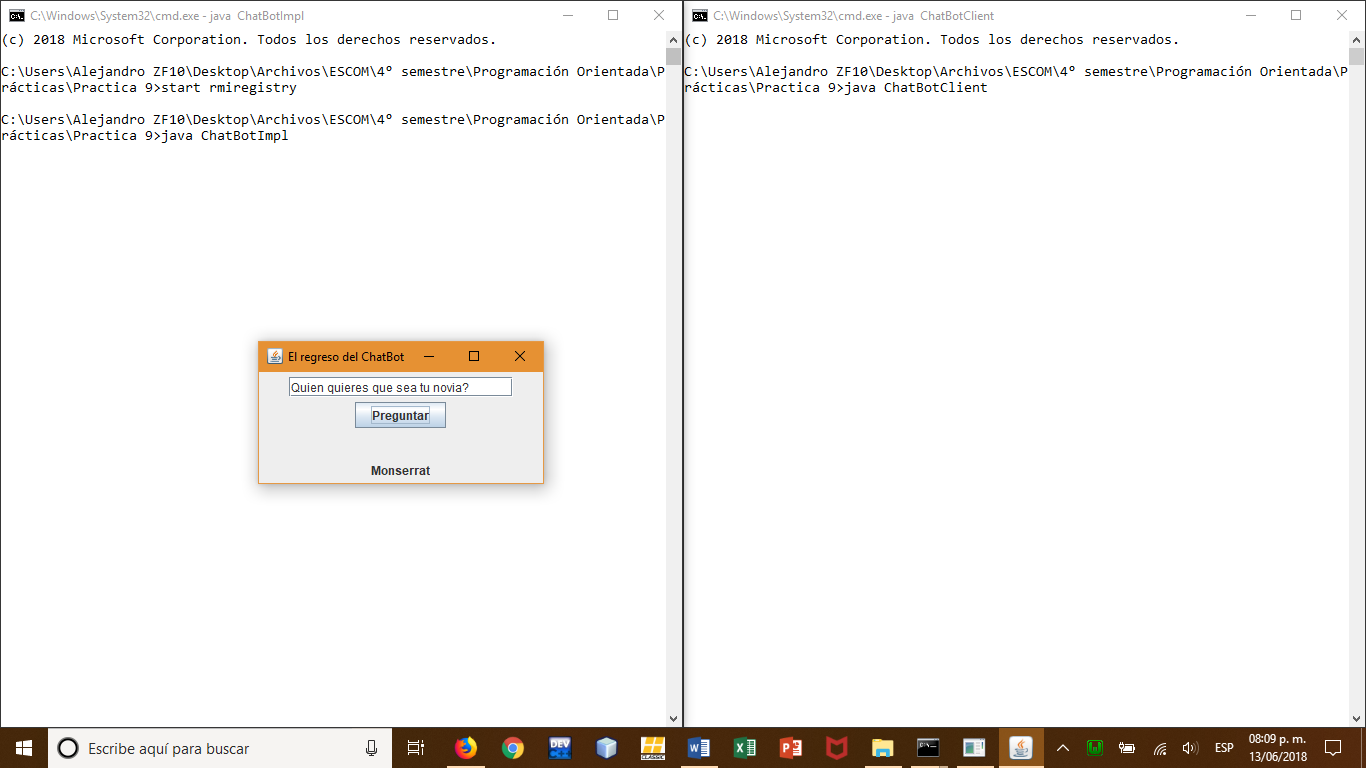
}catch(Exception ex){

System.out.println("falla pregunta");

}

}

****

**CONCLUSIÓN**

RMI (Remote Method Invocation) sirve para invocar un método de forma remota. En un principio, la práctica se complicó por el manejo de RMI, pero siguiendo los 3 pasos básicos:

1. Implementar la interfaz remota
2. Crear una clase que implemente la interfaz remota
3. Implementar al cliente

Además, para tener un mayor manejo hay que lanzar demasiadas excepciones y así poder detectar donde se encuentra el error fácilmente.