**Actividad2**

**Obligatoria. Servidor RMI de constelaciones**

**Iván Núñez Rodríguez**

**Iván Serrano Núñez**

2º DAM

Programación de Procesos y Servicios

**Índice**

1. Código.
2. Resultado.
3. Enlace de Github

**Código**

**CLIENTE**

**ClinetRMI**

import java.rmi.registry.LocateRegistry;  
import java.rmi.registry.Registry;  
import java.util.Scanner;  
  
// Cliente RMI  
class ClienteRMI {  
 public static void main(String[] args) {  
 try {  
 Registry registro = LocateRegistry.*getRegistry*("localhost", 2021);  
 ConstelacionInterfaceRMI constelaciones = (ConstelacionInterfaceRMI) registro.lookup("Constelaciones");  
 PlanetaInterfaceRMI planetas = (PlanetaInterfaceRMI) registro.lookup("Planetas");  
 Scanner scanner = new Scanner(System.*in*);  
  
 while (true) {  
 System.*out*.println("\nSeleccione una opción:");  
 System.*out*.println("1. Buscar Constelación");  
 System.*out*.println("2. Buscar Planeta");  
 System.*out*.println("3. Salir");  
 System.*out*.print("Opción: ");  
 int opcion = scanner.nextInt();  
 scanner.nextLine();  
  
 if (opcion == 3) {  
 System.*out*.println("Sesión finalizada.");  
 break;  
 } else if (opcion == 1) {  
 System.*out*.print("Nombre de la constelación: ");  
 String nombreConst = scanner.nextLine().trim().toUpperCase();  
 System.*out*.println(constelaciones.buscarConstelacion(nombreConst));  
 } else if (opcion == 2) {  
 System.*out*.println("\nSeleccione una consulta sobre el planeta:");  
 System.*out*.println("1. Descripción");  
 System.*out*.println("2. Tamaño");  
 System.*out*.println("3. Temperatura");  
 System.*out*.print("Opción: ");  
 int subOpcion = scanner.nextInt();  
 scanner.nextLine();  
  
 System.*out*.print("Nombre del planeta: ");  
 String nombrePlaneta = scanner.nextLine().trim().toUpperCase();  
  
 switch (subOpcion) {  
 case 1:  
 System.*out*.println(planetas.obtenerDescripcion(nombrePlaneta));  
 break;  
 case 2:  
 System.*out*.println(planetas.obtenerTamano(nombrePlaneta));  
 break;  
 case 3:  
 System.*out*.println(planetas.obtenerTemperatura(nombrePlaneta));  
 break;  
 default:  
 System.*out*.println("Opción inválida.");  
 }  
 } else {  
 System.*out*.println("Opción inválida.");  
 }  
 }  
 scanner.close();  
 } catch (Exception e) {  
 e.printStackTrace();  
 }  
 }  
}

**ConstelacionRMI**

import java.rmi.Remote;  
import java.rmi.RemoteException;  
  
public interface ConstelacionInterfaceRMI extends Remote {  
 String buscarConstelacion(String nombre) throws RemoteException;  
}

**PlanetaInterfaceRMI**

import java.rmi.Remote;  
import java.rmi.RemoteException;  
  
public interface PlanetaInterfaceRMI extends Remote {  
 String obtenerDescripcion(String nombre) throws RemoteException;  
 String obtenerTamano(String nombre) throws RemoteException;  
 String obtenerTemperatura(String nombre) throws RemoteException;  
}

**SERVIDOR**

**Constelacion**

import lombok.AllArgsConstructor;  
import lombok.Data;  
  
import java.io.Serial;  
import java.io.Serializable;  
  
@Data  
@AllArgsConstructor  
class Constelacion implements Serializable {  
 @Serial  
 private static final long *serialVersionUID* = -4540135499251166707L;  
 private String nombre;  
 private String observaciones;  
}

**ConstelacionInterfaceRMI**

import java.rmi.Remote;  
import java.rmi.RemoteException;  
  
public interface ConstelacionInterfaceRMI extends Remote {  
 String buscarConstelacion(String nombre) throws RemoteException;  
}

**ConstelacionRMI**

import java.io.Serial;  
import java.rmi.RemoteException;  
import java.rmi.server.UnicastRemoteObject;  
import java.util.ArrayList;  
import java.util.List;  
  
class ConstelacionRMI extends UnicastRemoteObject implements ConstelacionInterfaceRMI {  
 @Serial  
 private static final long *serialVersionUID* = -9029108738150223029L;  
 private List<Constelacion> constelaciones;  
  
 protected ConstelacionRMI() throws RemoteException {  
 constelaciones = new ArrayList<>();  
 constelaciones.add(new Constelacion("Osa Mayor", "Se desplaza en círculos alrededor del polo norte."));  
  
 constelaciones.add(new Constelacion("Osa Menor", "Su estrella más conocida es la polar que se encuentra en la prolongación del eje de la tierra."));  
  
 constelaciones.add(new Constelacion("Tauro", "Una de las constelaciones más conocidas desde tiempos remotos."));  
  
 constelaciones.add(new Constelacion("Leo", "De las más brillantes del Zodíaco."));  
  
 constelaciones.add(new Constelacion("Escorpio", "Sus estrellas forman un escorpión."));  
  
 constelaciones.add(new Constelacion("Can Mayor", "Contiene la estrella Sirio, la más brillante en el cielo nocturno."));  
  
 constelaciones.add(new Constelacion("Casiopea", "Tiene forma de M o W. Es conocida desde mucha antigüedad."));  
  
 constelaciones.add(new Constelacion("El Boyero", "Contiene la estrella Arturo, uno de las más luminosas del cielo."));  
  
 constelaciones.add(new Constelacion("Cruz del sur", "Señala al polo sur. Constelación muy pequeña."));  
  
 constelaciones.add(new Constelacion("Acuario", "Una de las más antiguas. Incluye 56 estrellas."));  
  
 constelaciones.add(new Constelacion("Géminis", "Destaca por sus dos gemelos, las estrellas Cástor y Pólux."));  
 }  
  
 @Override  
 public String buscarConstelacion(String nombre) throws RemoteException {  
 return constelaciones.stream()  
 .filter(c -> c.getNombre().equalsIgnoreCase(nombre))  
 .map(Constelacion::toString)  
 .findFirst()  
 .orElse("Constelación no encontrada");  
 }  
}

**Planeta**

import lombok.AllArgsConstructor;  
import lombok.Data;  
  
@Data  
@AllArgsConstructor  
class Planeta {  
 private final String nombre;  
 private final String descripcion;  
 private final String tamano;  
 private final String temperatura;  
  
  
}

**PlanetaInterfaceRMI**

import java.rmi.Remote;  
import java.rmi.RemoteException;  
  
public interface PlanetaInterfaceRMI extends Remote {  
 String obtenerDescripcion(String nombre) throws RemoteException;  
 String obtenerTamano(String nombre) throws RemoteException;  
 String obtenerTemperatura(String nombre) throws RemoteException;  
}

**PlanetaRMI**

import java.io.Serial;  
import java.rmi.RemoteException;  
import java.rmi.server.UnicastRemoteObject;  
import java.util.HashMap;  
import java.util.Map;  
  
class PlanetaRMI extends UnicastRemoteObject implements PlanetaInterfaceRMI {  
 @Serial  
 private static final long *serialVersionUID* = 1L;  
 private final Map<String, Planeta> planetas;  
  
 protected PlanetaRMI() throws RemoteException {  
 planetas = new HashMap<>();  
 planetas.put("MERCURIO", new Planeta("Mercurio", "El planeta más cercano al Sol.", "4,879 km de diámetro", "430°C de día, -180°C de noche"));  
 planetas.put("VENUS", new Planeta("Venus", "Conocido por su atmósfera densa y temperaturas extremas.", "12,104 km de diámetro", "471°C"));  
 planetas.put("TIERRA", new Planeta("Tierra", "Nuestro hogar, el único planeta conocido con vida.", "12,742 km de diámetro", "-88°C a 58°C"));  
 planetas.put("MARTE", new Planeta("Marte", "El planeta rojo, con la montaña más alta del sistema solar.", "6,779 km de diámetro", "-87°C a -5°C"));  
 planetas.put("JUPITER", new Planeta("Júpiter", "El gigante gaseoso con la Gran Mancha Roja.", "139,820 km de diámetro", "-145°C"));  
 planetas.put("SATURNO", new Planeta("Saturno", "Famoso por sus anillos impresionantes.", "116,460 km de diámetro", "-178°C"));  
 planetas.put("URANO", new Planeta("Urano", "Un gigante helado con rotación inclinada.", "50,724 km de diámetro", "-224°C"));  
 planetas.put("NEPTUNO", new Planeta("Neptuno", "El planeta más lejano del sistema solar.", "49,244 km de diámetro", "-218°C"));  
 }  
  
 @Override  
 public String obtenerDescripcion(String nombre) throws RemoteException {  
 return planetas.getOrDefault(nombre, new Planeta("", "Planeta no encontrado", "", "")).getDescripcion();  
 }  
  
 @Override  
 public String obtenerTamano(String nombre) throws RemoteException {  
 return planetas.getOrDefault(nombre, new Planeta("", "", "Planeta no encontrado", "")).getTamano();  
 }  
  
 @Override  
 public String obtenerTemperatura(String nombre) throws RemoteException {  
 return planetas.getOrDefault(nombre, new Planeta("", "", "", "Planeta no encontrada")).getTemperatura();  
 }  
}

**ServidorRMI**

import java.rmi.registry.LocateRegistry;  
import java.rmi.registry.Registry;  
  
class ServidorRMI {  
 public static void main(String[] args) {  
 try {  
 Registry registro = LocateRegistry.*createRegistry*(2021);  
 registro.rebind("Constelaciones", new ConstelacionRMI());  
 registro.rebind("Planetas", new PlanetaRMI());  
 System.*out*.println("Servidor RMI en ejecución con Constelaciones y Planetas.");  
 } catch (Exception e) {  
 e.printStackTrace();  
 }  
 }  
}

**Ejecución**

**Servidor**

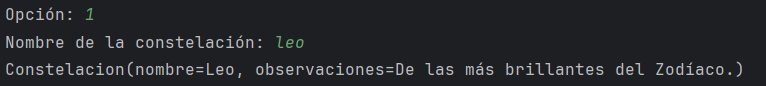
**Texto

El contenido generado por IA puede ser incorrecto.**

**Cliente**

**Texto

El contenido generado por IA puede ser incorrecto.**

****

**Texto

El contenido generado por IA puede ser incorrecto.**

**Interfaz de usuario gráfica, Texto

El contenido generado por IA puede ser incorrecto.**

**Interfaz de usuario gráfica, Texto

El contenido generado por IA puede ser incorrecto.**

**Interfaz de usuario gráfica, Texto

El contenido generado por IA puede ser incorrecto.**

**Enalce a Github**

<https://github.com/Ivannunezrodriguez/Programacion_de_Servicios_y_Procesos_24-25/tree/d2c63453b605c9b149ade9731105e2200988ed0f/Actividades/tema_2/Actividad%20UF3-1>